PROCEEDINGS


October 28 - November 1, 2016

The Twelfth Conference of
The Association for Machine Translation
in the Americas

http://www.amtaweb.org/amta-2016-in-austin-tx

# VOLUME 1:
# MT Researchers' Track

**Editors:**
**Spence Green & Lane Schwartz**

AMTA
Association for Machine Translation in the Americas

AMTA 2016

October 28 – November 1, 2016 -- Austin, TX, USA

*Proceedings of*

# *AMTA 2016,*
# *Vol. 1: MT Researchers' Track*

Spence Green & Lane Schwartz, Eds.


Association for Machine Translation in the Americas
http://www.amtaweb.org

# Introduction

AMTA occupies a unique place among MT research conferences, with one eye trained on the continuing development of novel research techniques and another on the practical application of those techniques in the commercial and government arenas. This year's research program reflects that diversity. This year we accepted 15 papers for oral presentation out of a total of 29 submissions. The accepted papers cover a nice variety of topics including low resource and interactive MT, post-editing, and domain adaptation. Over the past two years, the machine translation research landscape has undergone a rapid transition away from the phrase-based techniques that have been dominant for the past 10+ years, and towards new methods using artificial neural networks. This year's program reflects that trend, with a number of papers applying neural techniques across a broad range of MT tasks.

As chairs of the research track, we would like to thanks all of the authors as well as the many reviewers whose hard work enabled this conference. It has also been a great pleasure to work with George Foster and the other members of the AMTA 2016 organizing committee.

Please enjoy this year's AMTA research track.

Lane Schwartz
Spence Green

## Research Program Committee

Yaser Al-Onaizan
Tim Anderson
Amittai Axelrod
Wilker Aziz
Graeme Blackwood
Marine Carpuat
Daniel Cer
Boxing Chen
Colin Cherry
David Chiang
Steve DeNeefe
Loic Dugast
Nadir Durrani
Marcello Federico
Minwei Feng
Mikel Forcada
George Foster
Kevin Gimpel
Yvette Graham
Spence Green
Eva Hasler
Yifan He
Ulf Hermjakob
Hieu Hoang
Fei Huang
Matthias Huck
Marcin Junczys-Dowmunt
Philipp Koehn
Roland Kuhn
Shankar Kumar
Alon Lavie
Gregor Leusch
Lemao Liu
Qun Liu
Saab Mansour
Daniel Marcu
Arne Mauser
Arul Menezes
Haitao Mi
Michael Pust
Baskaran Sankaran
Rico Sennrich
Michel Simard
Raymond Slyh
Jörg Tiedemann
Christoph Tillmann
Taro Watanabe
Andy Way
Philip Williams
Dekai Wu
François Yvon
Bing Zhao

## Contents


1 Instance Selection for Online Automatic Post-Editing in a Multi-domain Scenario

Rajen Chatterjee, Mihael Arcan, Matteo Negri and Marco Turchi

16 Machine Translation Quality and Post-Editor Productivity

Marina Sanchez-Torron and Philipp Koehn

27 Fuzzy-match repair using black-box machine translation systems: what can be expected?

John Ortega, Felipe Sánchez-Martínez and Mikel L. Forcada

40 Fast, Scalable Phrase-Based SMT Decoding

Hieu Hoang, Nikolay Bogoychev, Lane Schwartz and Marcin Junczys-Dowmunt

53 An Effective Diverse Decoding Scheme for Robust Synonymous Sentence Translation

Youngki Park, Hwidong Na, Hodong Lee, Jihyun Lee and Inchul Song

65 Ranking suggestions for black-box interactive translation prediction systems with multilayer perceptrons

Daniel Torregrosa, Juan Antonio Pérez-Ortiz and Mikel L. Forcada

79 Multi-domain Adaptation for Statistical Machine Translation Based on Feature Augmentation

Kenji Imamura and Eiichiro Sumita

93 Bilingual Methods for Adaptive Training Data Selection for Machine Translation

Boxing Chen, Roland Kuhn, George Foster, Colin Cherry and Fei Huang

107 Neural Interactive Translation Prediction

Rebecca Knowles and Philipp Koehn

121 Guided Alignment Training for Topic-Aware Neural Machine Translation

Wenhu Chen, Evgeny Matusov, Shahram Khadivi and Jan-Thorsten Peter

135 Improving Neural Machine Translation on resource-limited pairs using auxiliary data of a third language

Ander Martínez and Yuji Matsumoto

149 Which Words Matter in Defining Phrase Reorderings in Statistical Machine Translation?

Hamidreza Ghader and Christof Monz

163 Translation of Unknown Words in Low Resource Languages

Biman Gujral, Huda Khayrallah and Philipp Koehn

177 Automatic Construction of Morphologically Motivated Translation Models for Highly Inflected, Low-Resource Languages

John Hewitt, Matt Post and David Yarowsky

191 Investigating the Impact of Various Partial Diacritization Schemes on Arabic-English Statistical Machine Translation

Sawsan Alqahtani, Mahmoud Ghoneim and Mona Diab

# Instance Selection for Online Automatic Post-Editing in a Multi-domain Scenario

**Rajen Chatterjee** chatterjee@fbk.eu
University of Trento, Trento, Italy
Fondazione Bruno Kessler, Trento, Italy

**Mihael Arcan** mihael.arcan@insight-centre.org
Insight Centre for Data Analytics
National University of Ireland, Galway, Ireland

**Matteo Negri** negri@fbk.eu
Fondazione Bruno Kessler, Trento, Italy

**Marco Turchi** turchi@fbk.eu
Fondazione Bruno Kessler, Trento, Italy

**Abstract**

In recent years, several end-to-end online translation systems have been proposed to successfully incorporate human post-editing feedback in the translation workflow. The performance of these systems in a multi-domain translation environment (involving different text genres, post-editing styles, machine translation systems) within the automatic post-editing (APE) task has not been thoroughly investigated yet. In this work, we show that when used in the APE framework the existing online systems are not robust towards domain changes in the incoming data stream. In particular, these systems lack in the capability to learn and use domain-specific post-editing rules from a pool of *multi-domain* data sets. To cope with this problem, we propose an online learning framework that generates more reliable translations with significantly better quality as compared with the existing online and batch systems. Our framework includes: *i)* an instance selection technique based on information retrieval that helps to build domain-specific APE systems, and *ii)* an optimization procedure to tune the feature weights of the log-linear model that allows the decoder to improve the post-editing quality.

## 1 Introduction

Nowadays, machine translation (MT) is a core element in the computer-assisted translation (CAT) framework. The motivation for integrating MT in the CAT framework lies in its capability to provide useful suggestions for unseen segments, which helps to increase the translators' productivity. However, it has been observed that MT is often prone to systematic errors that human post-editing has to correct before publication. The by-product of this "translation as post-editing" process is an increasing amount of parallel data consisting of MT output on one side and its corrected version on the other side. This data can be leveraged to develop automatic post-editing (APE) systems capable not only to spot recurring MT errors, but also to correct them. Thus, integrating an APE system inside the CAT framework can further improve the

quality of the suggested segments, reduce the workload of human post-editors and increase the productivity of the translation industry. As pointed out in (Parton et al., 2012) and (Chatterjee et al., 2015b), from the application point of view APE components would make it possible to:

- Improve the MT output by exploiting information unavailable to the decoder, or by performing deeper text analysis that is too expensive at decoding stage;
- Cope with systematic errors of an MT system whose decoding process is not accessible;
- Provide professional translators with improved MT output quality to reduce (human) PE effort;
- Adapt the output of a general-purpose MT system to the lexicon/style requested in a specific application domain.

In the last decade several works have shown that the quality of the machine translated text can be improved significantly by post-processing the translations with an APE system (Simard et al., 2007a; Dugast et al., 2007; Terumasa, 2007; Pilevar, 2011; Béchara et al., 2011; Chatterjee et al., 2015b, 2016). These systems mainly follow the phrase-based machine translation approach where the MT outputs (with optionally the source sentence) are used as the source language corpus and the post-edits are used as the target language corpus. A common trait of all these APE systems is that they were developed in a batch mode, which consists of training the models over a batch of parallel sentences, optimizing the parameters over a development set, and then decoding the test data with the tuned parameters. Although these standard approaches showed promising results, they lack the ability to incorporate human feedback in a real-time translation workflow. This led to the development of online learning algorithms that can leverage the continuous streams of data arriving in the form of human post-editing feedback to dynamically update the models and tune the parameters on-the-fly within the CAT framework. In recent years, several online systems have been proposed in MT (see Section 2 for more details) to address the problem of incremental training of the models or on-the-fly optimization of feature weights. Few online MT systems have also been applied to the APE scenario (Simard and Foster, 2013; Lagarda et al., 2015) in a controlled working environment in which the systems are trained and evaluated on homogeneous/coherent data where the training and test sets share similar characteristics. Moving from this controlled lab environment to real-world translation workflow, where training and test data can be produced by different MT systems, post-edited by various translators and belong to several text genres, makes the task more challenging, because the APE systems have to adapt to all these diversities in real-time. We define this scenario as a *multi-domain* translation environment (MDTE), where a domain is made of segments belonging to the same text genre and the MT outputs are generated by the same MT system. To reproduce this scenario, in our experiments we run the online APE systems on the concatenation of two datasets belonging to different domains.

A preliminary evaluation in the MDTE scenario reveals that online systems are not robust enough to learn and adapt towards the dynamics of the data, mainly because they try to leverage all the seen data without considering the peculiarities of each domain. In the long-run, these systems tend to become more and more generic, which may not be useful and even harmful to automatically post-edit domain-specific segments. To address this problem, for the first time, we propose an online APE system that is able to efficiently work in a MDTE scenario. Our intuition is that an online APE model trained with few but relevant data (with respect to the segment to be post-edited) can be more reliable than using all the available data *as-is*. To validate this intuition, we propose an online APE system based on an instance selection (IS) technique that is able to retrieve the most relevant training instances from a pool of multi-domain data for each

segment to post-edit. The selected data are then used to train and tune the APE system on-the-fly. The relevance of a training sample is measured by a similarity score that takes into account the context of the segment to be post-edited. This technique allows our online APE system to be flexible enough to decide if it has the correct knowledge for post-editing a sentence or if it is safer to keep the MT output untouched, avoiding possible damages of correction made with insufficient/unreliable knowledge. The results of our experiments with various data sets show that our online learning approach based on IS is: *i)* able to outperform the batch and the other online APE techniques in the single domain scenario, and *ii)* robust enough to work in a MDTE to generate reliable post-edits with significantly better performance than the existing online APE systems.

## 2 Online Translation Systems

Online translation systems aim to incorporate human post-editing feedback (or the corrected version of the MT output) into their models in real-time, as soon as it becomes available. This feedback helps the system to learn from the mistakes made in the past translations and to avoid repeating them in future translations. This continuous learning capability will eventually improve the quality of the translations and consequently increase the productivity of the translators/post-editors (Tatsumi, 2009) working with MT suggestions in a CAT environment. The basic workflow of an online translation system goes through the following steps repeatedly: *i)* the system receives an input segment; *ii)* the input segment is translated and provided to the post-editor to fix any errors in it; and *iii)* the human post-edited version of the translation is incorporated back into the system, by stepwise updating the underlying models and parameters. In the APE context, the input is a machine-translated segment (optionally with its corresponding source segment), which is processed by the online APE system to fix errors, and then verified by the post-editors. Several online translation systems have been proposed over the years (Hardt and Elming, 2010; Bertoldi et al., 2013; Mathur et al., 2013; Simard and Foster, 2013; Ortiz-Martınez and Casacuberta, 2014; Denkowski et al., 2014; Wuebker et al., 2015, inter alia). In this section, we describe two online systems that have been used in the APE task (PEPr, and Thot), and one in the MT scenario which is similar to our proposed system (Realtime cdec):

**PEPr: Post-Edit Propagation:** Simard and Foster (2013) proposed a method for post-edit propagation (PEPr), which learns post-editors' corrections and applies them on-the-fly to further MT output. Their proposal is based on a phrase-based SMT system, used in an APE setting with online learning mechanism. To perform post-edit propagation, this system was trained incrementally using pairs of machine-translated *(mt)* and human post-edited *(pe)* segments as they were produced. When receiving a new pair *(mt, pe)*, word alignments are obtained by using Damerau-Levenshtein distance. In the next step the phrase pairs are extracted and appended to the existing phrase table. The whole process is assumed to take place within the context of a single document. For every new document the APE system begins with an "empty" model. Since the post-editing rules are learned for a given document they can be more precise and useful for that document, but the limitation is that knowledge gained after processing one document is not utilized for other similar documents. This limitation can be addressed by our system (Section 3), in which we maintain one global knowledge base to store all the processed documents, still being able to retrieve post-editing rules specific to a document to be translated.

**Thot:** The Thot toolkit (Ortiz-Martınez and Casacuberta, 2014) is developed to support fully automatic and interactive statistical machine translation.[1] It was also used by Lagarda et al. (2015) in an online setting for the APE task, to perform large-scale experiments with several

[1] `https://github.com/daormar/thot`

data sets for multiple language pairs, with base MT systems built using different technologies (rule-based MT, statistical MT). In the majority of their experiments online APE successfully improved the quality of the translations obtained from the base MT system by a significant margin. To update the underlying translation and language models with the user feedback, a set of sufficient statistics was maintained that can be incrementally updated. In the case of language model, only the n-gram counts are required to maintain sufficient statistics. To update the translation model, an incremental version of EM algorithm is used to first obtain word alignment and then phrase pairs counts were extracted to update the sufficient statistics. Other features like source/target phrase-length models or distortion model are implemented by means of geometric distributions with fixed parameters. The sentence length model is implemented by means of Gaussian distributions. However, the feature weights of the log-linear model are static throughout the online learning process, as opposed to our method that updates the weights on-the-fly. Also, this method learns post-editing rules from all the data processed in real-time, whereas, our approach learns from the most relevant data points.

**Realtime cdec:** Denkowski et al. (2014) proposed an online model adaptation method to leverage human post-edited feedback to improve the quality of an MT system in a real-time translation workflow. To build the translation models they use a static suffix array (Zhang and Vogel, 2005) to index initial data (or a seed corpus), and a dynamic lookup table to store information from the post-edited feedback. To decode a sentence, the statistics of the translation options are computed both from the suffix array and from the lookup table. An incremental language model is maintained and updated with each incoming human post-edit. To update the feature weights they used an extended version of the margin-infused relaxed algorithm (MIRA) (Chiang, 2012). The decoding is treated as simply the next iteration of MIRA, where a segment is first translated and then its corresponding reference/post-edition is provided to the model, and MIRA updates the parameters. While this system was earlier used in the context of MT, in this work we use it to investigate its applicability in online APE. A key difference between this approach and ours is the sampling technique. The former uses suffix-arrays to always retrieve the top $k$ source phrases, whereas in our approach the number of samples (or the training instances) is dynamically set to use only the most relevant ones. Another difference is visible in the parameter optimization step. Realtime cdec optimizes the feature weights of the log-linear model after decoding each segment, whereas, our method optimizes the weights specifically for the segment to be post-edited.

## 3 Instance Selection for Online APE System

The online systems described in Section 2 compute and update the feature scores of the log-linear models based on all the previously seen data. This indicates that, in the long-run, the model will tend to become more and more generic, since the data processed in the online scenario may belong to multiple domains as explained in Section 1. Having a generic model might not be useful to retrieve the domain-specific post-editing rules needed to fix errors in a particular document. One solution is to build document-specific APE models as proposed by Simard and Foster (2013). In their approach, however, once the entire document is processed the models are reset back to their original state, due to which the knowledge gained from the current document is lost. To preserve all the knowledge gained in the online learning process, at the same time being able to apply specific post-editing rules when needed, we propose an instance selection technique for online APE. Our proposed framework, as shown in Figure 1, uses a global knowledge base to preserve all the data points seen in the online process, and has the ability to retrieve specific data points whose context is similar to the segment to be post-edited. These data points are used to build reliable APE models. When there are no reliable data points in the knowledge base, the MT output is kept untouched, as opposed to the existing APE systems, which tend to

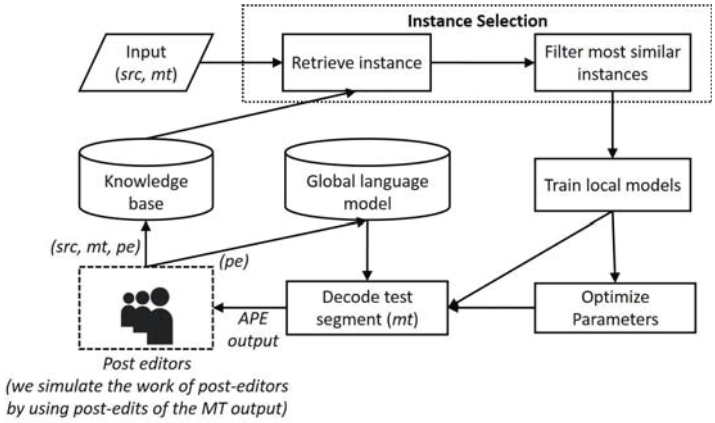



Figure 1: Architecture of our online APE system

always "translate" the given input segment independently from the reliability of the applicable correction rules. This approach of post-editing with reliable information only makes our system more precise compared with others (see results in Section 5): that is when a post-editing rule is applied it is more likely to improve the quality of the translation. When no reliable knowledge is available for the correction, the MT output is left untouched.

We propose online APE but we actually "emulate" it by processing the data points one at a time. Our proposed algorithm assumes to have the following data to run the online experiments: *i)* source (*src*); *ii)* MT output (*mt*); and *iii)* human post-edits (*pe*) of the MT output. At the beginning the knowledge base of our online APE system is empty and it will be updated whenever a new instance (a tuple containing parallel segments from all the above mentioned documents) is processed. When the system receives an input (*src*, *mt*), it proceeds through the following steps:

**Instance Selection.** Initially, it selects the most relevant training instances from a pool of *multi-domain* data stored in our knowledge base. This will help to build a reliable APE model for each input segment processed in real-time. The relevance of the training instances with respect to the input segment is measured in terms of a similarity score based on the term frequency—inverse document frequency (*tf-idf*), generally used in information retrieval. The larger the number of words in common between the training and the input sentences, the higher is the score. In our system, these scores are computed using the Lucene library.[2] Only those training instances that have a similarity score above a certain threshold (decided over a held-out development set) are used to build the system. In case there are no training instances available, we preserve the input segment as it is. Indeed, we assume that APE with unreliable information can damage the *mt* segment instead of improving the translation quality. This is one of the main outcomes of the first APE pilot task organized last year within the WMT initiative (Bojar et al., 2015) and, as we will see from our results, it represents a major problem for the approaches that always translate the given input segments. The proposed instance selection technique (or sampling mechanism) differs from the one proposed in real-time cdec (Denkowski et al., 2014), which uses suffix-arrays to select the top $k$ instances. In our approach the sample size is in fact dynamically set in order to select only the most similar ones. This allows us to build more reliable models (since the underlying data better resembles the test segment), and to gain speed

[2] `https://lucene.apache.org/`

when the sample size is small. The use of a *tf-idf* similarity measure was proposed before in the context of machine translation by Hildebrand et al. (2005) to create a pseudo in-domain corpus from a big out-of-domain corpus. Our work is the first to investigate it for the APE task in an online learning scenario.

**Model Creation.** From the selected instances we build several local models. The first is the language model: A tri-gram local language model is built over the target side of the training corpus with the IRSTLM toolkit (Federico et al., 2008). Since the selected training data closely resembles the input segment, we believe that the local LM can capture the peculiarities of the domain to which the input segment belongs. Along with the local LM we always use a tri-gram global LM, which is updated whenever a human post-edition (*pe*) is received. The other local models are the translation and the reordering models: these local models are built over the training instances retrieved from the knowledge base. Since the training instances are very similar to the input segment, the post-editing rules learned from these local models are more reliable for the test segment. These models are build with the Moses toolkit (Koehn et al., 2007) and the word alignment of each sentence pair is computed using the incremental GIZA++ software.[3]

**Parameter Optimization.** The parameters are optimized over a section of the selected instances (development set). The size of this development set is critical: if it is too large, then the parameter optimization will be expensive. On the other hand, if it is too small the tuned weights might not be reliable. To achieve fast optimization with reliably-tuned weights, multiple instances of MIRA are run in parallel on several small development sets and all the resulting weights are then averaged. For this purpose, the data selected by the instance selection module are randomly split in training and development sets three times. A minimum number of selected sentence pairs is required to trigger the parameter optimisation process. If this minimum value is not reached, the optimization step is skipped because having few sentences might not yield to reliable weights. In this case, the weights computed on the previous input segment are used. In our experiments, we observed that this solution is more reliable and efficient than the feature weights obtained with a single tuning, as it was previously proposed in (Cettolo et al., 2011). We believe this procedure to optimize the feature weights over a development set that closely resembles the test segment can help to obtain weights more suitable to the segment to be post-edited.

**Decode Test Segment.** To decode the input segments, all the local models (language, translation, reordering) are built with all the selected instances. The log-linear feature weights are computed by taking the arithmetic mean of the tuned weights for the three data splits. The decoding process is performed with the Moses toolkit recalling that the input segment is kept untouched when no reliable information is available in the knowledge base.

**Update Global Repository.** In a real translation workflow, the automatically post-edited version (or the MT output, if there were no training data available) is provided to a post-editor for correction, and the corrected version is incorporated back into the system. To avoid the unnecessary costs of involving human post-editors in-the-loop when running these experiments, we simulate this condition by using the human post-edits of the MT output (which are already available in the data set). Each newly processed instance is added to our knowledge base, and the global language model is updated with the post-edited segment.

[3] `https://code.google.com/archive/p/inc-giza-pp/`

## 4 Experimental Setup

### 4.1 Data

To examine the performance of the online APE systems in a multi-domain translation environment, we select two data sets for the English-German language pair belonging to the information technology (IT) domain. Although they come from the same domain (IT), they feature variability in terms of vocabulary coverage, MT errors, and post-editing style. The two data sets are respectively a subset of the Autodesk Post-Editing Data corpus [4] and the resources used at the second round of the APE shared task at the First Conference on Machine Translation (WMT2016) (Bojar et al., 2016).[5] The data sets are pre-processed to obtain a joint-representation that links each source word with a MT word (*mt#src*). This representation has been proposed in the context-aware APE approach by Béchara et al. (2011) and leverages the source information to disambiguate post-editing rules. Recently, Chatterjee et al. (2015b) also confirmed this approach to work better than translating from raw MT segments over multiple language pairs. The joint-representation is used as a source corpus to train all the APE systems reported in this paper and it is obtained by first aligning the words of source (*src*) and MT (*mt*) segments using MGIZA++ (Gao and Vogel, 2008), and then each *mt* word is concatenated with its corresponding *src* words.

The Autodesk training, development, and test sets consist of 12,238, 1,948, and 1,956 segments respectively, while the WMT2016 data contains 12,000, 1,000, and 2,000 segments. Table 1 provides some additional statistics of the source (*mt#src*) and target (*pe*) training corpus, the repetition rate (RR) to measure the repetitiveness inside a text (Bertoldi et al., 2013), and the average TER score for both the data sets (computed between MT and PE). It is interesting to note that the Autodesk data set has on average shorter segments compared with the WMT2016 corpus. This suggests that learning and applying post-editing rules in the Autodesk corpus can be easier than using the WMT2016 segments, because dealing with long segments generally increases the complexity of the rules extraction and decoding processes. Moreover, the WMT2016 data set has a repetition rate similar to the Autodesk even though it has more tokens. This indicates that the data is more sparse raising the difficulty of extracting reliable post-editing rules. Looking at the TER score, the smaller value of the WMT2016 data set compared with the Autodesk one suggests that the room for improvement is lower, because there are less corrections to perform and the chance to deteriorate the original MT output is larger.

| | Tokens | | Types | | Avg. segment length | | RR | TER |
|---|---|---|---|---|---|---|---|---|
| | *mt#src* | *pe* | *mt#src* | *pe* | *mt#src* | *pe* | (*mt#src*) | |
| Autodesk | 153,943 | 160,801 | 31,939 | 15,023 | 12.57 | 13.13 | 4.938 | 45.35 |
| WMT2016 | 210,573 | 214,720 | 32211 | 16,388 | 17.54 | 17.89 | 4.907 | 26.22 |

Table 1: Data statistics

The diversity of the two data sets is further measured by computing the vocabulary overlap between the two joint-representations. This is performed internally to each data set (splitting the training data in two halves) and across them. As expected, in the first case the vocabulary overlap is much larger ($> 40\%$) than in the second one ($\sim 15\%$), and this indicates that the two data sets are quite different and few information can be shared. All the aforementioned aspects show the large variability in the corpora making them suitable to emulate the multi-domain translation environment.

[4] https://autodesk.app.box.com/v/autodesk-postediting

[5] http://www.statmt.org/wmt16/ape-task.html

### 4.2 Evaluation metrics

The performance of the different APE systems is evaluated using three different metrics: Translation Error rate (TER) (Snover et al., 2006), BLEU (Papineni et al., 2002) and Precision (Chatterjee et al., 2015a). TER and BLEU measure the similarity between the MT outputs and their references by looking at n-grams overlap (TER at word level, BLEU from 1 to 4 words). To give a better insight on the APE performance, we also report Precision, computed as the ratio of the number of sentences an APE system improves (with respect to the MT output) over all the sentences it modifies.[6] Values larger than 50% indicate that the APE system is able to improve the quality of most of the sentences it changes.

Statistical significance tests are computed using the paired bootstrap resampling technique (Koehn, 2004) for the BLEU metric and the stratified approximate randomization test (Clark et al., 2011) for TER.

### 4.3 Terms of comparison

We evaluate our online learning approach against four different terms of comparison.

**MT.** Our baseline is the *"do-nothing"* system that simply returns the MT outputs without changing them. As discussed in (Bojar et al., 2015), this baseline can be particularly hard to beat when the repetition rate of the data is low and due to the tendency of the APE systems to over-correct the MT output.

**Batch APE.** This APE system is developed in a batch mode following the approach proposed in Chatterjee et al. (2015b). It is similar to the context-aware method (Béchara et al., 2011), but it uses word alignments produced by the monolingual machine translation APE technique proposed in Simard et al. (2007b). Being a batch method, it cannot learn from the test set, but it leverages all the training points at the same time.

**Online APE.** We compare our approach against two online systems: *i)* the Thot toolkit that had been previously used in the online APE task, and *ii)* Realtime cdec that, among the other online MT systems, is the closest to our approach (*i.e.* it uses a data selection mechanism), but has never been tested in the APE scenario. Another online APE approach is PEPr that was meant for document level APE, but since we are working with data sets that do not have any intrinsic document structure, we do not find it to be a suitable term of comparison.

## 5 Experiments and Results

Our preliminary objective is to examine if the online learning methods are able to achieve results that are competitive with those of batch methods, which are potentially favored by the possibility to leverage all the training data at the same time. For this test, all the algorithms are evaluated in the classic *in-domain* setting, where training, development, and test sets are sampled from the same data set or domain. All the online APE methods are run in two modes; *i)* batch: the test set is not used in the learning process (to have a fair comparison with the batch APE), *ii)* online: the test set is leveraged in the online learning process. The experiments are performed for both the data sets (Autodesk and WMT2016), and their corresponding results are reported in Table 2 and Table 3 respectively. The parameters of our approach (*i.e.* similarity score threshold and minimum number of selected sentence) are optimised on the development set following a grid search strategy. We set the threshold values to 0.8 and 1 respectively for the Autodesk and WMT2016 datasets and the minimum number of selected sentences to 20.

[6] For each sentence in the test set, if the TER score of the APE output is different than the TER score of the MT then the sentence is considered as a modified sentence

| | Batch mode | | | Online mode | | |
|---|---|---|---|---|---|---|
| | BLEU | TER | Precision (%) | BLEU | TER | Precision (%) |
| MT | 39.28 | 46.48 | N/A | N/A | N/A | N/A |
| Batch APE | 44.14 | 43.24 | 61.34 | N/A | N/A | N/A |
| cdec | 43.13$^{\dagger}$ | 43.86$^{\dagger}$ | 54.22 | 43.19$^{\dagger}$ | 43.69$^{\dagger}$ | 54.75 |
| Thot | 43.21$^{\dagger}$ | 44.70$^{\dagger}$ | 55.69 | 43.34$^{\dagger}$ | 44.62$^{\dagger}$ | 56.27 |
| Our approach | **44.68**$^{\dagger}$ | **41.98**$^{\dagger}$ | **79.26** | **44.76**$^{\dagger}$ | **41.95**$^{\dagger}$ | **79.20** |

Table 2: Autodesk in-domain ($^{\dagger}$: statistically significant wrt. Batch APE with $p<0.05$)

| | Batch mode | | | Online mode | | |
|---|---|---|---|---|---|---|
| | BLEU | TER | Precision (%) | BLEU | TER | Precision (%) |
| MT | 62.11 | 24.76 | N/A | N/A | N/A | N/A |
| Batch APE | 63.06 | 25.07 | 48.55 | N/A | N/A | N/A |
| cdec | 61.99$^{\dagger}$ | 25.26 | 45.17 | 61.80$^{\dagger}$ | 25.35$^{\dagger}$ | 42.83 |
| Thot | 62.06$^{\dagger}$ | 25.26 | 42.92 | 62.22$^{\dagger}$ | 25.22 | 43.69 |
| Our approach | **62.97** | **24.53**$^{\dagger}$ | **61.46** | **63.19** | **24.39**$^{\dagger}$ | **62.62** |

Table 3: WMT2016 in-domain ($^{\dagger}$: statistically significant wrt. Batch APE with $p<0.05$)

From the results of the in-domain experiments with the Autodesk data set it is evident that our proposed online APE method performs not only better than *cdec* and *Thot* (both in batch and online mode) but also better than the strong batch APE method. It achieves significant improvements of 0.54 BLEU, 1.26 TER, and 17.9% precision over the batch APE, which already beats the other online methods. The improvement of our system can be attributed to its ability to learn from the most relevant data and to avoid over-correction by leaving the test segment untouched when no reliable information is found in the knowledge base. As discussed in Section 4.1, several factors like sentence length, sparsity, and translation quality make the WMT2016 data set more challenging to improve for all the online APE methods. In particular, due to the higher translation quality of the *mt* segments, the room for improvement gets lower and the chances of damaging the correct parts are higher. This is visible from the low precision scores reported in Table 3. All the APE methods (batch and online) damage the MT segments in the majority of the cases (precision is lower than 50%). The only exception is our approach that performs significantly better than the batch APE (in terms of TER) and is the only successful method to significantly improve the MT segments in the majority of the cases (61.46%). These experimental results confirm that our proposed online learning APE method based on instance selection to learn only from the most relevant data is sound and reliable.

Building on these results, the main goal of this research is to examine the performance of the online APE methods in a MDTE. This represents a more challenging condition since the system has to adapt to the dynamics of the data processed in a real-time scenario. To emulate this environment, all the online learning methods are trained and tuned on one data set (or domain) and evaluated on the other data set with the possibility to learn from it. In order to capture the peculiarities of the online learning methods over a long run with many data points, we use the training section of the second data set as a test set. The left side of Table 4 reports the performance of all the APE systems when they are trained and tuned on the WMT2016 data set and evaluated on the Autodesk data set. The experimental results reported in the right side of Table 4 are obtained by using the Autodesk data set to train and tune, and the WMT2016 to evaluate. The parameters of our approach (*i.e.* similarity score threshold and minimum number of selected sentence) are the same as computed in the in-domain setting.

| | WMT2016 - Autodesk | | | Autodesk - WMT2016 | | |
|---|---|---|---|---|---|---|
| | BLEU | TER | Precision (%) | BLEU | TER | Precision (%) |
| MT | 39.91 | 45.35 | N/A | 60.90 | 26.22 | N/A |
| Batch APE | 38.09$^{\dagger}$ | 46.91$^{\dagger}$ | 3.95 | 55.56$^{\dagger}$ | 30.03$^{\dagger}$ | 4.03 |
| cdec | 38.63$^{\dagger}$ | 46.26$^{\dagger}$ | 8.36 | 56.30$^{\dagger}$ | 28.98$^{\dagger}$ | 7.37 |
| Thot | 42.40$^{\dagger}$ | 43.45$^{\dagger}$ | 58.46 | 58.11$^{\dagger}$ | 28.67$^{\dagger}$ | 14.20 |
| Our approach | **43.59**$^{\dagger}$ | **42.44**$^{\dagger}$ | **76.38** | **60.49**$^{\dagger}$ | **26.44**$^{\dagger}$ | **41.37** |

Table 4: Performance of the APE systems in a multi-domain translation environment. ($^{\dagger}$: statistically significant wrt. MT, $p<0.05$; the best scores among the online systems are bold)

In Table 4, the poor performance of the batch APE, which can only leverage the knowledge from the training domain, indicates that the post-editing rules extracted from the training domain are not portable to the test one (even though both datasets belong to IT). This suggests the need of APE approaches that are able to adapt themselves to the incoming data in real-time. Comparing the performance of all the online approaches for both test sets, we notice that our system performs the best with significant gains in all the evaluation metrics. This confirms that our APE system, based on instance selection, is robust enough to work in a MDTE due to its capability to leverage only the most relevant information from a pool of multi-domain segments. Similar to the results on in-domain experiments, significant gains in performance are observed for the Autodesk test set. This does not happen for the WMT2016 test data, for which none of the online APE approaches is able to improve over the MT baseline. For this challenging data set, our approach has the minimal performance degradation (over MT), while the other online systems severely damage the MT segments as confirmed by their low precision (7.37% and 14.20% respectively). One of the common observations, both over the Autodesk and the WMT2016 test sets, is the large difference in precision (17.92% and 27.17% respectively) between the best (our approach) and the second best (Thot) online APE system. This indicates that our approach is more conservative and more suitable to extract and apply domain-specific post-editing rules from a pool of multi-domain data sets, which makes it a more viable and appropriate solution to be deployed in a real-world CAT framework. In the next section, we present some findings on the performance trends of different systems across the entire test set for the *multi-domain* scenario.

## 6 Performance Analysis

To understand and compare the behavior of different online learning approaches in the long-run, the plot in Figure 2 shows the moving average TER (window of 750 data points) at each segment of the Autodesk test set for the multi-domain experiment (Table 4). As it can be seen, our approach successfully maintains the best performance across the entire test set. As expected, at the beginning of the test set the performance of the online systems is close to the MT system, since there is not much relevant data available to learn from. As time progresses and more segments are processed, a clear trend of performance improvement (with respect to MT) is visible for our method and for the Thot system. This does not hold in the case of cdec, maybe due to the sampling techniques used in the suffix array, which is unable to retrieve relevant samples from the pool of multi-domain data to decode the test segments.

For the WMT2016 test set the moving average TER is shown in Figure 3. As said before, improving translation quality on this test set is more challenging, which is reflected in the graph. Although none of the systems is able to improve over the MT baseline, our system manages to consistently stay close to the MT performance throughout the test set, whereas, all other systems show significant drops. This ensures that our approach is more robust against the domain-shift

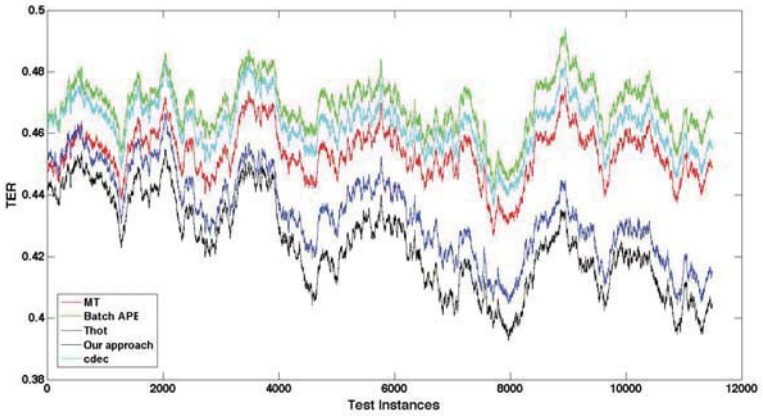



Figure 2: Moving average TER for the Autodesk test set in a *multi-domain* scenario

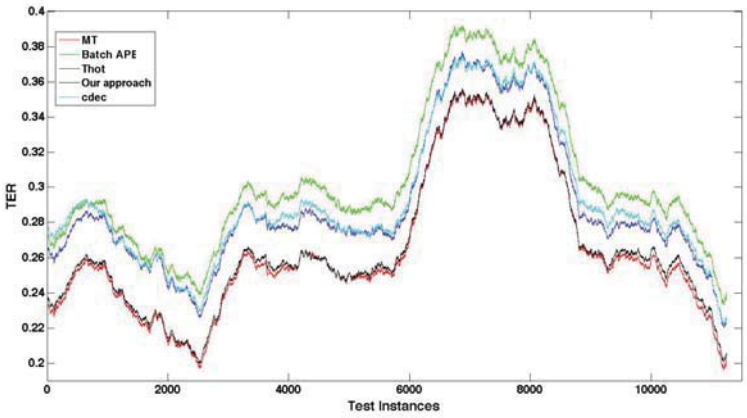



Figure 3: Moving average TER for the WMT2016 test set in a multi-domain scenario

and even in this difficult scenario it is able to maintain stable performance close to the MT without a large deterioration.

To gain further insights about the performance at the segment level, the plot in Figure 4 compares our approach against Thot for the first 300 segments of the Autodesk test set used in the multi-domain experiment. It shows the differences between the segment-level TER of the MT ($TER_{MT}$) and our approach ($TER_{Our\ approach}$), and MT and Thot ($TER_{Thot}$) automatically post-edited segments. We notice that our approach modifies less segments compared with Thot, because it builds a model only if it finds relevant data in the knowledge base, otherwise it leaves the MT segment untouched. These untouched MT segments, when modified by Thot, often lead to deterioration rather than to improvements (as seen by many negative peaks for Thot in the Figure 4). This suggests that, compared with the other online approaches, the output obtained with our solution has a higher potential for being useful to human translators. Such usefulness comes not only in terms of a more pleasant post-editing activity, but also in terms of time savings yield by overall better suggestions.

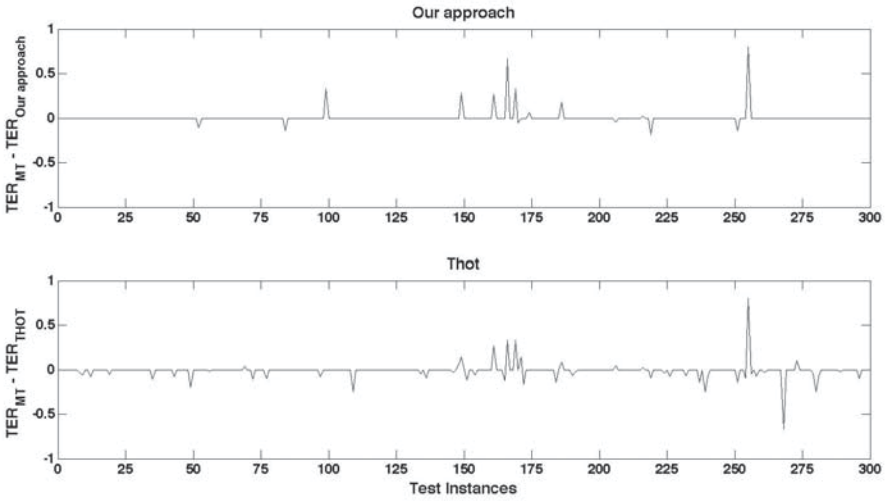



Figure 4: Our approach (top) vs Thot (bottom) performance comparison for the initial test segments ($> 0$ means improvements over the MT, $< 0$ means deterioration of the MT)

## 7 Conclusion

We addressed the problem of building robust online APE systems in a *multi-domain* translation environment in which the system has to continuously adapt to the dynamics of diverse data processed in real-time. Our evaluation revealed that the online systems that leverage all the available data without considering the peculiarities of each domain are not robust enough to work in a multi-domain translation environment, because they are unable to learn domain-specific post-editing rules. To overcome this limitation, we proposed an online learning framework based on instance selection that has the capability to filter out the most relevant information from a pool of multi-domain data for learning domain-specific post-editing rules. When no reliable information is available our system leaves the MT segments untouched, these segments when automatically post-edited by other systems are often found to get deteriorated. Therefore, the APE suggestions provided by our system to the translators/post-editors are more reliable with better translation quality.

From our experiments in a simulated multi-domain environment, we learn that the post-editing rules are not portable across domains which is revealed by the poor performance of the batch APE system that can leverage only the training data. In the case of online systems that leverage also the test set, it was still a challenging scenario (specially for the Autodesk-WMT2016 data set). Among all the online systems, our proposed approach has the highest improvement on the WMT2016-Autodesk data set, and the least degradation on the Autodesk-WMT2016 data set with respect to the MT quality. Experiments in the *in-domain* setting confirmed that our approach for instance selection is also useful in a single domain scenario. It performed significantly better than the batch APE that already beats cdec and Thot. One common observation from all the experiments in different working scenarios and with different data sets is that our system has the highest precision among all its competitors (MT, batch APE, cdec, and Thot). This indicate that when our system automatically post-edits MT segments, it is more likely to improve the quality of the MT output, which makes it a viable solution to be deployed in a real-word CAT framework.

## Acknowledgement

This work has been partially supported by the EC-funded H2020 project QT21 (grant agreement no. 645452), and by the Science Foundation Ireland research grant (no. SFI/12/RC/2289).

## References

Béchara, H., Ma, Y., and van Genabith, J. (2011). Statistical post-editing for a statistical mt system. In *Proceedings of the XIII MT Summit*, pages 308–315.

Bertoldi, N., Cettolo, M., and Federico, M. (2013). Cache-based online adaptation for machine translation enhanced computer assisted translation. *Proceedings of the XIV MT Summit*, pages 35–42.

Bojar, O., Chatterjee, R., Federmann, C., Graham, Y., Haddow, B., Huck, M., Jimeno Yepes, A., Koehn, P., Logacheva, V., Monz, C., Negri, M., Neveol, A., Neves, M., Popel, M., Post, M., Rubino, R., Scarton, C., Specia, L., Turchi, M., Verspoor, K., and Zampieri, M. (2016). Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation*, pages 131–198, Berlin, Germany. Association for Computational Linguistics.

Bojar, O., Chatterjee, R., Federmann, C., Haddow, B., Huck, M., Hokamp, C., Koehn, P., Logacheva, V., Monz, C., Negri, M., Post, M., Scarton, C., Specia, L., and Turchi, M. (2015). Findings of the 2015 workshop on statistical machine translation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 1–46, Lisbon, Portugal.

Cettolo, M., Bertoldi, N., and Federico, M. (2011). Methods for smoothing the optimizer instability in smt. In *Proceedings of the XII MT Summit*, pages 32–39.

Chatterjee, R., C. de Souza, J. G., Negri, M., and Turchi, M. (2016). The fbk participation in the wmt 2016 automatic post-editing shared task. In *Proceedings of the First Conference on Machine Translation*, pages 745–750, Berlin, Germany. Association for Computational Linguistics.

Chatterjee, R., Turchi, M., and Negri, M. (2015a). The fbk participation in the wmt15 automatic post-editing shared task. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 210–215.

Chatterjee, R., Weller, M., Negri, M., and Turchi, M. (2015b). Exploring the planet of the apes: a comparative study of state-of-the-art methods for mt automatic post-editing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 156–161.

Chiang, D. (2012). Hope and fear for discriminative training of statistical translation models. *Journal of Machine Learning Research*, 13(Apr):1159–1187.

Clark, J. H., Dyer, C., Lavie, A., and Smith, N. A. (2011). Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 176–181.

Denkowski, M., Dyer, C., and Lavie, A. (2014). Learning from post-editing: Online model adaptation for statistical machine translation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 395–404.

Dugast, L., Senellart, J., and Koehn, P. (2007). Statistical post-editing on systran's rule-based translation system. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 220–223.

Federico, M., Bertoldi, N., and Cettolo, M. (2008). Irstlm: an open source toolkit for handling large scale language models. In *Proceedings of Interspeech*, pages 1618–1621.

Gao, Q. and Vogel, S. (2008). Parallel implementations of word alignment tool. In *Proceedings of Software Engineering, Testing, and Quality Assurance for Natural Language Processing*, pages 49–57.

Hardt, D. and Elming, J. (2010). Incremental re-training for post-editing smt. In *Proceedings of AMTA*.

Hildebrand, A. S., Eck, M., Vogel, S., and Waibel, A. (2005). Adaptation of the translation model for statistical machine translation based on information retrieval. In *Proceedings of EAMT*, pages 133–142.

Koehn, P. (2004). Statistical significance tests for machine translation evaluation. In *Proceedings of EMNLP*, pages 388–395.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., et al. (2007). Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics. System Demonstrations*, pages 177–180.

Lagarda, A. L., Ortiz-Martínez, D., Alabau, V., and Casacuberta, F. (2015). Translating without in-domain corpus: Machine translation post-editing with online learning techniques. *Computer Speech & Language*, 32(1):109–134.

Mathur, P., Cettolo, M., Federico, M., and Kessler, F.-F. B. (2013). Online learning approaches in computer assisted translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation, ACL*, pages 301–308.

Ortiz-Martınez, D. and Casacuberta, F. (2014). The new thot toolkit for fully-automatic and interactive statistical machine translation. In *14th Annual Meeting of the European Association for Computational Linguistics: System Demonstrations*, pages 45–48.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

Parton, K., Habash, N., McKeown, K., Iglesias, G., and de Gispert, A. (2012). Can Automatic Post-Editing Make MT More Meaningful? In *Proceedings of EAMT*, pages 111–118.

Pilevar, A. H. (2011). Using statistical post-editing to improve the output of rule-based machine translation system. *IJCSC*.

Simard, M. and Foster, G. (2013). Pepr: Post-edit propagation using phrase-based statistical machine translation. In *Proceedings of the XIV MT Summit*, pages 191–198.

Simard, M., Goutte, C., and Isabelle, P. (2007a). Statistical Phrase-Based Post-Editing. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 508–515.

Simard, M., Ueffing, N., Isabelle, P., and Kuhn, R. (2007b). Rule-based translation with statistical phrase-based post-editing. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 203–206.

Snover, M., Dorr, B., Schwartz, R., Micciulla, L., and Makhoul, J. (2006). A study of translation edit rate with targeted human annotation. In *Proceedings of AMTA*, pages 223–231.

Tatsumi, M. (2009). Correlation between automatic evaluation metric scores, post-editing speed, and some other factors. In *Proceedings of the XII MT Summit*, pages 332–339.

Terumasa, E. (2007). Rule based machine translation combined with statistical post editor for japanese to english patent translation. In *Proceedings of the XI MT Summit*, pages 13–18.

Wuebker, J., Green, S., and DeNero, J. (2015). Hierarchical incremental adaptation for statistical machine translation. In *Proceedings of EMNLP*, pages 1059–1065.

Zhang, Y. and Vogel, S. (2005). An efficient phrase-to-phrase alignment model for arbitrarily long phrase and large corpora. In *Proceedings of EAMT*, pages 294–301.

# Machine Translation Quality and Post-Editor Productivity

**Marina Sanchez-Torron** msnc017@aucklanduni.ac.nz
School of Cultures, Languages and Linguistics
University of Auckland
Auckland 1010, New Zealand
**Philipp Koehn** phi@jhu.edu
Department of Computer Science
Johns Hopkins University
Baltimore, MD 21218-2608, USA

**Abstract**

We assessed how different machine translation (MT) systems affect the post-editing (PE) process and product of professional English–Spanish translators. Our model found that for each 1-point increase in BLEU, there is a PE time decrease of 0.16 seconds per word, about 3-4%. The MT system with the lowest BLEU score produced the output that was post-edited to the lowest quality and with the highest PE effort, measured both in HTER and actual PE operations.

## 1 Introduction and Related Work

There is a relatively fair amount of empirical research on post-editing machine translation output (PE) focusing on assessing potential time and quality improvements over human translation with or without translation memories (TM). A common finding is that, despite differences among participants, PE is on average faster than unassisted or TM-assisted translation. Some examples of studies finding such speed benefits are those by Guerberof (2009), Flournoy and Duran (2009), Groves and Schmidtke (2009), Plitt and Masselot (2010) and Skadiņš et al. (2011).

In terms of PE quality, studies have shown, through the use of human judgments, that PE leads to quality comparable to (García, 2010) or better than other types of translation (Guerberof, 2009; Fiederer and O'Brien, 2009; Carl et al., 2011; Green et al., 2013; Läubli et al., 2013). There is therefore strong empirical evidence pointing at the speed and quality benefits of PE.

While many factors may affect productivity in a PE workflow, MT quality is one that is relatively easy to measure. Previous studies to have investigated how MT quality affects PE speed are those by Tatsumi (2009) and O'Brien (2011). They found different levels of correlations between MT quality, measured on a sentence level with automatic metrics, and PE speed. Krings (2001) and De Sutter (2012) too found that human judgments of MT quality correlated to PE speed. Koehn and Germann (2014) found their worst MT system entailed 20% more editing activity than their best one.

To the best of our knowledge, this is the first study to attempt to assess how different MT systems, of the same type but with different quality levels, affect the PE productivity of professional translators. By investigating how MT quality affects both PE time and PE quality, we aim at providing MT users and researchers with another approach to examining the usefulness of MT for PE purposes.

## 2 Study Setup

Nine translators were hired through ProZ[1], self-described as the biggest online translation workplace. Selected participants were offered a fixed compensation based on the standard, general, English–Spanish translation rates displayed on the website. Each translator post-edited four news texts of about 650 words each. Texts had similar complexity levels and were presented to translators in randomized order to dilute possible familiarization or fatigue effects. All four texts were translated into Spanish with nine MT systems (cf. Section 4.1). The output of all nine systems was assigned randomly to participants. As in Cettolo et al. (2013) and Koehn and Germann (2014), to deal with between-participant variability, the following restrictions were implemented:

- All translators post-edited all source sentences.
- No translator post-edited the same source sentence twice.
- All translators were exposed to roughly the same amount of output of all MT systems.

Translators were asked to post-edit to full, human-like quality. They worked remotely on the open-source, web-based, computer aided translation (CAT) tool CASMACAT (Cognitive Analysis and Statistical Methods for Advanced Computer Aided Translation)[2] (Alabau et al., 2013).

## 3 Translator Profile

All participants were native Spanish, professional translators, with at least 2 years' experience using CAT tools. All were educated to college level. Except for TR4 and TR5, all participants had degrees in Translation. Table 1 summarizes their main characteristics.

| Translator | Translation experience (years) | PE certification | PE experience (years) |
|---|---|---|---|
| TR1 | 2 to 5 | No | <2 |
| TR2 | 2 to 5 | No | 2 to 5 |
| TR3 | 5 to 10 | Yes | 5 to 10 |
| TR4 | >10 | No | 2 to 5 |
| TR5 | 5 to 10 | No | None |
| TR6 | >10 | Yes | <2 |
| TR7 | 5 to 10 | No | 5 to 10 |
| TR8 | 2 to 5 | No | 2 to 5 |
| TR9 | >10 | No | 2 to 5 |

Table 1: Translators' background

Translators' perceptions of PE and MT were elicited through eight questions answered with a Likert scale. Because of the small sample size, we grouped the answers from the five initial levels into three: *Strongly disagree* and *Disagree* were grouped into a new level *No*; and *Agree* and *Strongly agree* into a new level *Yes*. *Neutral* was left as such. Table 2 displays a summary of participants' PE and MT perceptions.

[1] http://www.proz.com
[2] http://www.casmacat.eu/

| | No | Neutral | Yes |
|---|---|---|---|
| I am comfortable post-editing to human-like (perfect) quality | 3 | 1 | 5 |
| I am comfortable post-editing to less-than-perfect quality | 4 | 1 | 4 |
| I prefer PE to translating from scratch (without a TM) | 3 | 2 | 4 |
| MT helps me maintain translation consistency | 4 | 1 | 4 |
| MT helps me translate faster | 2 | 4 | 3 |
| PE is more laborious than translating from scratch or with a TM | 3 | 4 | 2 |
| I prefer PE to processing 85-94% TM matches | 1 | 5 | 3 |
| I prefer PE to editing a human translation | 5 | 3 | 1 |

Table 2: Translators' perceptions of PE and MT

Answers show therefore a mix of opinions towards PE. Overall, translators are comfortable post-editing but do not prefer it to editing a human translation.

## 4 Variables of interest

### 4.1 Machine Translation Quality

Initiatives providing free resources for MT development and evaluation help boost MT research collaboration and efforts. One of these efforts is the WMT evaluation campaign of the Association for Computational Linguistics (ACL). Among the data they freely provide are training and test data sets. As part of their 8th Workshop on Statistical Machine Translation (WMT 2013)[3], the organizers released a test set comprised of 3,000 source English sentences and their corresponding Spanish human reference translations.

We trained nine MT systems with training data from the European Parliament proceedings, News Commentary, Common Crawl, and United Nations. The systems are phrase-based Moses systems (Koehn et al., 2007) with hierarchical lexicalized reordering (Galley and Manning, 2008), operation sequence model (Durrani et al., 2013), and sparse lexical features, using all available language model data (target side of the full parallel corpus, plus the provided monolingual news corpus and LDC Gigaword). The best system reaches comparable quality to the best system participating in the WMT 2013 evaluation campaign.

We iteratively halved the parallel training corpus to obtain systems of inferior quality. The quality of the MT systems was measured with case-sensitive BLEU (Papineni et al., 2002) on the official WMT 2013 test set. Table 3 summarizes MT systems' quality and training corpus size.

| System | BLEU | Training sentences | Training words (English) |
|---|---|---|---|
| MT1 | 30.37 | 14,700k | 385M |
| MT2 | 30.08 | 7,350k | 192M |
| MT3 | 29.60 | 3,675k | 96M |
| MT4 | 29.16 | 1,837k | 48M |
| MT5 | 28.61 | 918k | 24M |
| MT6 | 27.89 | 459k | 12M |
| MT7 | 26.93 | 230k | 6.0M |
| MT8 | 26.14 | 115k | 3.0M |
| MT9 | 24.85 | 57k | 1.5M |

Table 3: MT Systems' quality and training corpus size

[3] http://www.statmt.org/wmt13/

### 4.2 Human-mediated Translation Edit Rate

Human-mediated Translation Edit Rate (HTER; Snover et al., 2006) measures the minimum number of operations (insertions, deletions, substitutions and shifts) needed to convert a machine translation into its post-edited version. It is computed by dividing the number of operations by the number of words in the post-edited version. HTER can be used as a measure of MT quality: the fewer the changes that need to be applied to the machine translation, the more similar it is to the post-edited, reference translation and therefore the higher the MT quality. Likewise, HTER can also be used as a measure of technical PE effort: the fewer changes necessary to convert the machine translation into its post-edited version, the less the effort exerted by the translator.

In this study, HTER scores were computed for all nine systems, based on the machine translated texts and their non-minimally post-edited versions, with the freely available tercom software[4].

### 4.3 Actual Edit Rate

HTER is concerned about the PE product, not the process. It therefore does not measure translators' actual edit operations, which may involve going back and applying corrections to previously post-edited parts of the text. We are interested in examining how actual edit operations vary across systems. Edit operations (i.e., insertions and deletions) are measured as the keystroke and/or mouse combinations leading to the insertion or deletion event, which do not necessarily correspond to the number of characters inserted or deleted. For instance, if a translator deletes a word by selecting it with the mouse and pressing *backspace*, it counts as one deletion event. If they move a word by cutting it and pasting it somewhere else, it constitutes one deletion and one insertion event. Insertion and deletion events were normalized by the number of words in the machine translated text to obtain a what we call here the *Actual Edit Rate*, henceforth AER.

### 4.4 Post-editing Time

Mean PE time per word per system was calculated by dividing the time spent post-editing each MT system's output by the number of source words translated by each system. Table 4 shows time measurements as PE time in seconds per word (spw) and as PE speed in words per hour (wph).

| System | Mean PE time (spw) | Mean PE speed (wph) |
|---|---|---|
| MT1 | 4.06 | 887 |
| MT2 | 4.38 | 822 |
| MT3 | 4.23 | 851 |
| MT4 | 4.54 | 793 |
| MT5 | 4.35 | 828 |
| MT6 | 4.36 | 826 |
| MT7 | 4.66 | 773 |
| MT8 | 4.94 | 729 |
| MT9 | 5.03 | 716 |

Table 4: Systems' mean PE time (seconds per word) and PE speed (words per hour)

[4] https://github.com/jhclark/tercom

### 4.5 Post-editing Quality

Error-based quality assessment frameworks allow for the quantification of translation quality according to the type and severity of errors present in the translation. Previous PE studies (Guerberof, 2009; Temizöz, 2013) have used LISA's Quality Model 3.1, one such framework traditionally used in localization settings, to evaluate the quality of the post-edited texts. This model, however, is not officially available anymore. We therefore assessed PE quality with another error-based model, a quality metric compliant with QTLaunchPad's Multidimensional Quality Metric (MQM) framework.

We used the decision trees and guidelines provided in Burchardt and Lommel (2014) as a reference during the issue annotation process. Quality was assessed along the dimensions of Accuracy (Mistranslations, Omissions, Untranslated and Additions) and Fluency (Grammar, Style and Typography). We excluded Spelling from our quality metric as some translators pointed out that the spell checker did not work. Following LISA's standard weight scale, minor errors were given a weight of 1 and major errors were given a weight of 5.

## 5 Analysis and Results

Sentences with a logged period of inactivity of 2 minutes or more were excluded from analysis as such long pauses are likely not indicative of difficulties posed by the underlying MT system. Also excluded were sentences inadvertently skipped by translators and sentences with unreliable measurements[5]. In total, out of 1233 observations, 1202 were considered valid and submitted for analysis.

### 5.1 Post-Editing Time by Machine Translation System

For each MT system, we plotted mean PE time against BLEU score:

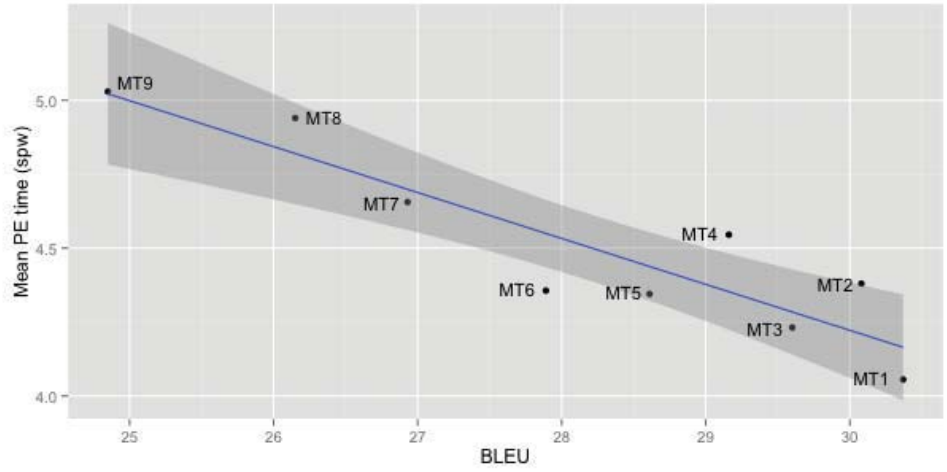



Figure 1: Scatter plot of systems' mean PE time against systems' BLEU and regression line with 95% confidence bounds

[5]CASMACAT logs editing time on a segment basis as the interval between the opening and closing of each segment. When translators access the PE interface, the first segment in the document is opened automatically. Translators do not usually start post-editing right away, instead they scroll through the document. Once acquainted with its contents, they go up the first segment, post-edit it and, when finished, close it. Most editing times logged for first segments in our study are in fact spent browsing through the whole document and are therefore unreliable.

A linear regression was applied to predict PE time based on MT quality. Our model was significant ($F(1,7) = 33.62$), with an $R^2$ of .828. It describes the effect of system's BLEU on mean PE time as following a decreasing linear relationship. Specifically, for every 1-point increase in BLEU, there is a decrease in PE time of approximately 0.16 seconds per word.

| | Estimate | Std. error | *t* value | *p* value | 95% confidence interval |
|---|---|---|---|---|---|
| intercept | 8.88 | 0.76 | 11.74 | <.001 | [7.37, 10.39] |
| slope | -0.16 | 0.03 | -5.80 | <.001 | [-0.21, -0.10] |

Table 5: Linear regression results

Assumption checks confirm the validity of the results: the plot of residual versus fitted values shows some noise but no distinctive pattern, and although residuals show a slight departure from normality, this is expected in small samples.

### 5.2 HTER and AER by Machine Translation System

We investigate how both HTER and AER vary between MT systems. Table 6 displays HTER scores and mean AER by MT system:

| System | HTER | AER |
|---|---|---|
| MT1 | 40.75 | 3.36 |
| MT2 | 40.85 | 3.05 |
| MT3 | 42.41 | 3.03 |
| MT4 | 41.57 | 3.58 |
| MT5 | 42.29 | 3.61 |
| MT6 | 43.57 | 3.66 |
| MT7 | 44.79 | 3.33 |
| MT8 | 46.15 | 3.57 |
| MT9 | 50.30 | 4.20 |

Table 6: Systems' HTER (%) and AER (events per word)

As expected, we see an almost continuous gradual increase in HTER as the quality of the MT system decreases. In contrast, our data does not allow us to establish any significant association between AER and MT quality. Keyboard activity may just not be as sensitive to MT quality as PE time. Nevertheless, MT9, the system with the lowest BLEU score has both the highest HTER and AER of all systems, representing an increase of 23.43% and 22% respectively over MT1, the best system.

### 5.3 Post-Editing Quality by Machine Translation System

Using an error-based framework to assess text quality usually involves determining an arbitrary pass/fail threshold for textual units. There is not a theoretical body of literature concerning these frameworks so we set the minimum sentence quality acceptance level at a percentage commonly referred to in industry documents, i.e., 95%[6].

PE quality is measured for the whole post-edited output, via both the MQM score and the count of sentences falling in the *Fail* and *Pass* categories, and reported by MT system. Also reported are normalized issue counts classified according to their severity (note that no critical errors were found in any of the texts).

[6] An error-free translation scores 100%. To calculate a sentence's MQM score with standard LISA severity weights the following formula applies: MQM Score (%) = 100 - (($Issues_{Minor}$ + 5 * $Issues_{Major}$ + 10 * $Issues_{Critical}$)/Sentence length)*100. A 28-word sentence with 2 minor issues and 1 major issue would have therefore a score of 100-(2+5)/28*100= 75%.

| System | MQM Score | Fail | Pass | Minor issues/k | Major issues/k |
|---|---|---|---|---|---|
| MT1 | 97.86 | 15 | 117 | 12.39 | 1.72 |
| MT2 | 97.19 | 20 | 113 | 12.83 | 3.12 |
| MT3 | 98.01 | 16 | 117 | 12.59 | 1.75 |
| MT4 | 96.76 | 26 | 109 | 15.32 | 3.40 |
| MT5 | 97.84 | 18 | 116 | 11.93 | 2.04 |
| MT6 | 98.63 | 10 | 124 | 9.88 | 1.02 |
| MT7 | 97.31 | 17 | 115 | 11.39 | 3.45 |
| MT8 | 97.47 | 22 | 112 | 10.48 | 3.38 |
| MT9 | 95.81 | 32 | 103 | 14.28 | 4.76 |

Table 7: Indicators of translation quality of post-edited translations by MT system

As expected, given that participants are professional translators post-editing to high quality, sentence-level MQM scores follow a left-skewed distribution (867 of the 1202 sentences score 100%). A Pearson's chi-square test found differences in the Fail/Pass proportions between MT systems ($x^2(8)= 19.40$, $p < .05$), with a post-hoc pairwise comparison with Holm's adjustment finding significant the differences between the MT6-MT9 pair. While the Fail/Pass ratio or the MQM scores are not significantly different for all the other pairs, MT9, the system with the lowest BLEU score, produced the output that ended up with the lowest MQM score and the highest Fail/Pass ratio: 1 in 4 sentences were post-edited to below the acceptable 95% MQM score.

In terms of error categories, minor issues are more or less equally divided into Fluency and Adequacy issues across systems, while major issues are practically all Adequacy issues, mostly Mistranslations.

### 5.4 Post-Editing Quality vs. Post-Editing Time by Machine Translation System

We plotted the Fail/Pass ratio of the post-edited output against the mean PE time for each MT system:

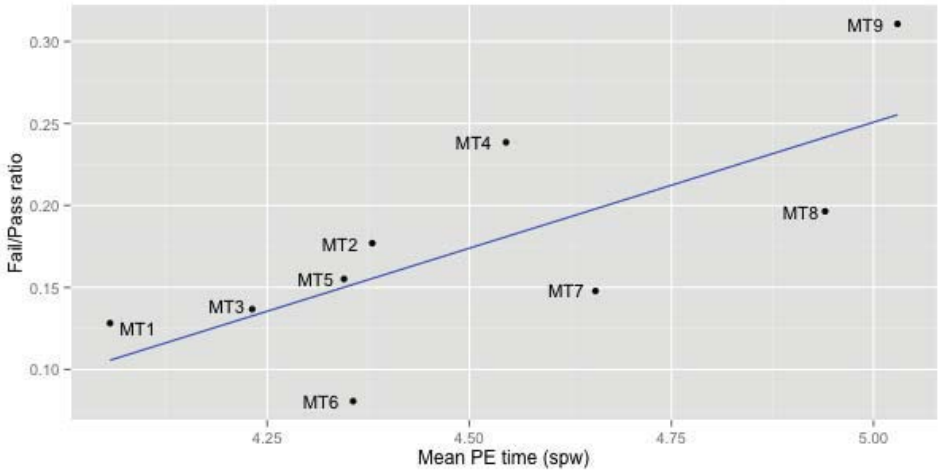



Figure 2: Scatter plot of systems Fail/Pass ratios against mean PE times with regression line

Our model describes a positive linear relationship between PE time and Fail/Pass ratio: the more time is spent post-editing, the higher the Fail/Pass ratio of the post-edited output. The

model is significant ($F(1,7) = 8.06$), with an $R^2$ of .535. Such a low $R^2$ points at additional factors affecting the quality of the post-edited texts. This will be investigated in further studies.

### 5.5 Post-Editing Time and Quality by Translators

Table 8 shows that differences in HTER, AER and PE time between translators are more pronounced than between MT systems.

| | HTER | AER | Mean PE time (spw) | MQM Score | Fail | Pass |
|---|---|---|---|---|---|---|
| TR1 | 44.79 | 2.29 | 4.57 | 98.65 | 10 | 124 |
| TR2 | 42.76 | 3.33 | 4.14 | 97.13 | 23 | 102 |
| TR3 | 34.18 | 2.05 | 3.25 | 96.50 | 26 | 106 |
| TR4 | 49.90 | 3.52 | 2.98 | 98.10 | 17 | 120 |
| TR5 | 54.28 | 4.72 | 4.68 | 97.45 | 17 | 119 |
| TR6 | 37.14 | 2.78 | 2.86 | 97.43 | 24 | 113 |
| TR7 | 39.18 | 2.23 | 6.36 | 97.92 | 18 | 112 |
| TR8 | 50.77 | 7.63 | 6.29 | 97.20 | 19 | 117 |
| TR9 | 39.21 | 2.81 | 5.45 | 96.48 | 22 | 113 |

Table 8: Indicators of translation quality of post-edited translations by translator

The inter-subject variability reported in Table 8 mirrors the findings of previous empirical PE studies such as those mentioned in Section 1.

The slowest translator, TR7 has nevertheless both low HTER and AER. TR7's log shows they left the CASMACAT interface (by accessing another tab in the browser) and re-accessed CASMACAT an average of over 100 times per text. Without a screen recorder, we do not know what the translator was doing outside CASMACAT, but it is likely that they were engaged in translation-related web searches, possibly because the texts posed comparatively more difficulties for them.

TR8, the second slowest post-editor, has the second highest HTER and the highest AER. Comparing both variables among translators, we see that TR8 has a HTER comparable to that of TR4, yet TR8's AER is more than double that of TR4. This indicates that TR8 did, on average, considerable overwriting before settling on a final post-edited version, likely slowing them down in the process.

In our set, the two fastest translators, TR6 and TR4, left the CASMACAT interface the fewer number of times of all (5 and 3 times per text, respectively, on average), an indication that they did not need to consult many online translation resources. TR6 and TR4 are not only the fastest but also the most experienced translators of all participants, considering experience both in terms of length (>10 years for both) and translation volume in the preceding 12 months (40,000-55,000 and 25,000-39,900 words, respectively).

The two translators with industry PE certifications, TR6 and TR3, were the first and third fastest post-editors. They produced the texts with the two lowest HTER scores. While differences in the quality of the post-edited output are not statistically significant for translators, TR6 and TR3 produced two of the three translations with the highest Fail/Pass ratios. The second highest Fail/Pass ratio was produced by TR2, the less experienced translator, both in length of experience (2 to 5 years) and translation volume in the 12 preceding months (<10,000 words).

Lastly, we investigated the relationship between translators' PE time and PE quality by plotting Fail/Pass ratio against mean PE times. We did not find any association between translators' PE time and PE quality, as evidenced by the lack of pattern in the scatter plot in Figure 3:

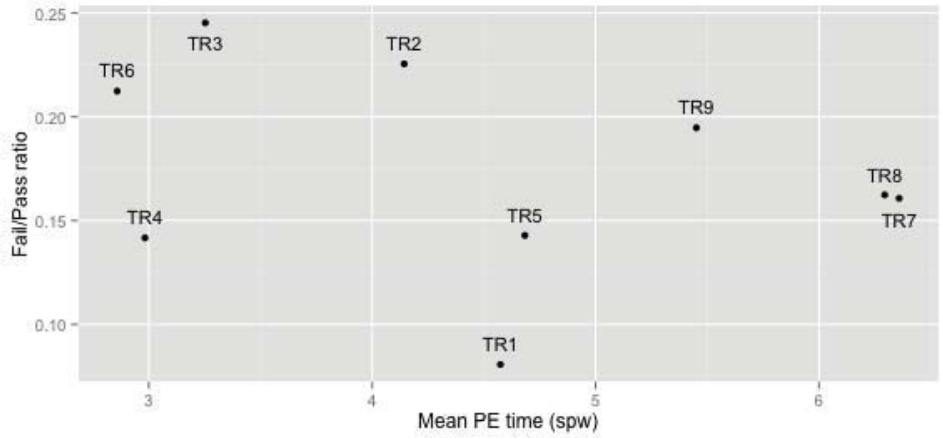



Figure 3: Scatter plot of translators Fail/Pass ratios against mean PE times with regression line

## 6 Conclusions

We presented a study that measured the impact of machine translation quality on post-editor speed and final translation quality. We found a linear relationship between machine translation quality, as measured by the BLEU score of the system, and post-editing speed of about 0.16 seconds/word post-editing time decrease per BLEU point increase. This is about a 3–4% speed increase for each BLEU point. We also found that worse machine translation output ultimately led to worse translation quality after post-editing. As future lines of research, we suggest investigating whether these findings extend to other language pairs.

## Acknowledgements

This research was supported by a grant from The University of Auckland's Faculty of Arts Doctoral Research Fund.

## References

Alabau, V., Bonk, R., Buck, C., Carl, M., Casacuberta, F., García-Martínez, M., González, J., Koehn, P., Leiva, L., Mesa-Lao, B., et al. (2013). CASMACAT: An open source workbench for advanced computer aided translation. *The Prague Bulletin of Mathematical Linguistics*, 100:101–112.

Burchardt, A. and Lommel, A. (2014). Practical guidelines for the use of MQM in scientific research on translation quality. `http://www.qt21.eu/downloads/MQM-usage-guidelines.pdf`. Accessed 07/12/2016.

Carl, M., Dragsted, B., Elming, J., Hardt, D., and Jakkobsen, A. L. (2011). The process of post-editing: a pilot study. In *8th International Conference NLPSC workshop. Special issue: Human-machine interaction in translation. Special theme: Human-machine interaction in translation*, pages 131–142.

Cettolo, M., Niehus, J., Stlker, S., Bentivogli, L., and Federico, M. (2013). Report on the 10th IWSLT evaluation campaign. In *10th Workshop on Spoken Language Translation (IWSLT)*.

De Sutter, N. (2012). MT evaluation based on post-editing: a proposal. In Depraetere, I., editor, *Text, Translation, Computational Processing [TTCP] : Perspectives on Translation Quality*, pages 125–144. De Gruyter Mouton, Berlin and Boston.

Durrani, N., Fraser, A., Schmid, H., Hoang, H., and Koehn, P. (2013). Can Markov models over minimal translation units help phrase-based SMT? In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, Sofia, Bulgaria. Association for Computational Linguistics.

Fiederer, R. and O'Brien, S. (2009). Quality and machine translation: A realistic objective? *The journal of Specialised translation*, 11:52–72.

Flournoy, R. and Duran, C. (2009). Machine translation and document localization at Adobe: From pilot to production. In *Machine Translation Summit XII*.

Galley, M. and Manning, C. D. (2008). A simple and effective hierarchical phrase reordering model. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 848–856, Honolulu, Hawaii.

García, I. (2010). Is machine translation ready yet? *Target*, 22(2):7–21.

Green, S., Heer, J., and Manning, C. D. (2013). The efficacy of human post-editing for language translation. In *2013 IGCHI Conference on Human Factors in Computing Systems*, pages 439–448.

Groves, D. and Schmidtke, D. (2009). Identification and analysis of post-editing patterns for MT. In *Machine Translation Summit XII*, pages 429–436.

Guerberof, A. (2009). Productivity and quality in the post-editing of outputs from translation memories and machine translation. *Localisation Focus. The International Journal of Localisation*, 7(1):11–21.

Koehn, P. and Germann, U. (2014). The impact of machine translation quality on human post-editing. In *EACL 2014 Workshop on Humans and Computer assisted Translation*.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., and Herbst, E. (2007). Moses: open source toolkit for statistical machine translation. In *ACL*. Association for Computational Linguistics.

Krings, H. P. (2001). *Repairing Texts: Empirical Investigations of Machine Translation Post-Editing Processes*. Kent, Ohio, Kent State University Press.

Läubli, S., Fishel, M., Massey, G., Ehrensberger-Dow, M., and Volk, M. (2013). Assessing post-editing efficiency in a realistic translation environment. In *Workshop on Post-editing Technology and Practice*, pages 83–91.

O'Brien, S. (2011). Towards predicting post-editing productivity. *Machine Translation*, 25(3):197–215.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. ACL.

Plitt, M. and Masselot, F. (2010). A productivity test of statistical machine translation post-editing in a typical localisation context. *The Prague Bulletin of Mathematical Linguistics*, 93:7–16.

Skadiņš, R., Puriņš, M., Skadiņa, I., and Vasiļjevs, A. (2011). Evaluation of SMT in localization to under-resourced inflected language. In *15th International Conference of the European Association for Machine Translation (EAMT)*, pages 35–40.

Snover, M., Dorr, B., Schwartz, R., Makhoul, J., Micciulla, L., and Weischedel, R. (2006). A study of translation error rate with targeted human annotation. In *7th Biennial Conference of the Association for Machine Translation in the Americas (AMTA 2006)*, pages 223–231.

Tatsumi, M. (2009). Correlation between automatic evaluation metric scores, post-editing speed, and some other factors. In *Machine Translation Summit XII*.

Temizöz, Ö. (2013). *Postediting Machine Translation Output and its Revision: Subject-matter experts versus Professional Translators*. PhD thesis, Universitat Rovira i Virgili.

# Fuzzy-match repair using black-box machine translation systems: what can be expected?

**John E. Ortega** jeo10@dlsi.ua.es
**Felipe Sánchez-Martínez** fsanchez@dlsi.ua.es
**Mikel L. Forcada** mlf@dlsi.ua.es
Dept. de Llenguatges i Sistemes Informatics, Universitat d'Alacant, E-03071, Alacant, Spain

**Abstract**

Computer-aided translation (CAT) tools often use a translation memory (TM) as the key resource to assist translators. A TM contains translation units (TU) which are made up of source and target language segments; translators use the target segments in the TU suggested by the CAT tool by converting them into the desired translation. Proposals from TMs could be made more useful by using techniques such as fuzzy-match repair (FMR) which modify words in the target segment corresponding to mismatches identified in the source segment. Modifications in the target segment are done by translating the mismatched source sub-segments using an external source of bilingual information (SBI) and applying the translations to the corresponding positions in the target segment. Several combinations of translated sub-segments can be applied to the target segment which can produce multiple repair candidates. We provide a formal algorithmic description of a method that is capable of using any SBI to generate all possible fuzzy-match repairs and perform an oracle evaluation on three different language pairs to ascertain the potential of the method to improve translation productivity. Using DGT-TM translation memories and the machine system Apertium as the single source to build repair operators in three different language pairs, we show that the best repaired fuzzy matches are consistently closer to reference translations than either machine-translated segments or unrepaired fuzzy matches.

## 1 Introduction

Computer-aided translation (CAT) tools often use a translation memory (TM), containing translation units (TU), as the key resource to assist translators. TU are made up of source and target language segments and translators use the target segments in the TU suggested by the CAT tool by converting them into the desired translation. When an exact match (100%, $s = s'$) is not available one can use a *fuzzy-match repair* method to *repair* a translation proposal $t$. The aim of these methods is to replace the sub-segments in $t$ that are the translation of the sub-segments in $s$ that do not appear in $s'$ by the translation of the corresponding sub-segment in $s'$. Fuzzy-match repair is gaining more traction in modern tools such as DejaVu[1] and MemoQ[2] as a reliable method of replacing words in a proposed target-language (TL) segment $t$ by using a source of bilingual information (SBI) such as the very same TM being used, a dictionary, or an on-line translation tool.

[1] http://www.atril.com/content/10-deepminer-fuzzy-matches-repair-458
[2] https://www.memoq.com/whats-new-in-memoq-2015

Ortega et al. (2014) describe a method that is capable of using any SBI for fuzzy-match repair. This method first aligns the words in the SL segment $s$ of the TU being repaired $(s, t)$ with the words in the segment to be translated $s'$ and identifies the mismatched words in $s$ and $s'$, i.e. the sub-segments they do not have in common. It then uses the SBIs available to identify the sub-segments in $t$ that are the translations of the mismatched sub-segments in $s$ in a way similar to that used by Esplà-Gomis et al. (2011), and then to build a set of *patching operators* by translating the mismatched sub-segments in $s'$. Each patching operator specifies the TL sub-segment $\tau$ in $t$ that needs to be repaired and the TL sub-segment $\tau'$ to be used for repairing. Combinations of patching operators can then be applied to obtain a set of candidate repaired TL segments from which the one to be finally used can be selected.

In this paper, we revisit Ortega et al. (2014)'s approach to fuzzy-match repair and go one step further; in particular in this paper we:

- provide an algorithmic description of the method;
- introduce a set of *principled* restrictions by establishing a set of compatibility rules between patching operators so that two patching operators are not applied on the same mismatch;
- extensively evaluate the method at the document level on DGT-TM[3] texts in three different language pairs, namely English–Spanish, Spanish–French and Spanish–Portuguese;
- provide some insight on the results by studying how often patching operators can actually be built using the SBI available.

The rest of the paper is organised as follows. The next section discusses related work on fuzzy-match repair and stresses the main differences with respect to the approach described in this paper. Section 3 then provides an algorithmic description of our fuzzy-match repair method, whereas Section 4 describes the rationale behind the principled restrictions that prevent two patching operators from working on the same mismatch. Sections 5 and 6 discuss the experimental settings and the results of an oracle evaluation we have conducted to determine the potential of the method. The paper ends with some remarks and a description of future research lines.

## 2 Related Work

In the literature, one can find many papers addressing the combination of machine translation and translation memories, most of which explore different ways of integrating sub-segments from the translation memory into the decoding process of a phrase-based statistical machine translation system (Biçici and Dymetman, 2008; Simard and Isabelle, 2009; Zhechev and Genabith, 2010; Koehn and Senellart, 2010; Li et al., 2016). Alternative approaches, such as those by Dandapat et al. (2011), Hewavitharana et al. (2005) and Kranias and Samiotou (2004), use instead the target segment $t$ in a translation unit $(s, t)$ as the *backbone* or the *basis* of the translation to be produced and describe ways to *repair* it by modifying those sub-segments in $t$ that are the translation of the mismatched sub-segments in $s$. The method proposed here, which extends that of Ortega et al. (2014), belongs to this second group.

Dandapat et al. (2011)'s method first aligns, in a way similar to ours, the words in $s$ and $s'$ using the (word-based) edit distance (Levenshtein, 1966) and marks the mismatched sub-segments in $s$ and $s'$ for translation. It then aligns the mismatch sub-segments in $s$ with their

[3]The translation memory of the Directorate General for Translation of the European Comission, `https://ec.europa.eu/jrc/en/language-technologies/dgt-translation-memory`

counterparts in $t$ by using a sub-segmental translation memory built on the user's translation memory following the standard method to obtain phrase tables in statistical MT (Koehn, 2010). Finally the sub-segments in $t$ aligned to mismatched sub-segments in $s$ are replaced by the translations of the corresponding sub-segments in $s'$ as they are found in the sub-segmental translation memory. The main differences with the approach described here are (a) that Dandapat et al. (2011) do not take into account the context words around the mismatches —which may lead to incorrect translations due to *boundary friction* problems such as incorrect agreement or incomplete word reorderings— and (b) that they rely on the user's translation memory (which may be small) rather than on an external SBI.

Hewavitharana et al. (2005) first use a modified IBM model 1 to align the mismatched words in $s$ to sequences of one or more words ("phrases") in $t$ and then directly map the sequence of source-side one-word edit operations (substitutions, deletions and insertions) needed to convert $s$ into $s'$, the segment to be translated, into an identical sequence of edit operations on the corresponding word sequences in $t$ to generate the fuzzy-match repaired translation. An important strength of their method is that multiple alternative target-side edits are possible for each source-side insertion or substitution, and that they score them using a probabilistic model. An important limitation of their method (as compared with ours) is the lack of context around source-side one-word edits.

Kranias and Samiotou (2004) use several linguistic resources —such as bilingual dictionaries and lists of suffixes and closed-class words— to align the words in $s$ to those in $t$ and then uses these alignments to identify the words in $t$ to be repaired. Finally, the words to be repaired are replaced (edited, inserted or deleted) by the translation of the corresponding mismatch in $s'$ obtained using machine translation. This method is similar to the one we describe in this paper but differs in that it only uses context around the mismatches when the new segment $s'$ contains words not found in $s$. In contrast, we always use context when available around all mismatches, which allows us to treat insertions, deletions and substitutions in the same way, and to mitigate the incomplete reordering and agreement errors that may occur because of not using context.

Finally, it is worth noting that commercial computer-aided translation software have recently begun to implement fuzzy-match repair. For example, MemoQ[4] implements a feature called MatchPatch that uses term bases and other resources for fuzzy-match repair, while Déjà Vu implements a feature called DeepMiner[5] that extracts sub-segments from the very same translation memory being used for their use for fuzzy-match repair. Unfortunately, details about how these methods work are not available.

## 3 Algorithm for Fuzzy-Match Repair

We describe a fuzzy-match repair algorithm that generates a set of candidate fuzzy-match- repaired segments from a translation unit $(s, t)$ and the SL segment to be translated $s'$ by using any SBI. First, we describe the algorithm used to build the list of patching operators to be used for fuzzy-match repair; then, we describe the algorithm that explores all possible combinations of patching operators to generate the set of candidate fuzzy-match repaired segments.

In order to build the list of patching operators (see Algorithm 1), first the alignment between the words in the SL segment to be translated $s'$ and those in the SL segment $s$ in the TU being repaired is obtained by a method based on the (word-level) edit-distance algorithm.[6] The string-positioned sub-segment pairs $(\sigma, \sigma')$, containing unaligned (unmatched) words and their corresponding positions in $s$ and $s'$, are then obtained by using the phrase-pair extraction algorithm used in phrase-based statistical machine translation (Koehn, 2010, section 5.2.3) to

[4] `https://www.memoq.com/whats-new-in-memoq-2015`

[5] `http://www.atril.com/software/dj-vu-x3-professional`

[6] If more than one optimal path is available to align $s'$ and $s$, on of them is chosen arbitrarily.

**Algorithm 1 BuildPatchOp**$(s', (s, t))$ generates the set of patching operators to use.

**Input:** SL segment to be translated $s'$; TU $(s, t)$ to be repaired
**Output:** A list of patching operators $P$

```
P ← ()  ▷ Initially P is an empty list
A ← EditDistanceAligner(s', s)  ▷ Get the word alignment between s and s'
for (σ, σ') ∈ ExtractPhrasePairs(s', s, A) do
    M ← Translate(σ)  ▷ M is a set with translations of σ
    M' ← Translate(σ')  ▷ M' is a set with translations of σ'
    for μ ∈ M do
        for μ' ∈ M' do
            for τ ∈ FindInSegment(μ, t) do
                τ' ← AttachTranslationToString(τ, μ')
                append (σ, σ', τ, τ') to P
            end for
        end for
    end for
end for
return P
```

obtain bilingual phrase pairs. After this, for each sub-segment pair $(\sigma, \sigma')$ the pair of sets of translations into the TL $(M, M')$ is obtained by using the SBI available. Finally, the translations in those sets are used to build patching operators by looking for all the occurrences in $t$ of the target sub-segments $\mu \in M$ to get the corresponding string-positioned target sub-segments $\tau$, and then attaching to each $\tau$ the target sub-segment $\mu'$ to get $\tau'$.

The following example illustrates how the set of patching operators is built. Suppose the segment $s'$ = *Bill found out about the fraud* to be translated into Spanish with the help of the TU $(s, t)$ = (*Gina found out about the news*, *Gina se enteró de las noticias*). The unmatched (unaligned) words in $s'$ are *Bill* and *fraud*, whereas the unmatched (unaligned) words in $s$ are *Gina*, and *news*. After word alignment these are the sub-segments pairs $(\sigma, \sigma')$ (up to length 3) which contain at least an unmatched word together with their translations $(\mu, \mu')$ into Spanish:[7]

| $\sigma$ | $\sigma'$ | $\mu$ | $\mu'$ | $\mu$ in $t$? |
|---|---|---|---|---|
| *Gina found* | *found* | *Gina encontró* | *encontró* | no |
| *Gina found* | *Bill found* | *Gina encontró* | *Bill encontró* | no |
| *Gina found out* | *found out* | *Gina se enteró* | *se enteró* | yes |
| *Gina found out* | *Bill found out* | *Gina se enteró* | *Bill se enteró* | yes |
| *found* | *Bill found* | *encontró* | *Bill encontró* | no |
| *found out* | *Bill found out* | *se enteró* | *Bill se enteró* | yes |
| *about the* | *about the fraud* | *sobre el* | *de la estafa* | no |
| *about the news* | *about the* | *de noticias* | *sobre el* | yes |
| *about the news* | *about the fraud* | *de las noticias* | *de la estafa* | yes |
| *the* | *the fraud* | *el* | *la estafa* | no |
| *the news* | *the* | *las noticias* | *el* | yes |
| *the news* | *the fraud* | *las noticias* | *la estafa* | yes |

Only in those cases in which $\mu$, the translation of $\sigma$, is found in the target segment $t$ of the TU being repaired a patching operator can be built; this is indicated by the fifth column in the table

[7] Note that the string-positioned sub-segment pairs $(\sigma, \sigma')$ extracted from $s$ and $s'$ always contain an aligned word in $s$ or $s'$. In this example we are assuming that the sets of translations $M$ and $M'$ of $\sigma$ and $\sigma'$ are singletons.

**Algorithm 2 Patching**$(P, O, n, (s,t), t^{\simeq}, D, T)$ generates all possible fuzzy-match repaired segments by backtracking.

**Input:** List of patching operators $P$; set of patching operators $O$ applied so far; position in $P$ of the patching operator being considered, $n$; TU to be repaired $(s,t)$; fuzzy-match repaired segment being built $t^{\simeq}$; boolean $D$ indicating whether the $n$-th patching operator in $P$ will be attempted to apply (true) or not (false), list $T$ containing fuzzy-match-repaired segments

1: **if** $D$ **then**
2: **if** $\mathrm{Compatible}(P_n, O, (s,t))$ **then**
3: $\mathrm{ApplyPatchOp}(P_n, t^{\simeq})$
4: $O \leftarrow O \cup \{P_n\}$ ▷ Add compatible patching operator
5: **else**
6: **return** ▷ Prune this branch of the recursion tree
7: **end if**
8: **end if**
9: **if** $n = \mathrm{length}(P)$ **then**
10: **append** $t^{\simeq}$ **to** $T$ ▷ Add candidate fuzzy-match repaired segment to list $T$
11: **return** ▷ All the patching operators have been considered
12: **else**
13: $\mathrm{Patching}(P, O, n+1, (s,t), t^{\simeq}, \textbf{true}, T)$ ▷ Continue by applying operator $n+1$
14: $\mathrm{Patching}(P, O, n+1, (s,t), t^{\simeq}, \textbf{false}, T)$ ▷ Continue by not applying operator $n+1$
15: **end if**

above.

Algorithm 2 generates the set of all possible fuzzy-match repaired segments by using those sets in $\mathcal{P}(P)$ (the power set of $P$) containing compatible patching operators. This is achieved through a backtracking algorithm that performs a recursive depth-first search and incrementally builds fuzzy-match repaired segments $t^{\simeq}$; the algorithm is initialized with two calls $\mathrm{Patching}(P, \emptyset, 1, (s,t), t, \textbf{false}, ())$ and $\mathrm{Patching}(P, \emptyset, 1, (s,t), t, \textbf{true}, ())$, where $()$ stands for an empty list. At each level of the recursion tree a new patching operator is considered and tested for applicability ($D = \textbf{true}$) or discarded ($D = \textbf{false}$). For a patching operator to be applicable it needs to be compatible with the set of patching operators $O$ applied so far to build $t^{\simeq}$ (see Section 4). If it is compatible with the rest of patching operators in $O$, the patching operator is added to $O$ and applied (lines 3–4); otherwise the branch of the recursion tree is cut. When a leaf of the recursion tree is reached (i.e. $n = \mathrm{length}(P)$) the corresponding fuzzy-match repaired segment $t^{\simeq}$ is added to the list $T$ of candidate fuzzy-match repaired segments. The algorithm $\mathrm{ApplyPatchOp}(o, t^{\simeq})$ replaces in $t^{\simeq}$ the sub-segment $\tau$ by $\tau'$; this can be safely done if patching operator $P_n$ is compatible with the other patching operators applied so far.

This algorithm assumes that patching operators that are compatible can be applied in any order because the repaired segment to be generated would be the same. Thanks to this assumption, the worst-case complexity of the algorithm is $O(2^n)$, with $n = \mathrm{length}(P)$, in which case $2^n$ repaired segments are produced. If the algorithm had to explore the application of all the patching operators in $P$ and in all possible orders its worst-case complexity would be super-exponential.

For the example introduced above, Algorithm 2 would produce 128 repaired segments if all patching operators were compatible. However, most of them are not compatible because they edit the same words in $t$ (see next section) and the algorithm ends up producing only 25 repaired segments. Some of these 25 repaired segments are identical but are produced by applying a different set of patching operators. For instance, the

repaired segment *Bill se enteró de la estafa* is produced by applying the patching operator (*Gina found out*, *Bill found out*, *Gina se enteró*, *Bill se enteró*) and either the patching operator (*about the news*, *about the fraud*, *de las noticias*, *de la estafa*) or the patching operator (*the news*, *the fraud*, *las noticias*, *la estafa*).

## 4 Restrictions

Ortega et al. (2014) introduce three restrictions: two related to the type of sub-segments used to build the patching operators and a third one related to the words in $t$ being edited. The first two restrictions —one restricting the length of the sub-segments and the other one requiring a certain amount of context words around mismatches— are optional and were introduced to reduce the number of patching operators to be considered. These optional restrictions throw away *legal* repairs that are however considered to be of low quality and will not be applied for the experiments reported in Section 5.

The third restriction cannot be avoided and is needed in order to prevent two patching operators from editing the same word in $t$. However, it may happen that two patching operators working on the same mismatch do not edit any of the words in $t$ but introduce missing ones. In those cases, the fuzzy-match repair algorithm of Ortega et al. (2014) may end up producing candidate fuzzy-match repaired segments $t^{\simeq}$ with repeated words. The following example illustrates this situation. Suppose the segment $s' =$ *the size does not exceed 100 cm* to be translated with the help of the translation unit $(s, t) =$ (*the size does not exceed 100*, *el tamaño no supera los 100*) whose target segment can be repaired with the two patching operators $(\sigma_1, \sigma'_1, \tau_1, \tau'_1) =$ (*exceed 100*, *exceed 100 cm*, *supera los 100*, *supera los 100 cm*) and $(\sigma_2, \sigma'_2, \tau_2, \tau'_2) =$ (*100*, *100 cm*, *los 100*, *los 100 cm*). As both patching operators do not edit (change) any word in $t$ they could be applied one after the other and produce the fuzzy-match repaired segment $t^{\simeq} =$ *el tamaño no supera los 100 cm cm*, which contains duplicated words due to the fact that the word *cm* is to be inserted by both operators.

To avoid this problem we need to identify when two patching operators work on the same mismatch, and to do so one needs to check the mismatches both in $s$ and $s'$ because there may be words in $s$ not appearing in $s'$ (the mismatch only shows up in $s$), or words that do not appear in $s$ but are introduced in $s'$ (the mismatch only shows up in $s'$, as in the example above). Hence two patching operators $o_i = (\sigma_i, \sigma'_i, \tau_i, \tau'_i)$ and $o_j = (\sigma_j, \sigma'_j, \tau_j, \tau'_j)$ will be marked as incompatible if they edit the same word in $t$ (as in the work by Ortega et al. (2014)) or they meet the following condition:

$$(\mathrm{mismatch}(\sigma_i, s) \cap \mathrm{mismatch}(\sigma_j, s) \neq \emptyset) \vee (\mathrm{mismatch}(\sigma'_i, s') \cap \mathrm{mismatch}(\sigma'_j, s') \neq \emptyset)$$

where $\mathrm{mismatch}(x, y)$ returns the set of mismatch words covered by sub-segment $x$ in segment $y$.

It is worth nothing that this new restriction may mark as incompatible two patching operators that, even though they work on the same mismatch, do not edit the same words in $t$. In those cases it is still advisable to forbid the application of the two patching operators since it is very likely that they work on the same region in $t$ and their application interfere with one another. The following example illustrates this situation. Suppose the segment $s' =$ *the size is around 100 cm* to be translated with the help of the translation unit $(s, t) =$ (*the size is about 50 cm*, *el tamaño es de unos 50 cm*) whose target segment can be repaired with the two patching operators $o_1 = (\sigma_1, \sigma'_1, \tau_1, \tau'_1) =$ (*is about*, *is around*, *es de unos*, *está alrededor de*) and $o_2 = (\sigma_2, \sigma'_2, \tau_2, \tau'_2) =$ (*about 50*, *around 100*, *de unos 50*, *de unos 100*). Both operators share a mismatch (*about*) but do not edit the same words in $t$: $o_1$ edits the word *es* (which is replaced by *está*), introduces

| | | en–es | es–pt | es–fr |
|---|---|---|---|---|
| **TM** | # TUs | 196,294 | 150,567 | 149,479 |
| | Avg. SL segment length | 9.61 | 27.24 | 27.35 |
| **Test set** | # SL segments | 1993 | 1983 | 1983 |
| | # SL words | 40238 | 45334 | 46350 |
| | Avg. SL segment length | 20.19 | 22.67 | 21.73 |

Table 1: Data about the translation memories and test sets used in the experiments.

the word *alrededor* and removes (edits) the word *unos*; $o_2$ edits the word *50* and replaces it by *100*. The two operators can be applied at the same time if operator $o_2$ is applied first —the repaired target segment being $t^{\simeq}$ = *el tamaño está alrededor de 100 cm*— but not the other way around. Recall that the algorithm described in Section 3 assumes that patching operators can be applied independently of each other and the order in which they are applied does affect the final result.

## 5 Experimental settings

To evaluate the potential of the fuzzy-match repair algorithm described in Section 3, we have performed an oracle evaluation (see below) on three different language pairs: English–Spanish (en–es), Spanish–Portuguese (es–pt) and Spanish–French (es–fr). These language pairs have been chosen to study how the method behaves when translating between closely-related languages (e.g. Spanish–Portuguese and Spanish–French) and when the languages involved in the translation are not so closely related (English–Spanish). In addition, of the two closely-related language pairs we have used, Spanish and Portuguese are more alike than Spanish and French: Spanish and Portuguese are both pro-drop, Ibero-Romance languages —they permit null-subject sentences— whereas French is a non-pro-drop Gallo-Romance language.

As for the corpora used for the experiments, we have used three translation memories, one per language pair, extracted from the DGT-TM 2015 multilingual translation memory;[8] each translation memory contains between 145,000 and 200,000 translation units. In addition, we have also extracted three test sets from the same source. Each test set contains around 2,000 parallel segments with source segments no longer than 100 words. The experiments consist of simulating the translation of each source segment in the test sets by using the translation memories and using the corresponding target-language segment as a reference for evaluation. Table 1 provides additional information about the translation memories and test sets used.

As a source of bilingual information we have used the free/open-source machine translation platform Apertium (Forcada et al., 2011),[9] which provides a single translation for each source segment;[10] more precisely, we have used the language-pair packages `apertium-en-es`,[11] `apertium-es-pt`[12] and `apertium-fr-es`.[13] Apertium has been used both to build patching operators by translating sub-segments $\sigma$ into the target language and to translate the segments in the test set for which a fuzzy match above the given threshold has not been found. Table 2 provides the word error rate (WER) and BLEU scores attained by Apertium when translating the source-language segments in the test set; the percentage of out-of-vocabulary words (OOV) is also reported. As can be seen, the translations performed by

[8] `https://ec.europa.eu/jrc/en/language-technologies/dgt-translation-memory`

[9] `https://www.apertium.org`

[10] That is, sets $M$ and $M'$ in lines 4 and 5 of Algorithm 1 are singletons in this case.

[11] SVN revision 64348.

[12] SVN revision 62539.

[13] SVN revision 62696.

| | en–es | es–pt | es–fr |
|---|---|---|---|
| WER | 65.3% | 47.4% | 55.2% |
| BLEU | 18.6% | 36.4% | 24.7% |
| OOV | 2.6% | 2.4% | 2.4% |

Table 2: Apertium's performance on the test sets and percentage of out-of-vocabulary words (OOV).

Apertium need less post-editing in the case of the two closely-related language pairs (es–pt and es–fr) than in the case of English–Spanish.

Finally, we evaluate the potential of our fuzzy-match repair method with fuzzy-match score thresholds of 60%, 70% and 80% with the aim of studying whether out method is more capable of repairing fuzzy matches above a given threshold. In this regard it is worth noting that professional translators usually set the fuzzy-match score threshold above 60% (Bowker, 2002).

### 5.1 Oracle Evaluation

The way to study the potential of our approach for fuzzy-match repair has been to generate, for each source segment $s'$ in the test set, the set of all possible fuzzy-match repaired target segments $T$ and then use $t'$, the translation of $s'$, to choose the best one and evaluate its quality. Obviously, in a real setting $t'$ would not be available and the best fuzzy-match repaired segment would need to chosen using a method similar to those used for estimating the quality of machine translation output (Specia and Soricut, 2013; Avramidis, 2013).

What follows is a detailed explanation of the procedure we have followed with each source segment $s'$ in the test set:

1. Retrieve the set of translation units $U$ whose fuzzy-match score $\mathrm{FMS}(s', s)$ is above the desired fuzzy-match threshold $\theta$.

2. If there is no translation unit $(s, t)$ so that $\mathrm{FMS}(s', s) \geq \theta$, i.e. $U = \emptyset$, use machine translation to get a translation for $s'$. Otherwise use the TU $(s, t) \in U$ with the highest $\mathrm{FMS}(s', s)$ and produce the set $T$ with all possible target fuzzy-match repaired segments.

3. Select the fuzzy-match repaired segment $t^{\simeq *} \in T$ with the minimum edit distance to $t'$.

Once all the segments in the test set have been processed the translations produced are evaluated by comparing them to the target segments in the test set and computing the error rate over the whole test set as follows:

$$\frac{\sum_{i=0}^{N} \mathrm{ED}(t_i^*, t_i')}{\sum_{i=0}^{N} \max(|t_i^*|, |t_i'|)} \tag{1}$$

where $\mathrm{ED}(x, y)$ returns the word-based edit distance between the segments $x$ and $y$, $N$ is the number of segments in the test set, and $|x|$ is the number of words of segment $x$. This way of computing the error rate resembles the way in which the fuzzy-match score is computed.[14]

## 6 Results and Discussion

Table 3 shows, for the three different language pairs on which we have evaluated our approach and for three different fuzzy-match score thresholds (FMT) —60%, 70% and 80%—, the error rate computed as described in Equation (1) when:

[14]For instance, OmegaT (`http://www.omegat.org`) computes the fuzzy-matching score between $s$ and $s'$ as $1 - \frac{\mathrm{ED}(s,s')}{\max(|s|,|s'|)}$.

| FMT: 60% | en–es | | | es–pt | | | es–fr | | |
|---|---|---|---|---|---|---|---|---|---|
| | TM | MT | FMR | TM | MT | FMR | TM | MT | FMR |
| Error (%) | 55.0 | 65.3 | 36.5 | 56.5 | 47.4 | 31.3 | 56.4 | 55.2 | 34.7 |
| Er. (%) on matches | 20.1 | 65.3 | 17.9 | 22.5 | 47.4 | 17.0 | 20.3 | 55.2 | 16.5 |
| # matches | 1184 | 1993 | 1184 | 1221 | 1983 | 1221 | 1206 | 1983 | 1206 |
| Avg. length | 22.6 | 22.1 | 22.6 | 21.1 | 20.6 | 21.1 | 22.8 | 22.4 | 22.8 |

| FMT: 70% | en–es | | | es–pt | | | es–fr | | |
|---|---|---|---|---|---|---|---|---|---|
| | TM | MT | FMR | TM | MT | FMR | TM | MT | FMR |
| Error (%) | 61.0 | 65.3 | 38.5 | 62.4 | 47.4 | 31.8 | 62.3 | 55.2 | 35.6 |
| Er. (%) on matches | 16.3 | 65.3 | 14.6 | 18.0 | 47.4 | 13.9 | 15.8 | 55.2 | 12.8 |
| # matches | 828 | 1993 | 828 | 777 | 1983 | 777 | 786 | 1983 | 786 |
| Avg. length | 22.4 | 22.1 | 22.5 | 20.8 | 20.6 | 21.1 | 22.6 | 22.4 | 22.6 |

| FMT: 80% | en–es | | | es–pt | | | es–fr | | |
|---|---|---|---|---|---|---|---|---|---|
| | TM | MT | FMR | TM | MT | FMR | TM | MT | FMR |
| Error (%) | 69.7 | 65.3 | 42.6 | 70.1 | 47.4 | 33.8 | 69.5 | 55.2 | 38.2 |
| Er. (%) on matches | 13.1 | 65.3 | 11.9 | 15.3 | 47.4 | 11.9 | 12.2 | 55.2 | 9.7 |
| # matches | 660 | 1993 | 660 | 641 | 1983 | 641 | 649 | 1983 | 649 |
| Avg. length | 22.3 | 22.2 | 22.4 | 20.8 | 20.6 | 21.1 | 22.5 | 22.4 | 22.8 |

Table 3: For the three different language pairs considered in our evaluation and for three different means of translation —translation memory (TM), machine translation (MT) and fuzzy-match repair (FMR)— and fuzzy-match score thresholds (FMT), the table gives the error rate over the whole test set, the error rate over the segments in the test set for which a match above the given threshold is found in the translation memory, the amount of these segments (# matches) and the average length of the target segments produced.

**TM:** the target segment in the translation unit with the highest fuzzy-match score is used as a translation, if available; otherwise, an empty translation is used, and therefore the error reflects the need to type the words in the reference translation.

**MT:** the same machine translation system used as SBI (Apertium) is used to translate the source segments in the test set.

**FMR:** the translation to be evaluated is obtained by applying the fuzzy-match repair algorithm described in Section 3 with the translation unit with the highest fuzzy-match score, if available; otherwise, the translation is produced using machine translation.

Two error rates are reported, one computed on the whole test set and another computed only on the set of segments for which a TU with a fuzzy-match score above the given threshold is found (error on matches). The former provides and indication of the actual translation effort a translator would made to translate the source segments in the test set. The latter provides an indication of the performance of our method for fuzzy-match repair (FMR) without the interference of whole-segment machine translation, since it focuses only on those segments for which there is a translation unit to repair. This allows to directly compare FMR performance to that of using the target segment in the best TU without any repair (TM). In addition, the number of source segments for which a match is found in the translation memory and the average length of the translations produced are provided.

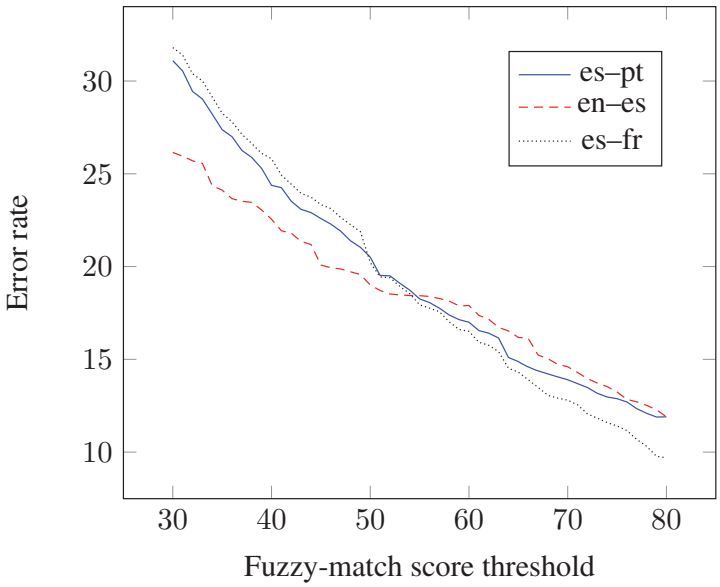
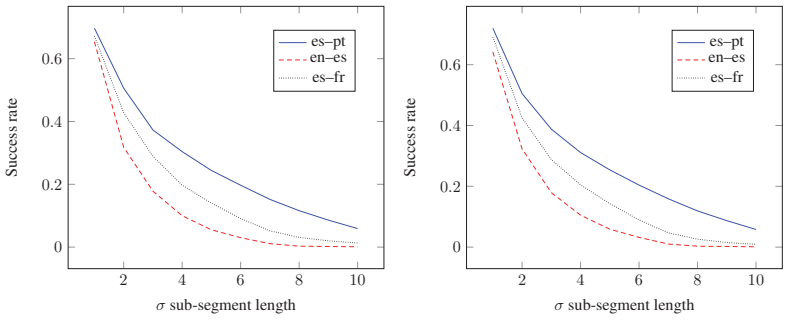



Figure 1: For the three language pairs used for evaluation, error rate over the segments in the test set for which a match above the fuzzy-match score threshold is found in the translation memory.

As can be seen, our method for fuzzy-match repair has the potential (recall that this is an oracle evaluation) to improve the translator's productivity for all three different language pairs: the error rate is both below that of using the target segment in the best translation unit (TM) and below that of using machine translation (MT). Furthermore, it is worthwhile to note that a good part of the difference in performance between the three language pairs can be attributed to the performance of the MT system; if we pay attention to the performance of FMR when the evaluation only focuses on those segments for which a match has been found we can see that the scores reported are quite similar for all language pairs, even though this does not happen in the case of the MT scores reported, i.e. our method for fuzzy-match repair appears to be quite robust to MT errors.

The error rate over the whole test set grows with the fuzzy-match score threshold (FMT). This happens because the greater this threshold is, the less source segments can be translated using fuzzy-match repair and, as a consequence, the amount of segments that are translated with Apertium grows. If we focus only on those segments that can be translated by means of FMR, we can see that the error rate decreases as the threshold grows; Figure 1 show how the error rate on matches behaves as a function of the fuzzy-match score threshold. This is the expected behaviour because as the threshold grows the amount of words to repair decreases.

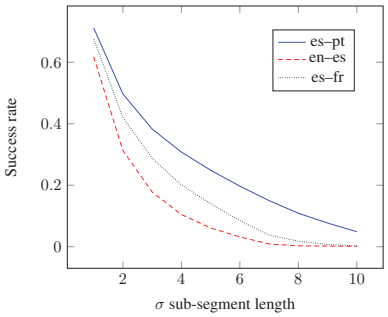
With respect to the process of building patching operators, and provided that the performance of the machine translation system differs between the language pairs, it is worth studying how successful it is our method when it comes to use Apertium to build patching operators. Figure 2 plots the success rate when building patching operators as a function of the length of the source sub-segments $\sigma$ for a fuzzy-match score threshold of 60%, 70% and 80%; as can be seen, success rates for different fuzzy-match thresholds behave very similarly. A patching operator is successful when the translation of the sub-segment $\sigma$ of $s$ is found in $t$, that is, when the machine translation system and the proposed translation unit exactly agree on the translation




(a) $\sigma$ success rate at 60% FMT




(b) $\sigma$ success rate at 70% FMT




(c) $\sigma$ success rate at 80% FMT

Figure 2: For the three different language pairs and for fuzzy-match score thresholds (FMT) of 60%, 70%, and 80% success rate when building patching operators as a function of the number of words in the source sub-segments $\sigma$ being translated.

of a source sub-segment: this acts as a safety feature, as patching is not attempted when this agreement is absent. This is why our method is robust to machine translation errors.

As can be seen, the longer the sub-segments the harder it is that the translation obtained from the SBI is found in $t$. This behaviour is present in all the language pairs and is more pronounced when the translation involves non-related language pairs (en–es) than when the languages are closely related (es–pt). The average length of $\sigma$ in the patching operators used to build the repaired target segment chosen by the oracle when the fuzzy-match score threshold is set to 80% is around 2.8 words for en–es, 3.7 for es–fr and 4.7 for es–pt.

## 7 Concluding Remarks

In this paper we have extended the approach of Ortega et al. (2014), which uses any external source of bilingual information to repair fuzzy matches coming from a translation memory, to prevent two patching operators from working on the same mismatch, and we have extensively evaluated its performance on three different language pairs and provided a more formal algorithmic description.

The oracle evaluation we have conducted reveals the potential of our approach to fuzzy-match repair. For three different language pairs we consistently improve the quality of the translations produced —both with respect to raw machine translation or unrepaired fuzzy matches— even though the SBI we have used (the machine translation system Apertium) performs below the state of the art for some language pairs. We hope that by combining different SBIs, e.g.

different machine translation systems as well as bilingual concordancers,[15] the quality of the repaired segments increase.

As a future work we plan to combine different SBI and try different methods to automatically select the best fuzzy-match repair for a given SL segment. In particular we will adapt existing techniques used for sentence-level machine translation quality estimation and devise a set of features specially designed to tackle this particular problem.

## References


Avramidis, E. (2013). Sentence-level ranking with quality estimation. *Machine Translation*, 27(3-4):239–256.

Biçici, E. and Dymetman, M. (2008). Dynamic translation memory: Using statistical machine translation to improve translation memory fuzzy matches. *Computational Linguistics and Intelligent Text Processing*, pages 454–465.

Bowker, L. (2002). *Computer-aided translation technology: a practical introduction*. University of Ottawa Press.

Dandapat, S., Morrissey, S., Way, A., and Forcada, M. L. (2011). Using example-based MT to support statistical MT when translating homogeneous data in a resource-poor setting. In *Proceedings of the 15th conference of the European Association for Machine Translation*, pages 201–208. Leuven, Belgium.

Esplà-Gomis, M., Sánchez-Martínez, F., and Forcada, M. L. (2011). Using machine translation in computer-aided translation to suggest the target-side words to change. In *Proceedings of the 13th Machine Translation Summit*, pages 172–179, Xiamen, China.

Forcada, M. L., Ginestí-Rosell, M., Nordfalk, J., O'Regan, J., Ortiz-Rojas, S., Pérez-Ortiz, J. A., Felipe Sánchez-Martínez, G. R.-S., and Tyers, F. M. (2011). Apertium: a free/open-source platform for rule-based machine translation. *Machine Translation*, 25(2):127–144.

Hewavitharana, S., Vogel, S., and Waibel, A. (2005). Augmenting a statistical translation system with a translation memory. In *Proceedings of the 10th conference of the EAMT on 'Practical applications of machine translation'*, pages 126–132, Carnegie Mellon University, Pittsburgh, USA.

Koehn, P. (2010). *Statistical Machine Translation*. Cambridge University Press, New York, NY, USA, 1st edition.

Koehn, P. and Senellart, J. (2010). Convergence of translation memory and statistical machine translation. In *Proceedings of AMTA Workshop on MT Research and the Translation Industry*, pages 21–31, Edinburgh, United Kingdom and Paris, France.

Kranias, L. and Samiotou, A. (2004). Automatic translation memory fuzzy match post-editing: a step beyond traditional TM/MT integration. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation*, pages 331–334, Lisbon, Portugal.

Levenshtein, V. (1966). Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady.*, 10(8):707–710.

Li, L., Parra Escartín, C., and Liu, Q. (2016). Combining translation memories and syntax-based smt. *Baltic Journal of Modern Computing*, 4:165–177.


[15] Such as Reverso Context, `http://context.reverso.net/translation`, Linguee, `http://www.linguee.com/`, or TransSearch, `http://tsrali.com`

Ortega, J. E., Sánchez-Martínez, F., and Forcada, M. L. (2014). Using any machine translation source for fuzzy-match repair in a computer-aided translation setting. In *Proceedings of the 11th Biennial Conference of the Association for Machine Translation in the Americas (AMTA 2014, vol. 1: MT Rsearchers)*, pages 42–53, Vancouver, BC, Canada.

Simard, M. and Isabelle, P. (2009). Phrase-based machine translation in a computer-assisted translation environment. *Proceeding of the Twelfth Machine Translation Summit (MT Summit XII)*, pages 120–127.

Specia, L. and Soricut, R. (2013). Quality estimation for machine translation: preface. *Machine Translation*, 27(3-4):167–170.

Zhechev, V. and Genabith, J. V. (2010). Seeding statistical machine translation with translation memory output through tree-based structural alignment. In *Proceedings of SSST-4 - 4th Workshop on Syntax and Structure in Statistical Translation*, pages 43–49, Dublin, Ireland.

# Fast, Scalable Phrase-Based SMT Decoding

**Hieu Hoang** hieu@moses-mt.org
Moses Machine Translation CIC, UK

**Nikolay Bogoychev** s1031254@sms.ed.ac.uk
University of Edinburgh, Scotland

**Lane Schwartz** lanes@illinois.edu
University of Illinois, USA

**Marcin Junczys-Dowmunt** junczys@amu.edu.pl
Adam Mickiewicz University

**Abstract**

The utilization of statistical machine translation (SMT) has grown enormously over the last decade, many using open-source software developed by the NLP community. As commercial use has increased, there is need for software that is optimized for commercial requirements, in particular, fast phrase-based decoding and more efficient utilization of modern multicore servers.

In this paper we re-examine the major components of phrase-based decoding and decoder implementation with particular emphasis on speed and scalability on multicore machines. The result is a drop-in replacement for the Moses decoder which is up to fifteen times faster and scales monotonically with the number of cores.

## 1 Introduction

SMT has steadily progressed from a research discipline to commercial viability during the past decade as can be seen from services such as the Google and Microsoft Translation services. As well as general purpose services such as these, there is a large number of companies that offer customized translation systems, as well as companies and organization that implement in-house solutions. Many of these customized solutions use Moses as their SMT engine.

For many users, decoding is the most time-critical part of the translation process. Making use of the multiple cores that are now ubiquitous in todays servers is a common strategy to ameliorate this issue. However, it has been noticed that the Moses decoder, amongst others, is unable to efficiently use multiple cores (Fernández et al., 2016). That is, decoding speed does not substantially increase when more cores are used, in fact, it may actually *decrease* when using more cores. There have been speculation on the causes of the inefficiency as well as potential remedies.

This paper is the first we know of that focuses on improving decoding speed on multicore servers. We take a holistic approach to solving this issue, creating a decoder that is optimized for multi-core processing speed by concentrating on four main areas:

1. Faster memory management of data-structures through the use of customized memory pools
2. Exploring alternatives to cardinality-based hypothesis stack configuration

3. Re-examining the efficiency of phrase-table lookup using translation rule caching and data compression
4. Integrating the lexicalized re-ordering model into the phrase-table, thus eliminating the need for independent random lookup this model

The result is a decoder that is significantly faster than the Moses baseline for single-threaded operation, and scales with the number of cores.

We will maintain the Moses decoder's embarrassingly parallel, one sentence-per-thread decoding framework. As far as possible, model scores and functionality are compatible with Moses to aid comparison and ease transition for existing users. The source code is available in the existing Moses repository[1].

The rest of the paper will be broken up into the following sections. The rest of this section will discuss prior work and an outline of the phrase-based model. Section 2 will then describe the modifications to improve decoding speed. We describe the experiment setup in Section 3 and present results Section 4. We conclude in the last section and discuss possible future work.

### 1.1 Prior Work

Most prior work on increasing decoding speed look to optimizing specific components of the decoder or the decoding algorithm.

Heafield (2011) and Yasuhara et al. (2013) describes fast, efficient datastructures for language models. Zens and Ney (2007) describes an implementation of a phrase-table for an SMT decoder that is loaded on demand, reducing the initial loading time and memory requirements. Junczys-Dowmunt (2012) extends this by compressing the on-disk phrase table and lexicalized re-ordering model.

Chiang (2007) describes the cube-pruning and cube-growing algorithm which allows the tradeoff between speed and translation quality to the adjusted with a single parameter. Wuebker et al. (2012b) note that language model querying is amongst the most expensive operation in decoding. They sought to improved decoding speed by caching score computations early pruning of translation options. This work is similar to Heafield et al. (2014) which group hypotheses with identical language model context and incrementally expand them, reducing LM querying.

Fernández et al. (2016) was concerned with multi-core speed but treated decoding as a black box within a parallelization framework.

There are a number of phrase-based decoding implementations, many of which implements the extensions to the phrase-based model described above. The most well known is Moses (Koehn et al., 2007) which implements a number of speed optimizations, including cube-pruning. It is widely used for MT research and commercial use.

Joshua (Li et al., 2009) also supports cube-pruning for phrase-based models. Phrasal (Spence Green and Manning, 2014) supports a number of variants of the phrase-based model. Jane (Wuebker et al., 2012a) supports the language model look-ahead described in Wuebker et al. (2012b) in addition to other tools to speed up decoding such as having a separate fast, lightweight decoder. mtplz is a specialized decoder developed to implement the incremental decoding described in Heafield et al. (2014).

The Moses, Joshua and Phrasal decoders implement multithreading, however, they all report scalability problems, either in the paper (Phrasal) or via social media (Moses[2] and Joshua[3]).

Jane and mtplz are single-threaded decoders, relying on external applications to parallelize operations.

[1] https://github.com/moses-smt/mosesdecoder/tree/master/contrib/moses2

[2] https://github.com/moses-smt/mosesdecoder/issues/39

[3] https://twitter.com/ApacheJoshua/status/342022794097340416

This paper not only focuses on faster single-threaded decoding but also on overcoming the shortcomings of existing decoding implementations on multicore servers. Unlike Fernández et al. (2016), we will optimize decoding speed by looking inside the black box. We will compare multicore performance the best-of-breed phrase-table described in Junczys-Dowmunt (2012) with our own implementation. We will use the cube-pruning algorithm, however, the standard phrase-based decoding algorithm is also available and a framework exists to accommodate other decoding algorithms in future. We use KenLM (Heafield, 2011) due to it's popularity and consistent performance, but as with Moses, other language model implementations can be added later.

### 1.2 Phrase-Based Model

The objective of decoding is to find the target translation with the maximum probability, given a source sentence. That is, for a source sentence $s$, the objective is to find a target translation $\hat{t}$ which has the highest conditional probability $p(t|s)$. Formally, this is written as:

$$\hat{t} = \arg\max_{t} p(t|s) \tag{1}$$

where the *arg max* function is the search. The log-linear model generalizes Equation 1 to include more component models and weighting each model according to the contribution of each model to the total probability.

$$p(t|s) = \frac{1}{Z} \exp(\sum_{m} \lambda_m h_m(t, s)) \tag{2}$$

where $\lambda_m$ is the weight, and $h_m$ is the *feature function*, or 'score', for model $m$. $Z$ is the partition function which can be ignored for optimization.

The standard feature functions in the phrase-based model include:

1. a distortion penalty
2. a phrase-penalty,
3. a word penalty,
4. an unknown word penalty.
5. log transforms of the target language model probability $p(t)$,
6. log transforms translation model probabilities, $p_{TM}(t|s)$ and $p_{TM}(s|t)$, and word-based translation probabilities $p_w(t|s)$ and $p_w(s|t)$,
7. log transforms of the lexicalized re-ordering probabilities,

Of these feature functions, we will focus on optimizing the speed of the phrase-table and lexicalized re-ordering models.

### 1.3 Beam Search

A translation of a source sentence is created by applying a series of translation rules which together translate each source word once, and only once. Each partial translation is known as a *hypothesis*, which is created by applying a rule to an existing hypothesis. This *hypothesis expansion* process starts with a hypothesis that has translated no source word and ends with completed hypotheses that has translated all source words. The highest-scoring completed hypothesis, according to the model score, is considered the best translation, $\hat{t}$.

In the phrase-based model, each rule translates a contiguous sequence of source words. Successive applications of translation rules do not have to be adjacent on the source side, depending on the distortion limit. The target output is constructed strictly left-to-right from the target side from the series of translation rules.

A beam search algorithm is used to create the completed hypothesis set efficiently. Hypotheses are grouped into stacks where each stack holds a number of comparable hypotheses. Most phrase-based implementations group hypotheses according to coverage cardinality.

## 2 Proposed Improvements

We will also concentrate on four main areas for optimization.

### 2.1 Efficient Memory Allocation

The search algorithm creates and destroy a large number of intermediate objects such as hypotheses and feature function states. This puts a burden on the operating system due to the need to synchronize memory access, especially when using a large number of threads. Libraries such as tcmalloc (Ghemawat and Menage, 2009) are designed to reduce locking contention for multi-threaded application but in our case, this is still not enough.

We shall seek to improve decoding speed by replacing the operating system's general purpose memory management with our own custom memory management scheme. Memory will be allocated from a memory pool rather than use the operating system's general purpose allocation functions.

A memory pool is a large block of memory that has been given to the application by the operating system. The application is then responsible for allocating portions of this memory to its components when requested. We will use thread-specific memory pools to increase speed by avoiding locking contention during memory access. Our memory pools will be dynamic. That is, the memory requirement does not have to be known or specified before running the application, the pool can grow when required but they will never reduce in size. The pools are deleted only when the application ends.

To further increase memory management speed, objects in the memory pool are not deleted. Unused data structures accumulates in the pool until a reset event. The pool is assumed to be empty and simply reused after the event. We instantiate two memory pools per decoding thread, one which is never reset and another which is reset after the decoding of each sentence. Data structures are created in either pool according to their life cycle.

Accumulating unused objects in the memory pools can result in unacceptably high memory usage so object queues are available for high-churn objects which allows the decoder to re-cycle unused objects before the reset event. This also increase speed as LIFO queues are used so that the most recently accessed memory are used, increasing CPU cache hits.

### 2.2 Stack Configurations

The most popular stack configuration for phrase-based models is coverage cardinality, that is, hypotheses that have translated the same number of source words are stored in the same stack. This is implemented in Pharaoh, Moses and Joshua.

However, there are alternatives to this configuration. Och et al. (2001) uses a single stack for all hypotheses, Brown et al. (1993) uses coverage stacks (ie. one stack per unique coverage vector) while Wuebker et al. (2012a) and Zens and Ney (2008) apply both coverage and cardinality pruning. While useful, these prior works present only one particular stack configuration each. Ortiz-Martínez et al. (2006) explore a range of stack configurations by defining a granularity parameter which controls the maximum number of stacks required to decode a sentence.

We shall re-visit the question of stack configuration with a particular emphasis on how they can help improve the tradeoff between speed and translation quality. We will do so in the context of the cube-pruning algorithm, the algorithm that we will be using and which was not available to many of the earlier work.

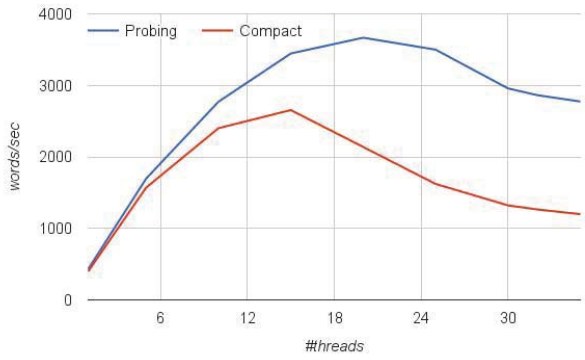



Figure 1: Moses decoding speed with two different phrase-table implementations

| | No cache | Caching |
|---|---|---|
| Decoding time | 2877 | 2540 (-12%) |

Table 1: Decoding speed (in words / sec with 32 threads) when using phrase-table cache

### 2.3 Phrase-Table Optimizations

For any phrase-table table of a realistic size, memory and loading time constraints requires us to use a load-on-demand implementation. Moses has several which we can make use of, each with differing performance characteristics. Figure 1 shows the decoding speed for the fastest two implementations. From this, it appears that the Probing phrase-table (Bogoychev and Lopez, 2016) has the fastest translation rule lookup, especially with large number of cores, therefore, we will concentrate exclusively on this implementation from hereon.

We propose two optimizations. Firstly, the translation rule caching mechanism in Moses saves the most recently used rules. However, this require locking and active management in clearing of old rules. The result is *slower* decoding, Table 1.

We shall explore a simpler caching mechanism by creating a static cache of the most likely translation rules to be used at the start of decoding.

Secondly, the Probing phrase-table use a simple compression algorithm to compress the target side of the translation rule. Compression was championed by Junczys-Dowmunt (2012) as the main reason behind the speed of their phrase-table but as we saw in Figure 1, this comes at the cost of scalability to large number of threads. We shall therefore take the opposite approach to and improve decoding speed by disabling compression.

### 2.4 Lexicalized Re-ordering Model Optimizations

Similar to the phrase-table, the lexicalized re-ordering model is trained on parallel data. A resultant model file is then queried during decoding. The need for random lookup during querying inevitably results in slower decoding speed. Previous work such as Junczys-Dowmunt (2012) improve querying speed with more compact data structures.

However, the model's query keys are the source and target phrase of each translation rule. Rather than storing the lexicalized re-ordering model separately, we shall integrating it into the translation model, eliminating the need to query a separate file. However, the model remain the same under the log-linear framework, including having its own weights.

This optimization has precedent in Wuebker et al. (2012a) but the effect on decoding speed

| | ar-en | fr-en |
|---|---|---|
| Phrase table | 17 | 5.8 |
| Language model (5-gram) | 3.1 | 1.8 |
| Lex re. model | 2.3 | 0.6 |

Table 2: Model sizes in GB

| | ar-en | fr-en |
|---|---|---|
| For speed testing | | |
| Set name | Subset of training data | |
| # sentences | 800k | 200k |
| # words | 5.8m | 5.9m |
| Avg words/sent | 7.3 | 29.7 |
| For model score testing | | |
| Set name | OpenSubtitles | newstest2011 |
| # sentences | 2000 | 3003 |
| # words | 14,620 | 86,162 |
| Avg words/sent | 7.3 | 28.7 |

Table 3: Test sets

were not published. We will compare results with using a separate model in this paper.

## 3 Experimental Setup

We trained a phrase-based system using the Moses toolkit with standard settings. The training data consisted of most of the publicly available Arabic-English data from Opus (Tiedemann, 2012) containing over 69 million parallel sentences, and tuned on a held out set. The phrase-table was then pruned, keeping only the top 100 entries per source phrase, according to $p(t|s)$. All model files were then binarized; the language models were binarized using KenLM (Heafield, 2011), the phrase table using the Probing phrase-table, lexicalized reordering model using the compact data structure (Junczys-Dowmunt, 2012). These binary formats were chosen for their best-in-class multithreaded performance. Table 2 gives details of the resultant sizes of the model files. For testing decoding speed, we used a subset of the training data, Table 3.

For verification with a different dataset, we also used a second system trained on the French-English Europarl corpus (2m parallel sentences). The two different systems have characterics that we are interested in analyzing; ar-en have short sentences with large models while fr-en have overly long sentences with smaller models. Where we need to compare model scores, we used held out test sets.

Standard Moses phrase-based configurations are used, except that we use the cube-pruning algorithm (Chiang, 2007) with a pop-limit of 400[4], rather than the basic phrase-based algorithm. The cube-pruning algorithm is often employed by users who require fast decoding as it gives them the ability to trade speed with translation quality via a simple pop-limit parameter.

As a baseline, we use a recent[5] version of the Moses decoder taken from the github repos-

[4]the pop-limit was chosen from public discussion on the Moses mailing list on an acceptable balance between decoding speed and translation quality with Moses for commercial use

[5]The experiments were performed between January and May 2016 with the latest github code to hand. The main ar-en experiments were rerun with the source code as of 8th June, 2016 to ensure there were no material difference.

| | Moses | | Our Work | |
|---|---|---|---|---|
| # threads | 1 | 32 | 1 | 32 |
| Memory | 24% | 39% | 11% | 13% |
| LM | 12% | 2% | 47% | 38% |
| Phrase-table | 9% | 5% | 2% | 4% |
| Lex RO | 8% | 2% | 2% | 2% |
| Search | 2% | 0% | 14% | 19% |
| Misc/Unknown | 45% | 39% | 24% | 29% |

Table 4: Profile of %age decoding time


Figure 2: Decoding speed of Moses and our decoder, using the same models

itory.

For all experiments, we used a Dell PowerEdge R620 server with 16 cores, 32 hyper-threads, split over 2 physical processors (Intel Xeon E5-2650 @ 2.00GHz). The server has 380GB RAM. The operating system was Ubuntu 14.04, the code was compiled with gcc 4.8.4 and Boost 1.59[6] and the tcmalloc library.

## 4 Results

### 4.1 Optimizing Memory

Over 24% of the Moses decoder running time is spent on memory management, Table 4. This increases to 39% when 32 threads are used, dampening the scalability of the decoder. By contrast, our decoder spends 11% on memory management and does not significantly increase with more threads.

Figure 2 compares the decoding speed for Moses and our decoder, using the same models, parameters and test set. Our decoder is 4.4 times faster with one thread, and 5.0 times faster using all cores. Like Moses, however, performance actually worsens after approximately 15 threads.

The commit hash was bc5f8d15c6ce4bc678ba992860bfd4be6719cee8

[6]http://boost.org/

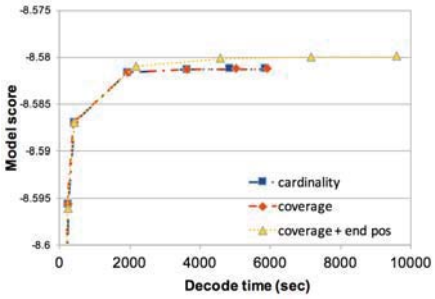



Figure 3: Trade-off between decoding time average model scores for different stack configurations

### 4.2 Stack Configuration

We investigated the effects of the following three stack configurations on model score and decoding speed:

1. coverage cardinality,
2. coverage,
3. coverage and end position of most recently translated source word.

Coverage cardinality is the same as that in Moses and Joshua. Coverage configuration uses one stack per unique coverage vector. *Coverage and end position* of most recently translated source word extends the coverage configuration by separating hypotheses where the position of the last translate word are different, even if the coverages are identical.

This is an optimization to reduce the number of checks on the distortion limit, which is dependent on the last word position. The check is a binary function $d(C_h, e_{hypo}, range_r)$, where $C_h$ is the coverage vector of hypothesis $h$, $e_h$ is the end position of most recent source word that has been translated, and $range_r$ is the coverage of the rule to be applied.

By grouping hypotheses according to coverage *and* end position, the distortion limit only needs to be checked for each group. However, stack pruning occurs on each hypothesis group independently, therefore, potentially affecting search errors and model scores.

Figure 3 present the tradeoff between decoding time and average model, created by varying the cube-pruning pop-limit. None of the different stack configurations significantly outperform the others in either quality or decoding speed. However, the coverage & end position produces slightly higher model scores at higher pop-limits, therefore, we continue to use this configuration throughout the rest of this paper.

We verified that the translation quality of our decoder is comparable to that of Moses in Figure 4, given the same parameters and models. This fits in with our intention of creating a drop-in replacement for the Moses decoder.

### 4.3 Translation Model

In the first optimization, we create a static translation model cache containing translation rules that translates the most common source phrases. This is constructed during phrase-table training based on the source counts. The cache is then loaded when the decoder is started. It does not require the overhead of managing an active cache but there is still some overhead in using a cache. Overall however, using a static cache result in a 10% decrease in decoding time if the optimum cache size is used, Table 5.

For the second optimization, we disable the compression of the target side of the translation rules. This increase the size of the binary files from 17GB to 23GB but the time saved not

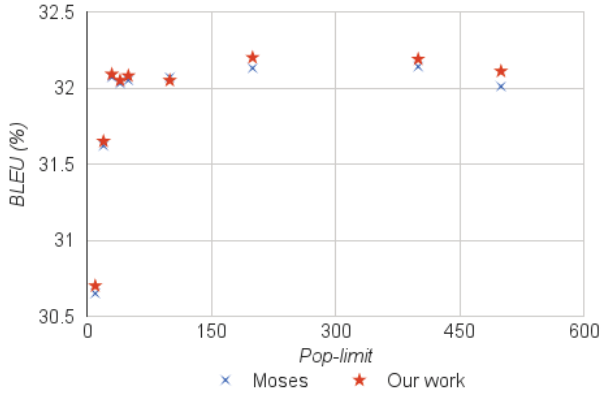



Figure 4: Translation quality for different pop-limits

| Cache size | Decoding Time | Cache Hit %age |
|---|---|---|
| Before caching | 229 | N/A |
| 0 | 239 (+4.4%) | 0% |
| 1,000 | 213 (-7.0%) | 11% |
| 2,000 | 204 (-10.9%) | 13% |
| 4,000 | 205 (-10.5%) | 14% |
| 10,000 | 207 (-9.7%) | 17% |

Table 5: Decoding time (in secs with 32 threads) for varying cache sizes

needing to decompress the data resulted in a 1.5% decrease in decoding time with 1 thread and nearly 7% when the CPUs are saturated, Table 6.

### 4.4 Lexicalized Re-ordering Model

The lexicalized re-ordering model requires a probability distribution of the re-ordering behaviour of each translation rule learnt from the training data. This is represented in the model file as a fixed number of probabilities for each rule, exactly how many probabilities is dependant on the model's parameterization during training. During decoding, a probability from this distribution is assigned to each hypothesis according to the re-ordering of the translation rule.

Rather than holding the model probability distributions in the separate file, we pre-process the translation model file to include the lexicalized re-ordering model distributions for each rule. During decoding, the probability distribution is then taken from the translation model instead of querying a separate file.

This resulted in a significant decrease in decoding time, especially with high number of cores, Figure 5. Decoding speed increased by 40% when using one thread but is 5 times faster when using 32 threads.

### 4.5 Scalability

Figure 6 shows decoding speed against the number of threads used. In our work, there is a constant increase in decoding speed when more threads are used, decreasing slightly after 16 threads when virtual cores are employed by the CPU. Overall, decoding is 16 times faster than single-threaded decoding when all 16 cores (32 hyperthreads) are fully utilized.

This contrast with Moses where speed increases up to approximately 16 threads but then

| # threads | Compressed pt | Non-compressed pt |
|---|---|---|
| 1 | 3052 | 3006 (-1.5%) |
| 5 | 756 | 644 (-14.8%) |
| 10 | 372 | 362 (-2.7%) |
| 15 | 284 | 250 (-12.0%) |
| 20 | 244 | 227 (-7.0%) |
| 25 | 218 | 209 (-4.1%) |
| 30 | 206 | 192 (-6.8%) |
| 35 | 203 | 189 (-6.9%) |

Table 6: Decoding time (in secs with 32 threads) for compressed and non-compressed phrase-tables

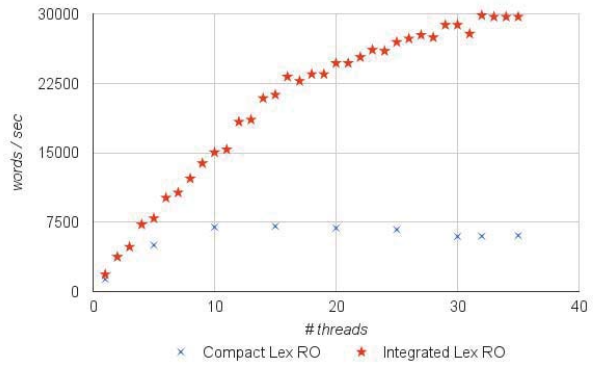



Figure 5: Decoding speed with Compact Lexicalized Re-ordering, and integrated into a model the phrase-table

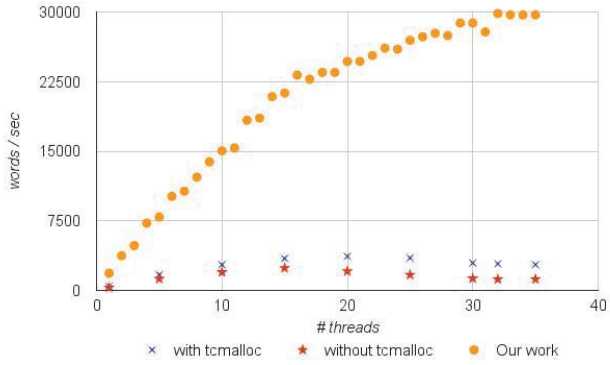



Figure 6: Comparison of decoding speed of our work and Moses (with & without the tcmalloc library

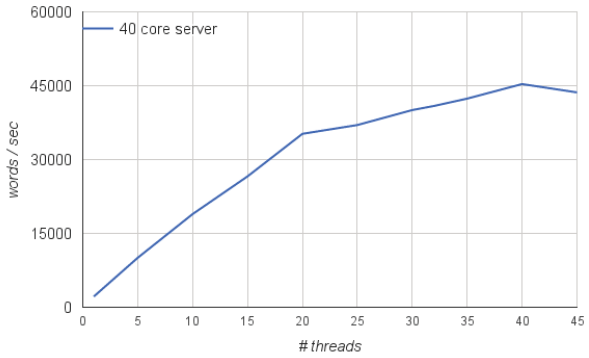



Figure 7: Decoding speed with more cores

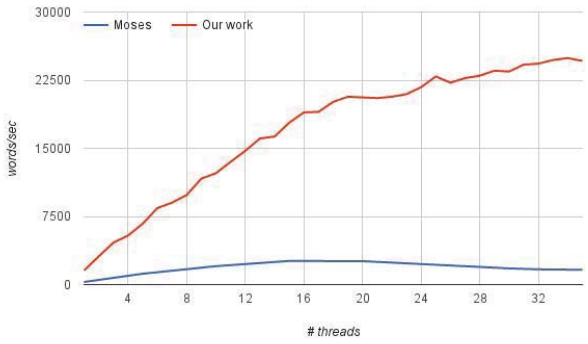



Figure 8: Decoding speed for fr-en model

become slower thereafter. Using the tcmalloc library has a small positive effect on decoding speed but does little to improve scalability

Our work is 4.3 times faster than Moses with a single-thread and 10.4 faster when all cores are used.

### 4.6 Other Models and Even More Cores

Our decoder show no scalability issues when we tested with the same model and tested set on a larger server, Figure 7.

We verify the results with the French-English phrase-based system and test set. The speed gains are even greater than the Arabic-English test scenario, Figure 8. Our decoder is 5.4 times faster than Moses with a single-thread and 14.5 faster when all cores are saturated.

It has been suggested that using a larger language model would overpower the improvements in decoding speed. We tested this conjecture by replacing the language model in the ar-en experiment with a 96GB language model. The time to load of language model is significant (394 sec) and was excluded from the translation speed. Results show that our decoder is 7 times faster than Moses and still scales monotonically until all CPUs are saturated, Figure 9.

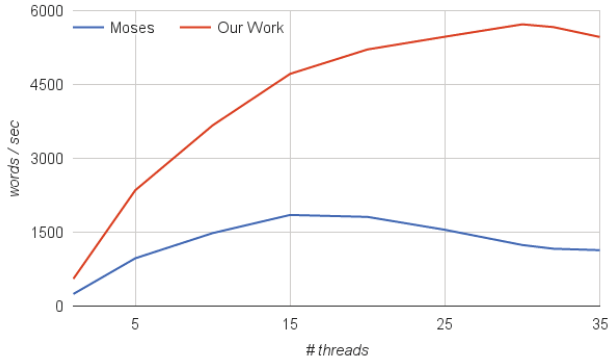



Figure 9: Decoding speed with a large language model

## 5 Conclusion

We have presented a new decoder that is compatible with Moses. By studying the shortcomings of the current implementation, we are able to optimize for speed, particularly for multicore operation. This resulted in double digit gains compared to Moses on the same hardware. Our implementation is also unaffected by scalability issues that has afflicted Moses.

In future, we shall investigate other major components of the decoding algorithm, particularly the language model which has not been touched in this paper. We are also keen to explore the underlying reasons for the scalability issues in Moses to get a better understanding where potential performance issues can arise.

## Acknowledgments

This work is sponsored by the Air Force Research Laboratory, prime contract FA8650-11-C-6160. The views and conclusions contained in this document are those of the authors and should not be interpreted as representative of the official policies, either expressed or implied, of the Air Force Research Laboratory or the U.S. Government.

Thanks to Kenneth Heafield for advice and code.

## References

Bogoychev, N. and Lopez, A. (2016). Fast and highly parallelizable phrase table for statistical machine translation. In *Proceedings of the First Conference on Statistical Machine Translation WMT16*, Berlin, Germany.

Brown, P. F., Della-Pietra, S. A., Della-Pietra, V. J., and Mercer, R. L. (1993). The mathematics of statistical machine translation. *Computational Linguistics*, 19(2):263–313.

Chiang, D. (2007). Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Fernández, M., Pichel, J. C., Cabaleiro, J. C., and Pena, T. F. (2016). Boosting performance of a statistical machine translation system using dynamic parallelism. *Journal of Computational Science*, 13:37–48.

Ghemawat, S. and Menage, P. (2009). Tcmalloc: Thread-caching malloc.

Heafield, K. (2011). KenLM: faster and smaller language model queries. In *Proceedings of the EMNLP 2011 Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland, United Kingdom.

Heafield, K., Kayser, M., and Manning, C. D. (2014). Faster Phrase-Based decoding by refining feature state. In *Proceedings of the Association for Computational Linguistics*, Baltimore, MD, USA.

Junczys-Dowmunt, M. (2012). A space-efficient phrase table implementation using minimal perfect hash functions. In Sojka, P., Hork, A., Kopecek, I., and Pala, K., editors, *15th International Conference on Text, Speech and Dialogue (TSD)*, volume 7499 of *Lecture Notes in Computer Science*, pages 320–327. Springer.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., and Herbst, E. (2007). Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic. Association for Computational Linguistics.

Li, Z., Callison-Burch, C., Dyer, C., Khudanpur, S., Schwartz, L., Thornton, W., Weese, J., and Zaidan, O. (2009). Joshua: An open source toolkit for parsing-based machine translation. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 135–139, Athens, Greece. Association for Computational Linguistics.

Och, F. J., Ueffing, N., and Ney, H. (2001). An efficient A* search algorithm for statistical machine translation. In *Workshop on Data-Driven Machine Translation at 39th Annual Meeting of the Association of Computational Linguistics (ACL)*.

Ortiz-Martínez, D., García-Varea, I., and Casacuberta, F. (2006). Generalized stack decoding algorithms for statistical machine translation. In *Proceedings on the Workshop on Statistical Machine Translation*, pages 64–71, New York City. Association for Computational Linguistics.

Spence Green, D. C. and Manning, C. D. (2014). Phrasal: A toolkit for new directions in statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 114–121. Citeseer.

Tiedemann, J. (2012). Parallel data, tools and interfaces in opus. In *LREC*, pages 2214–2218.

Wuebker, J., Huck, M., Peitz, S., Nuhn, M., Freitag, M., Peter, J.-T., Mansour, S., and Ney, H. (2012a). Jane 2: Open source phrase-based and hierarchical statistical machine translation. In *24th International Conference on Computational Linguistics*, page 483. Citeseer.

Wuebker, J., Ney, H., and Zens, R. (2012b). Fast and scalable decoding with language model look-ahead for phrase-based statistical machine translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 28–32. Association for Computational Linguistics.

Yasuhara, M., Tanaka, T., Norimatsu, J.-y., and Yamamoto, M. (2013). An efficient language model using double-array structures. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 222–232, Seattle, Washington, USA. Association for Computational Linguistics.

Zens, R. and Ney, H. (2007). Efficient phrase-table representation for machine translation with applications to online mt and speech translation. In *HLT-NAACL*, pages 492–499.

Zens, R. and Ney, H. (2008). Improvements in dynamic programming beam search for phrase-based statistical machine translation. In *Proc. of IWSLT*.

# An Effective Diverse Decoding Scheme for Robust Synonymous Sentence Translation

**Youngki Park** ypark@cnue.ac.kr
**Hwidong Na** hwidong.na@samsung.com
**Hodong Lee** hodong.lee@samsung.com
**Jihyun Lee** jihyuns.lee@samsung.com
**Inchul Song** icsong@samsung.com
Samsung Advanced Institute of Technology, Samsung Electronics
Suwon, 443-742, South Korea

**Abstract**

Existing neural machine translation (NMT) systems often mistranslate synonymous sentences into sentences with different meanings. This problem is particularly serious when there is lack of parallel corpora. In this paper, we propose a novel diverse decoding algorithm for accurate translation of synonymous sentences in an NMT system. We observe that the modeling power of NMT models may not be fully utilized because of insufficient exploration of search space by beam search. The proposed algorithm overcomes the problem by expanding search space coverage through diverse decoding. First, it performs greedy search using an NMT model to build an initial candidate list. Then, it expands the list by performing approximate *k*NN search over translation logs to find previously translated results of similar sentences and adding them to the list. Finally, it uses the NMT model again to rescore the candidate list and returns the candidate with the best score. The experimental results show that the proposed scheme enhances the BLEU score significantly over the state-of-the-art NMT system while being much faster.

## 1. Introduction

One of the main barriers to building a machine translation system for commercial use is that existing approaches often mistranslate sentences with the same meaning (synonymous sentences) into sentences with different meanings. For example, we may translate an English sentence, "How can I get to Gangnam Station?", into different Korean sentences, "강남역까지 어떻게 가죠?" / kang.nam.yek.kka.ci (to Gangnam station) ettehkey (how) kacyo (go) [1]? and "강남역까지 가는 길을 알려주세요" / kang.nam.yek.kka.ci (to Gangnam station) ka.nun (going) kil.ul (way) al.lye.cu.sey.yo (let me know). Here, these Korean sentences are synonymous sentences with the same meaning. If we back-translate them into English using Google Translate, the translations are "I'll go to Gangnam?" (an incorrect translation) and "Tell us how to get to Gangnam" (a correct translation), respectively. The second column in Table 1 shows the Korean-to-English translation results of five synonymous source sentences obtained by using Google Translate. As shown in the table, Google Translate translates some synonymous sentences in slightly different forms incorrectly.

Although recent neural machine translation (NMT) systems have improved translation accuracies greatly, they still generate mistranslations for some synonymous

[1] We annotate Korean words by Yale Romanization and indicate their meanings within parentheses. A period (.) indicates the syllable boundary.

| **Source Sentence** | **Google Translate**[2] | **DL4MT-tutorial** (Firat and Cho, 2016) |
|---|---|---|
| kang.nam.yek.kka.ci e.tteh.key ka.cyo ? | (X) I'll go to Gangnam? | (O) How can I go to Gangnam Station? |
| kang.nam.yek.kka.ci e.tteh.key ka.cyo | (X) I'll go to Gangnam | (X) "I'll go to Gangnam Station to Gangnam Station" |
| kang.nam.yek.kka.ci ka.nun kil.i e.tteh.key toyp.ni.kka | (X) How do I get to length Gangnam? | (X) What is going to Gangnam Station? |
| kang.nam.yek.kka.ci e.tteh.key ka.na.yo | (O) How do I get to Gangnam? | (O) How's going to Gangnam Station? |
| kang.nam.yek.kka.ci ka.nun kil.ul al.lye.cu.sey.yo | (O) Tell us how to get to Gangnam | (O) Tell me the way to Gangnam Station |

Table 1. Five example synonymous source sentences and their translations

sentences as shown in the third column of Table 1, which is the translation results of a state-of-the-art NMT system. There are two main reasons for such mistranslations. First, there is lack of parallel corpora for synonymous sentences. Thus, only a limited number of forms of synonymous sentences may be seen during training. Second, standard decoding algorithms, such as beam search, do not efficiently explore the search space. For NMT models with an increased modeling power, the problem of insufficient exploration of search space is more significant. For example, when we performed beam search using an NMT model to translate Sentence $A$ below (the answer is Sentence $B'$), the search returned Sentence $A'$, which is an incorrect translation:

- *Source Sentence A:* 한국의 아침 식단에서 김이 나는 따끈한 밥 한 공기는 대체 불가한 주식으로 오랫동안 여겨져 왔다. / han.kuk.uy (Korean) a.chim (morning) sik.tan.ey.se (menu) kim.i (steam) na.nun (emitting) tta.kkun.han (hot) pap (rice) han (one) kong.ki.nun (bowl) tay.chey (substitution) pul.ka.han (disallowing) cu.sik.u.lo (main dish) o.lays.tong.an (for a long time) ye.kye.cye (considered) wass.ta (have been) .
- *Sentence $A'$*: Kim on the morning calm in korea in the breakfast table in korea, kim's warm, steamed air has long been shadowed for a long week-end.
- *Sentence $B'$*: A hot, steamy bowl of rice was long considered an irreplace-able staple in Korean breakfast.

However, when we computed the scores of Sentence $A'$ and $B'$ using the NMT model, the model assigned a higher score to Sentence $B'$. In other words, the mistranslation is caused not by the NMT model's insufficient modeling power, but the failure of consideration of Sentence $B'$ during beam search. This problem may arise similarly for the synonymous sentences of Sentence $A$: some synonymous sentences, especially seen during training, may produce the correct answer as a candidate during beam search whereas others may not. The main idea of our approach is that if the

[2] https://translate.google.com/
Note the translation results were obtained on 7 July 2016.

translation results of synonymous sentences are all available, we could find the correct answer for any synonymous sentence by rescoring the results (using NMT model).

In this paper, we present a novel diverse decoding algorithm for robust synonymous sentence translation. It is designed to use the modeling power of an NMT model throughout the decoding process by expanding search space coverage. First, it performs greedy search using an NMT model to build an initial candidate list. Then, it expands the list with those candidates that might not be considered during beam search. This is done by finding synomyous sentences through approximate $k$NN search over translation logs and then adding previously translated results of those sentences to the list. Finally, it uses the NMT model again to rescore the candidate list and returns the candidate of the highest score as the final answer. The experimental results show that the proposed scheme increases the BLEU score significantly over the state-of-the-art NMT system while also being much faster. The rest of this paper is structured as follows. In Section 2, we review the state-of-the-art neural machine translation architecture as well as the beam search algorithm. In Section 3, we present a novel diverse decoding scheme that consists of greedy decoding, $n$-best list expansion by (approximate) $k$NN search, and $n$-best rescoring. In Section 4, we conduct extensive experiments to compare our approaches to beam search in a number of configurations. In Section 5, we discuss related work on diverse decoding. Finally, we conclude the paper and present future research directions in Section 6.

## 2. Neural Machine Translation

In order to build our baseline machine translators, we exploit the encoder-decoder architecture with attention mechanism presented by (Bahdanau et al., 2015). In addition, we apply the conditional gated recurrent unit in the hidden layer of the decoder (Firat and Cho, 2016). The encoder and decoder are formulated as follows:

**Encoder** Let $X, Y$ be a source and target sentence, respectively. If $X$ and $Y$ have the $T_x$ and $T_y$ number of tokens (words, subwords, or characters, etc.), we can represent them as $(x_1, \dots, x_{T_x})$ and $(y_1, \dots, y_{T_y})$. For each source word, the encoder produces an embedding vector and feed the vector into a bidirectional recurrent neural network:

$$\vec{h}_t = \vec{\phi}\big(e_x(x_t), \vec{h}_{t-1}\big),$$
$$\overleftarrow{h}_t = \overleftarrow{\phi}\big(e_x(x_t), \overleftarrow{h}_{t+1}\big),$$
$$h_t = [\vec{h}_t; \overleftarrow{h}_t],$$

where $\vec{\phi}/\overleftarrow{\phi}$ is the activation function for the forward/backward hidden state of the encoder, and $e_x(x_t)$ is a word embedding vector of the $t^{th}$ word. Two hidden states of the encoder $\vec{h}_t$ and $\overleftarrow{h}_t$ are concatenated as the hidden state $h_t$ of the $t^{th}$ word.

The context set $C$ consists of the hidden states $h_t$ of the encoder, and the initial hidden state of the decoder $s_0$ depends only on the context set $C$:

$$C = \{h_1, \dots, h_{T_x}\},$$
$$s_0 = f_{init}\big(f_{mean}(C)\big),$$

where $f_{mean}(\cdot)$ averages a set of vectors and $f_{init}(\cdot)$ is a nonlinear function.

**Decoder** A hidden state of the decoder $s_{t'}$ depends on its previous hidden state $s_{t'-1}$, its previous target word $y_{t'-1}$, and its context vector $c_{t'}$:

$$s_{t'} = \phi\big(s_{t'-1}, e_y(y_{t'-1}), c_{t'}\big),$$

where $\phi$ is the activation function of the decoder, and the context vector $c_{t'}$ depends on the context set $C$:

$$c_{t'} = f_{align}(s_{t'-1}, e_y(y_{t'-1}), C).$$

There are additional layers that calculate $p(y_{t'}|y_{<t'}, X)$ based on $s_{t'}$, $c_{t'}$ and $y'_{t-1}$. Here, $y_{<t'}$ is the previous target words before the time step $t'$. Thus the decoder can calculate the log probability of the target sentence given a source sentence as follows:

$$log\ p(Y|X) = \sum_{t'=1}^{T_y} log\ p(y_{t'}|y_{<t'}, X).$$

**Beam Search.** The aim of neural machine translators is to find $argmax_Y(log\ p(Y|X))$. However, the search space is so large that the exact algorithms cannot be used in most practical applications. Instead, we usually use the beam search algorithm for generating approximate results: as a first step, we set $t'$ to 1 and find $argmax^b_{y_{t'}} p(y_{t'}|y_{<t'}, X)$, where $argmax^b$ is the $b$ number of arguments that have the highest values. Then for each selected target word, we find the $argmax^b_{y_{t'}} p(y_{t'}|y_{<t'}, X)$ again for $t' = 2$. Since we have the $b^2$ number of sequences now, we prune the $b$ number of sequences and only keep the top-$b$ sequences that have the highest probabilities. If the end of sentence (EOS) marker is reached, we decrease $b$ by 1 and select the sentence as a hypothesis. This process continues until all of the hypothesis meet the EOS marker ($b = 0$).

## 3. Robust Synonymous Sentence Translation

Although the execution time of beam search is proportional to the beam size, the algorithm is one of the most efficient algorithms when the number of hypotheses maintained is in a reasonable size (smaller than 20). However, we found that the search quality of the beam search algorithm is not good enough, especially for synonymous sentence translation tasks. Even with a large beam size, the beam search often generates translation results with different meaning for synonymous source sentences.

In this section, we propose a novel, effective diverse decoding algorithm in order to cope with this problem. Our approach consists of three main parts: first, we perform the greedy search to get the 1-best (Section 3.1). After that, we obtain the additional $(n-1)$-best list by using similar source sentences and their translation results in the log database (Section 3.2). Last, we rescore the $n$-best list using the original source sentence (Section 3.3).

### 3.1. Greedy Search

Greedy search is one variant of the beam search algorithm (beam size 1). The algorithm selects the word of a highest probability at every time step, and continues this process until the end of sentence marker is found. In other words, given a source sentence $X$, the greedy search selects the word $\tilde{y}_{t'}$ for each time step $t'$:

$$\tilde{y}_{t'} = argmax_{y_{t'}}(\log p(y_{t'}|\tilde{y}_{<t'}, X).$$

After greedy search, we add the obtained result to our $n$-best list. Because we only obtain one translation result by greedy search, we will get additional $n-1$ results in the following subsections.

Although we can get additional candidates by using larger beam size, we prefer to use greedy search because of the two reasons: (1) greedy search is much cheaper operation than beam search. Not only is greedy search about four times faster than the beam search with beam size 2, but also if we double the beam size, the decoding time will be almost doubled, because we have to maintain and update the increased number of hypotheses along with the corresponding hidden states at every time step. (2) As pointed out by

(Cho, 2016; Chung et al., 2016; Li and Jurafsky, 2016), beam search does not provide diverse $n$-best lists effectively so that a large beam size (above 20) does not help to increase the quality of translations in many cases.

### 3.2. $n$-best List Expansion

**$n$-best List Expansion by $k$NN Search.** Let $X^i$, $Y^i$ and $s_0^i$ be the $i^{th}$ source sentence, $i^{th}$ target sentence, and the initial hidden state derived from the $i^{th}$ source sentence, respectively. Recall that $s_0^i$ is obtained as a by-product of $X^i$-to-$Y^i$ translation. Thus whenever we translate $X^i$ into $Y^i$, we can store the pair $\langle s_0^i, Y^i \rangle$ into our own log database.

We assume that there is a log database $D$ that contains more than the $k$ number of $\langle s_0^i, Y^i \rangle$ pairs. Given a source sentence $X^i$, we calculate $s_0^i$ using $X^i$ and finding the $k$ number of $s_0^j$ in $D$ that have the lowest Euclidean distances between $s_0^i$ and $s_0^j$:

$$R = argmin_{j \neq i}^{k} d\left(s_0^i, s_0^j \in D\right).$$

Here, $argmin^k$ returns the k most similar sentences to the source.

Since each $s_0^j \in$ R has its corresponding translation $Y^j$, and |R| = k, we can obtain the corresponding $k$ number of translations. Here, we set $k = n - 1$, because our aim is to obtain additional $n - 1$ candidates.

The rationale behind this approach is that an initial hidden state $s_0$ is a vector that embeds the corresponding source sentence, and that the encoder-decoder with attention could learn the embeddings effectively. In Section 4, we will show that this approach shows the high-level of accuracy even when there are many initial hidden states in the database.

**Approximate $k$NN Search.** Finding $k$-nearest neighbors for a source sentence takes a huge amount of time in the real-world scenario due to the two main reasons: first, an initial hidden state is usually a high-dimensional vector (the dimension of 512, 1024 or more) so that even single Euclidean distance calculation takes nonnegligible time. Second, calculating Euclidean distances for every possible pair consumes a huge amount of time when there are many translation logs in the database. In other words, given a source sentence $X^i$, it takes lots of time to calculate $d\left(s_0^i, s_0^j\right)$ for every $s_0^j$ in the database.

We cope with the first problem by applying spherical hashing (Heo et al., 2012) which is one of the most efficient Locality Sensitive Hashing (LSH) schemes. One main characteristics of spherical hashing is that the algorithm is *data-dependent*, which means that the performance does not greatly depend on the data distributions. It converts a high-dimensional vector into a *signature* (relatively low-dimensional vector) by defining the $H$ number of binary hash functions:

$$sig\left(s_0^j\right) = \langle f_1\left(s_0^j\right), f_2\left(s_0^j\right), \dots, f_H(s_0^j) \rangle,$$

where $f_1(\cdot), f_2(\cdot), \dots, f_H(\cdot)$ are the binary hash functions learnt by the spherical hashing algorithm.

Although spherical hashing itself can find $k$-nearest neighbors effectively, we further improve the algorithm by applying Signature Selection LSH (S2LSH) (Park et al., 2015). The process of this algorithm is as follows: the first step is to generate a *signature pool* consisting of many diverse signatures. Each signature in the signature pool can be generated using $M$ random and distinct integers $r_1, r_2, \dots, r_M$ each ranged from 1 and $H$:

$$sig_l\left(s_0^j\right) = \langle f_{r_1}\left(s_0^j\right), f_{r_2}\left(s_0^j\right), \dots, f_{r_M}(s_0^j) \rangle$$

When a query vector $s_0^q$ is given, the algorithm determines which signatures are the most effective for finding $k$-nearest neighbors of $s_0^q$ in real time. The effectiveness of

signature $l$ for query $s_0^q$ is defined by the percentage of the $k$-nearest neighbors of $q$ among the vectors that have the same signature $l$. As a next step, we find the candidates of $k$-nearest neighbors of $s_0^q$ and calculate the *exact* Euclidean distances between them. By using this pruning process, the unnecessary Euclidean distance computations are significantly reduced while the accuracy is only slightly decreased.

### 3.3. $n$-best Rescoring

The next step is to rescore the $n$-best list obtained by the previous subsections. After the rescoring $n$-best lists, the target sentence with the best score will be returned to the user. There are many ways to rescore the $n$-best: (1) the simplest method is to reuse the decoder that was used in greedy search for calculating the score. (2) If we learn additional decoders with different configurations, then the ensemble model can also be used for rescoring. (3) Another popular rescoring method is to use language model, which has been known as particularly efficient for statistical machine translation.

Let $X^i$, $N$ be a source sentence and its $n$-best list, respectively. Assume we reuse the decoder that was used in the greedy search process. In order to rescore the $n$-best list, we need to know $log\, P(Y^j|X^i)$ for all $Y^j \in N$. If $Y^j$ was obtained by greedy search, we do not have to recalculate $log\, P(Y^j|X^i)$. If $Y^j$ is obtained by $k$NN search, we need to calculate this probability because we only know the value of $log\, P(Y^j|X^l)$. Here, $X^l \neq X^i$.

Note the recalculation process is much faster than beam search (as will be shown in the experimental results of Section 4) since $log\, P(Y^j|X^i)$ for all $Y^j$ can be simultaneously calculated through the network. In addition, all of the hidden states of the decoder can be fed into softmax layer in a batch since we know the target words already. It is interesting because, like human beings, the scoring task is much easier than decoding for neural machine translators.

## 4. Experiments

### 4.1. Experimental Setup

For evaluation, we use the Korean-to-English language pair and exploit a state-of-the-art neural machine translation system introduced in Section 2 as a baseline. The system is based on *Theano* (Theano Development Team, 2016) and we modify the codes to use *Platoon*[3] for multi-gpu training. We use 4 GeForce GTX Titan GPUs.

**Data Preparation.** We use Korean-to-English parallel corpora for training. They consist of about 2.0 million parallel sentences from various types of domains such as news, lectures, emails, etc. These sentences are automatically tokenized, and for Korean language, they are word-spaced, and converted to basic consonants and vowels (Korean alphabets). Then based on these Korean/English alphabets, we construct about 30K subwords using the technique of (Sennrich et al., 2016). Because of a limited amount of GPU memory, we filter out the sentences of more than 50 words and use a minibatch of size 64.

Our test data consist of 3000 parallel sentences. Each Korean sentence has 9 synonymous sentences, and all of the synonymous sentences have the same English references. That is to say, there are 300 distinct English sentences in this test set. The test data is constructed based on the following process: first, we selected the most *popular* 300 English sentences from the user translation data from Web[4]. The selected English sentences are so simple that many people participated in translating them into Korean and that there are more than 10 Korean references for each English sentence. Since there are user ratings data that measure the quality of the Korean references, we selected only 10

[3] https://github.com/mila-udem/platoon
[4] http://usertranslation.naver.com/

Korean sentences of the highest quality for each English sentence and filtered out the other references.

**Model Training.** Our model follows the default setting of the DL4MT-tutorial: we use the GRUs for the recurrent neural networks and exploit the conditional GRU technique for the decoder. The number of hidden layers is 1 for both encoder and decoder. The encoder has 1000 hidden units for each forward and backward direction, the decoder has 1000 hidden units, and each subword is embedded into 500-dimensional vector space. We use the gradient clipping technique (Pascanu et al., 2012) with a threshold 1, and AdaDelta (Zeiler, 2012) as an optimizer. We reshuffle the entire corpus at the start of each epoch.

***k*NN Search.** For $k$NN search, we build two log databases: 3K DB and 100K DB: (1) 3K DB consists of the source sentences from our test set and their translation results. Note each source sentence has only 9 synonymous sentences in the test set; (2) 100K DB contains 97K source sentences from our training corpus and their translation results together with 3K DB.

For approximate $k$NN search, we use the default parameter settings proposed in (Park et al., 2015): we set $H$ to 1000 and $M$ to a random integer ranged from 5 and 15, and set the number of signatures in a signature pool to 500.

***n*-best Rescoring and Evaluation.** Before we rescore the $n$-best list, the score of each candidate is normalized by its decoding length. After generating translation list, the subwords are appropriately concatenated to the words. The translation quality is measured by *Tokenized BLEU* using the *multi-bleu* script from Moses.

### 4.2. Experimental Results

Figure 1 shows a comparison result of beam search (BS), our approach with $k$NN search over 3K sentences (SST 3K), our approach with $k$NN search over 100K sentences (SST-100K), and our approach with approximate $k$NN search over 100K sentences (SST*-100K). The BLEU scores according to the $n$-best list are described on the left-hand side of figure, and the total elapsed time are described on the right-hand side.

Note all of our approaches outperform beam search in terms of BLEU when there are sufficient number of candidates provided. Even in case of using SST*-100K, the BLEU score is higher than beam search. Also, all of our approaches are faster than beam search when $n$ is large enough: (1) SST-3K is the fastest, (2) SST-100K is faster than beam search when $n$ is larger than 16, and (3) the translation speed of SST-100K can be significantly improved by using SST*-100K. Note if we compare BS with SST-3K when $n$ is 64, SST-3K increases the BLEU score by $+0.99$ points and is more than 28 times faster than BS.

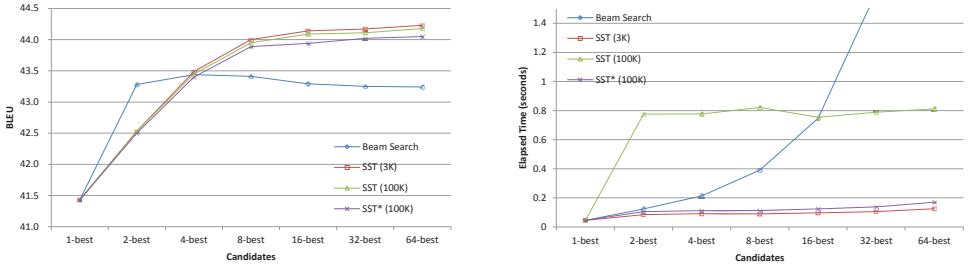



Figure 1. A comparison of beam search and our approaches in terms of BLEU and elapsed time

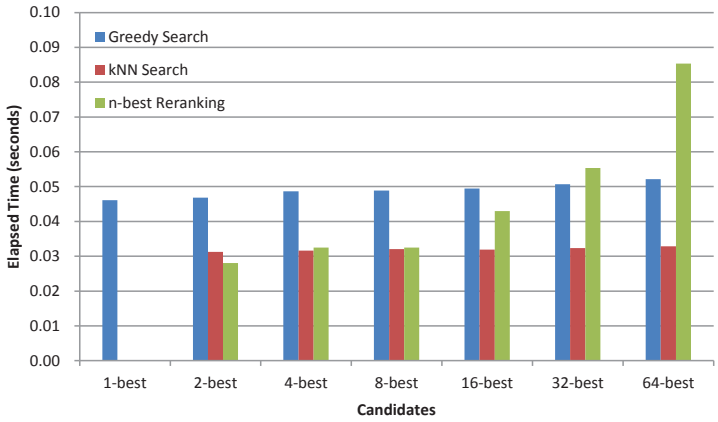



Figure 2. The elapsed time of greedy search, approximate $\boldsymbol{k}$NN search, and $\boldsymbol{n}$-best rescoring for SST*-100K

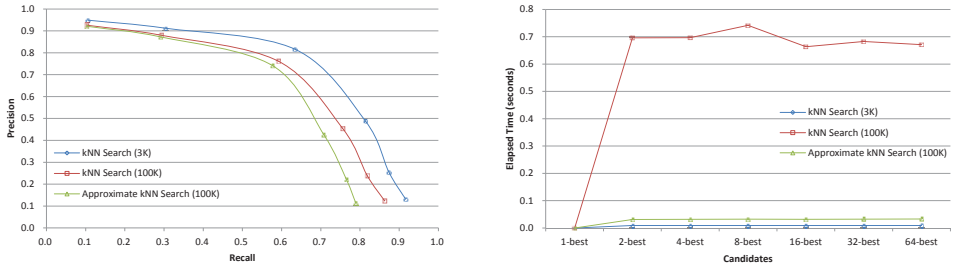



Figure 3. An analysis of recall, precision, and elapsed time for different $\boldsymbol{k}$NN methods

### 4.3. Performance Analysis

**Elapsed Time.** The total elapsed time of our approaches consists of three major components: greedy search time, (approximate) $k$NN search time, and $n$-best rescoring time. Figure 2 shows the elapsed time of each component for SST*-100K: (1) obviously, the greedy search time does not depend on the number of candidates; (2) the $k$NN search time also does not greatly depend on $n$, because the S2LSH algorithm carefully controls the elapsed time; (3) as the number of candidate increases, so does the $n$-best rescoring time.

Note that the $n$-best rescoring time increases slowly as the number of candidates increases: for example, while the rescoring time is 0.0325 seconds when $n$ is 8, it is 0.0430 seconds when $n$ is 16. That is to say, even if we have to rescore twice as much as before, the elapsed time would not be doubled. It is because we do the rescoring in a batch. Even when $n$ is 64, the rescoring time is still lower than 0.1 seconds.

**kNN Search.** Because the $k$NN search methods play a crucial role in our approches, we need to analyze their behaviors. The left-hand side of Figure 3 shows the recall-precision curve of different $k$NN operations: if we $k$NN search over 3K sentences, the recall is more than 80% when precision is 50%, which means that our encoder-decoder architecture embeds the sentences quite well. If we approximate/exact $k$NN search over 100K sentences, the recall and precision are slightly decreased, but still shows the high level of accuracy.

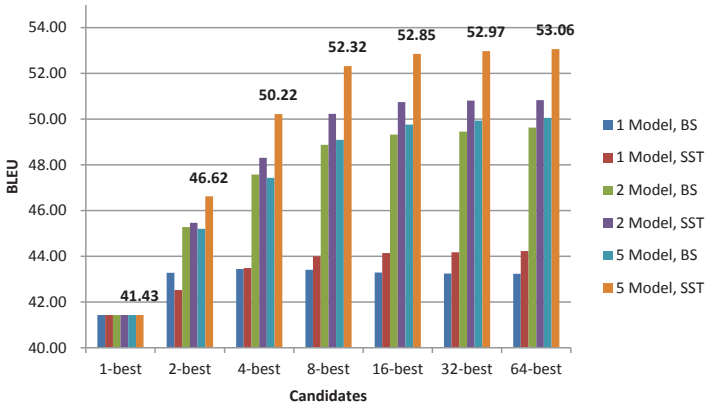



Figure 4. Effect of advanced rescoring models on BS and SST-3K

The right-hand side of Figure 3 shows the practicality of our approach. In summary, even for the large database which contains very diverse sentences with very small portion of synonymous ones, we can find the synonymous sentences in a very efficient way.

**Rescoring Methods.** Through the manual investigation of our $n$-best lists obtained, we found that the quality of candidates is much higher than we expected. Thus we wondered whether the difference of BLEU scores between SST and BS would be higher than before when we use a high-quality model (such as ensemble models) for rescoring.

Figure 4 shows the effect of advanced rescoring models on BS and SST-3K. Here, "1 Model BS" indicates the beam search algorithm with single-model rescoring, "2 Model BS" indicates the beam search algorithm with ensemble-model (2-model) rescoring, and so on. Assume $n = 8$. When we use a single model for rescoring, the difference of BLEU scores between BS and SST-3K is 0.59. However, when we use an ensemble model constructed by 2 different models for rescoring, our approach increases 1.36 and 6.82 BLEU points over 2 Model BS and 1 Model BS, respectively. When we use a more advanced rescoring model (5 Model SST), 3.23 and 8.91 BLEU points are increased over 5 Model BS and 1 Model BS, respectively. This is a surprising result because the difference could be higher if we train the ensemble models more carefully. Although the ensemble models can also be used for decoding instead of rescoring, we do not consider the ensemble model decoding, because this requires significant amount of execution time in practice.

**Examples of Improvements.** Table 2 shows examples of improvements when using our approach. The first column indicates the 14 synonymous source sentences, and the second column shows the result of beam search. The sentences of the third column are obtained by applying our approach to the source sentences, assuming that given a source sentence, the other 13 synonymous sentences are already translated and stored in the log database. The results show that while beam search generates six incorrect translations, our approach generates only three incorrect translations.

## 5. Related Work

(Li and Jurafsky, 2016) proposes a diverse decoding scheme, called *Diversity*. The main idea is that at each decoding time step $t'$, if the words $A$ and $B$ are both preceded by $y_{<t'}$ and $p(A|y_{<t'}, X) > p(B|y_{<t'}, X)$, then it decreases the value of $p(B|y_{<t'}, X)$. For example, suppose that there are four candidates, "he is", "he has", "it is" and "it has", their log-probabilities are $-2.5$, $-2.8$, $-3.0$ and $-3.1$, respectively, and the beam size is

| Source Sentence | Beam Search | Robust SST |
|---|---|---|
| kang.nam.yek.kka.ci e.tteh.key ka.cyo ? | **(X) How do you get to Gangnam Station?** | **(O) How can I go to Gangnam Station?** |
| kang.nam.yek.kka.ci e.tteh.key ka.cyo | (X) "I'll go to Gangnam Station to Gangnam Station." | (X) "Let's go through Gangnam Station." |
| kang.nam.yek.kka.ci ka.nun kil.ul al.lye cu.sey.yo . | (O) Tell me the way to Gangnam Station. | (O) Give me directions to Gangnam Station. |
| kang.nam.yek.kka.ci ka.nun kil.ul al.lye.cwe . | (O) Tell me the way to Gangnam Station | (O) Tell me the way to Gangnam Station. |
| kang.nam.yek.kka.ci ka.nun kil.i e.tteh.key toyp.ni.kka ? | **(X) What is going to Gangnam Station?** | **(O) What is the road going to Gangnam Station?** |
| kang.nam.yek.kka.ci e.tteh.key ka.na.yo ? | (O) How's going to Gangnam Station? | (O) How can I go to Gangnam Station? |
| kang.nam.yek.u.lo e.tteh.key ka.na.yo ? | (O) How's going to Gangnam Station? | (O) How can I get to the Gangnam Station? |
| kang.nam.yek.u.lo ka.nun kil.ul al.lye cu.sey.yo . | (O) Tell me the way to Gangnam Station. | (O) Please give me directions to Gangnam station. |
| kang.nam.yek.u.lo ka.nun kil.ul al.lye.cwe . | (O) Please let me know the way to Gangnam Station. | (O) Tell me the way to Gangnam Station. |
| kang.nam.yek.u.lo ka.nun kil.ul al.lye.cwe | (O) Please let me know the way to Gangnam Station. | (O) Please let me know the path to Gangnam Station. |
| kang.nam.yek.u.lo ka.nun kil.ul al.lye cu.sey.yo | **(X) Please answer the way to Gangnam Station."** | **(O) Give me a way to Gangnam Station.** |
| kang.nam.yek.u.lo ka.nun kil.ul al.lye cu.si.keys.sup.ni.kka ? | (O) Can you give me a way to Gangnam Station? | (O) Can you give me a way to Gangnam Station? |
| kang.nam.yek.u.lo e.tteh.key kal.kka.yo ? | (X) How do you go with Gangnam Station? | (X) How do you go with Gangnam Station? |
| kang.nam.yek.kka.ci e.tteh.key kal.kka.yo | (X) "How do we go from Gangnam Station to Gangnam Station?" | (X) How about going to Gangnam Station |

Table 2. An example of synonymous source sentences and their translations

2. Then, beam search would keep the first and second candidates, namely "he is" and "he has". On the other hand, because $p("is"|"he", X)$ is larger than $p("has"|"he", X)$ and $p("is"|"it", X)$ is larger than $p("has"|"it", X)$, Diversity decreases the log probabilities of the second and fourth candidates by, say, 0.5, which become $-3.3$ and $-3.6$, respectively. Then the first and third candidates, namely "he is" and "it is", would be kept, which results in more diversification.

(Cho, 2016) points out that the main disadvantage of beam search and Diversity is that they may incur high communication overhead when implemented on multiple-GPUs for decoding. Cho proposes a new decoding algorithm, called *noisy parallel approximate decoding (NPAD)*, in which $N$ parallel greedy searches are launched. In each greedy search, weight noise is randomly injected to the transition function of a recurrent neural network to consider a set of semantically similar configurations in the input space. Then the obtained $N$-best list is rescored and the one with the highest score is returned.

Although these state-of-the-art approaches have been proposed to improve beam search, none of them significantly has been able to increase the BLEU scores: (1) while Diversity increases the BLEU score of English-to-French translation by 1.0∼1.2 points with a large beam size (e.g., 200), with a small beam size (e.g., 10), it improves the BLEU score by only 0.03~0.07 points, as shown in (Cho, 2016). Since a large beam size may lead to a prohibitively long decoding time, Diversity is ill-suited for use in a real-world scenario. (2) Similarly, even with 50 simultaneous greedy searches, NPAD does not outperform the beam search with a beam size of 5.

## 6. Conclusion

In this paper, we propose a novel diverse decoding algorithm for accurate translation of synonymous sentences in an NMT system. We observe that the modeling power of NMT models may not be fully utilized because of insufficient coverage of search space by beam search. The proposed algorithm expands search space coverage by using previous translations of synonymous sentences through diverse decoding. The experimental results show that the proposed scheme enhances the BLEU score significantly over the state-of-the-art NMT system while being much faster.

One limitation of our approach is that, given a source sentence, there must be synonymous sentences in our log database. As future work, we plan to extend the applicability of our work by automatically generating and translating synonymous sentences in advance.

## References


Bahdanau, D., Cho, K. and Bengio, Y. (2016). Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv preprint arXiv: 1409.0473.*

Cho, K. (2016). Noisy Parallel Approximate Decoding for Conditional Recurrent Language Model. *arXiv: preprint arXiv: 1605.03835.*

Chung, J., Cho, K. and Bengio, Y. (2016). A Character-level Decoder without Explicit Segmentation for Neural Machine Translation. *arXiv preprint arXiv: 1603.06147.*

Firat, O and Cho, K. (2016). DL4MT-Tutorial: Conditional Gated Recurrent Unit with Attention Mechanism. *https://github.com/nyu-dl/dl4mt-tutorial/blob/master/docs/cgru.pdf.*

Heo, J. Lee, Y., He, J. and C., Y, S. (2012). Spherical Hashing. *In Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16-21, Rhode Island, USA.

Li, J., Jurafsky, D. (2016). Mutual Information and Diverse Decoding Improve Neural Machine Translation. *arXiv preprint arXiv: 1601.00372.*

Park, Y., Hwang, H., Lee, S. (2015). A Fast k-Nearest Neighbor Search Using Query-Specific Signature Selection. *In Proceedings of the 24th ACM International Conference on Information and Knowledge Management*, pages 1883-1886, New York, USA.

Pascanu, R., Mikolov, T. and Bengio, Y. On the Difficulty of Training Recurrent Neural Networks. *arXiv preprint arXiv: 1211.5063*.

Sennrich, R., Haddow, B., Birch, A. (2016). Neural Machine Translation of Rare Words with Subword Units. *arXiv preprint arXiv: 1508.07909*.

Theano Development Team (2016). Theano: A Python Framework for Fast Computation of Mathematical Expressions. *arXiv e-prints arXiv: 1605.02688.*

Zeiler, M. (2012). ADADELTA: An Adaptive Learning Rate Method. *arXiv preprint arXiv: 1212.5701*.

# Ranking suggestions for black-box interactive translation prediction systems with multilayer perceptrons

**Daniel Torregrosa** dtorregrosa@dlsi.ua.es
**Juan Antonio Pérez-Ortiz** japerez@dlsi.ua.es
**Mikel L. Forcada** mlf@dlsi.ua.es
Departament de Llenguatges i Sistemes Informàtics
Universitat d'Alacant, E-03690 Sant Vicent del Raspeig, Spain

**Abstract**

The objective of interactive translation prediction (ITP), a paradigm of computer-aided translation, is to assist professional translators by offering context-based computer-generated suggestions as they type. While most state-of-the-art ITP systems are tightly coupled to a machine translation (MT) system (often created ad-hoc for this purpose), our proposal follows a *resource-agnostic* approach, one that does not need access to the inner workings of the bilingual resources (MT systems or any other bilingual resources) used to generate the suggestions, thus allowing to include new resources almost seamlessly. As we do not expect the user to tolerate more than a few proposals each time, the set of potential suggestions need to be filtered and ranked; the *resource-agnostic* approach has been evaluated before using a set of intuitive length-based and position-based heuristics designed to determine which suggestions to show, achieving promising results. In this paper, we propose a more principled suggestion ranking approach using a regressor (a multilayer perceptron) that achieves significantly better results.

## 1 Introduction

Translation technologies are frequently used to assist professional translators. Common approaches use machine translation (MT) (Hutchins and Somers, 1992) or translation memories (Somers, 2003, Chapter 3) to produce a first (and usually inadequate) prototype of the translation which is then edited by the professional translator in order to produce a target-language text that is adequate for publishing. In both scenarios, the suggestion may be considered by the professional translators as a source of inspiration: they will assemble the final translation on some occasions by accepting and rearranging parts of the proposal, or on other occasions by introducing their own words when an appropriate equivalent is not found in the suggestion.

Our approach, described in a previous work (Pérez-Ortiz et al., 2014), follows a different paradigm known as *interactive translation prediction* (ITP), which, instead of presenting a translation proposal that gets reshaped by the target-language sentence formed in the translator's mind, focuses on offering translation suggestions as the translation is carried out. Most state-of-the-art ITP approaches obtain the suggestions by means of a modified (or tailor-made) statistical machine translation system (SMT) (Koehn, 2004) that is able to provide additional information (such as word alignments, alternative translations, and scores or probabilities for the translation). These systems are able to leverage more information from the bilingual resource than if it were used unmodified as a black-box, but doing so they inherit the common requirements of SMT,

namely, the dependency on the availability of extensive parallel corpora. It is worth noting that integrating other resources of bilingual information would be almost impossible in this kind of systems, as the ITP tool needs the additional information obtained from the underlying SMT engine.

Unlike those previously described, the approach described here is able to use any bilingual resource capable of delivering one or more translations into target language, regardless of how they are obtained and without the need to modify the resource; suggestions are created by generating all possible sub-segments of words in the source-language sentence (up to a given length) and then querying the available bilingual resources for their translations. The nature of these bilingual resources is not limited to MT systems, but they may also include translation memories, dictionaries, catalogues of bilingual phrases, or any combination of them. A neural-based machine learning algorithm trained on features extracted from the source sentence, from the current prefix of the target sentence, and from the translated sub-segments is used to rank and select which suggestions to show at each time step. Not having to rely on the inner workings of each system allows us to integrate new resources without modifying how the ITP system works; similarly, we do not need to modify the resources in any way. Both these features make it possible to use any resource the professional translator has access to in a seamless way.

These translated sub-segments are then offered to the user as the translation is being typed with the objective of saving keystrokes and, hopefully, time. In previous works we have explored the performance of our approach considering rule-based MT systems (Pérez-Ortiz et al., 2014) and in-domain and out-of-domain SMT systems (Torregrosa et al., 2014), using a naïve strategy based on a number of intuitive heuristics to rank and select a few suggestions that are shown to the user. In this paper, however, we aim to improve the strategy by using machine learning techniques. The improvement presented here is twofold: on the one hand, more coherent and principled models are introduced; on the other hand, the results we achieve are significantly better.

The remainder of the paper is organised as follows. After reviewing the state-of-the-art of ITP in Section 2, we describe our method for generating and ranking translation suggestions from bilingual resources in Section 3, emphasizing the differences between the sounder approach presented in this paper and the former heuristic ranking method. We then introduce in Section 4 our experimental set-up and show the results of its evaluation. Finally, we discuss the results and propose future lines of research in Section 5.

## 2 Related work

The systems which have most significantly contributed to the field of ITP are those built in the pioneering TransType project (Foster et al., 1997; Langlais et al., 2000), and its continuation, the TransType2 project (Macklovitch, 2006). These systems observe the current partial translation already typed by the user and, by exploiting an embedded statistical MT engine, propose one or more completions that are compatible with the sentence prefix entered by the user. The proposals offered may range from one or several words to a completion of the remainder of the target sentence. An automatic best-scenario evaluation with training and evaluation corpora belonging to the same domain (Barrachina et al., 2009) showed that it might theoretically be possible to use only 20–25% of the keystrokes in comparison with the unassisted translation for English–Spanish translation (both directions) and around 45% for English–French and English–German.

A number of projects continued the research where TransType2 had left off. Caitra (Koehn, 2009) is an ITP tool which uses both the phrase table and the decoder of a statistical MT system to generate suggestions. Researchers at the Universitat Politècnica de València have also made significant improvements to a TransType2-style system such as allowing users to accept discontinuous segments of the suggested translation (Domingo et al., 2016). The CASMACAT

project (Koehn et al., 2015) followed the same line of research, improving ITP using active and on-line learning (Alabau et al., 2014). Commercial translation memory systems also integrate some form of ITP as one of their basic features (see, for example, SDL Trados AutoSuggest 2.0[1]), and new translation tools such as Lilt (Green et al., 2014) focus on delivering ITP on an user-friendly web interface.

## 3 Method

As already described in other articles (Pérez-Ortiz et al., 2014; Torregrosa et al., 2014), our method starts by generating all possible whole-word sub-segments of the source-language sentence $S$ of lengths $l \in [1, L]$, where $L$ is the maximum source sub-segment length measured in words.[2] The resulting sub-segments are then translated by means of a bilingual resource (or a combination of bilingual resources). The set of *potential proposals* $P^S$ for sentence $S$ is made up of suggestions $p$, which in turn are made up of a target-language sub-segment $t_p$ and the starting $b_p$ and ending $e_p$ positions of the corresponding sub-segment in $S$.

These suggestions are then offered as the translation $T$ is being typed.[3] Let $\Pr(x)$ be the character-level set of prefixes of a string $x$, $T_k$ the $k$-th word of $T$, and $\hat{T} = T_1...T_{k-1}\hat{w}$ the partially translated sentence where $\hat{w} \in \Pr(T_k)$ is the currently typed prefix of $T_k$; we define the set of *compatible suggestions* $P^S_{\text{compatible}}$ as

$$P^S_{\text{compatible}}(\hat{w}) = \{p \in P^S : \hat{w} \in \Pr(t_p)\}$$

For example, given $S =$*"Mi sastre está sano"*, with $L = 2$, the set of potential proposals will be $P^S = \{$ *"My"*, *"My tailor"*, *"tailor"*, *"tailor is"*, *"is"*, *"is healthy"*, *"healthy"*$\}$; with $\hat{T} =$*"My t"* and $\hat{w} =$*"t"*, the set of compatible suggestions would be $P^S_{\text{compatible}} = \{$ *"tailor"*, *"tailor is"*$\}$.

As studied in (Pérez-Ortiz et al., 2014), the number of compatible suggestions depends not only on the value of $L$, but also on the specific word prefix; for example, when users type the letter *d* when translating a long sentence into Spanish, they will probably obtain a significant number of suggestions starting with *de*[4] originating from sub-segments located in different source positions. Obviously, only suggestions originating from the part of the source sentence currently being translated may be useful, but this position is difficult to determine unambiguously. As we do not expect users to tolerate a long list of suggestions, more elaborated strategies are needed both to rank suggestions and to reduce the list to a manageable size.

### 3.1 Previous approach

We have already proposed (Pérez-Ortiz et al., 2014) a naive way of ranking suggestions, based on the following assumptions:

- the source-language sentence $S$ and the target-language sentence $T$ have similar lengths, and translation is mainly monotonous; useful suggestions for the $n$-th word of $T$ will be generated from sub-segments close to the $n$-th word of $S$;

[1] `http://www.translationzone.com/products/trados-studio/autosuggest/`

[2] Suitable values for $L$ will depend on the bilingual resource: on the one hand, we expect higher values of $L$ to be useful for high-quality MT systems, such as those translating between closely related languages, since adequate translations may stretch to a relatively large number of words; on the other hand, $L$ should be kept small for low-quality MT systems whose translations quickly deteriorate as the length of the input sub-segment increases; of course, $L$ will be small for resources such as dictionaries.

[3] While $T$ is fixed during automatic tests, professional translators may change their minds during the process.

[4] The preposition *de* ('of') is one of the most frequent words in Spanish texts.

- long suggestions are seldom useful,[5] but when used there is a significant effort reduction;
- short suggestions are usually compatible,[6] but do not save too much effort when used.

We therefore devised the following selection scheme: when the user translates the $k$-th word, the shortest and longest (measured in number of words) suggestions originated from the closest position in $P^S_{\text{compatible}}$ are offered, followed by the shortest and longest of the second closest position, or, if no other position generated compatible suggestions, the second shortest and the second longest suggestions from the previous position, and so on, up to a maximum number of suggestions $M$.[7]

This heuristic performed remarkably well in spite of the simplicity of the approach: during a conducted preliminary test using $M = 4$ with human translators (Pérez-Ortiz et al., 2014), savings in the range of 25–65% keystrokes (depending on the language pair) were achieved, without any explicit complaints from users about being offered too many suggestions.

### 3.2 Neural network model

The approach discussed in the preceding section can still be improved: on the one hand, more rigorous and principled models rather than intuitive heuristics can be used; on the other hand, if we get a better ranking of suggestions we can reduce the number of suggestions offered (reducing the cognitive effort used for reading and selecting suggestions), the number of keystrokes or both. Consequently, we propose to replace the previous intuitive heuristics with a ranker based on a multilayer perceptron (Duda et al., 2000, Chapter 6).Four different systems will be trained, *Full feature set with usable suggestions*, *Full feature set with winning suggestions*, *Reduced feature set with usable suggestions*, and *Reduced feature set with winning suggestions*, depending on the set of features and the kind of suggestions they will learn to identify:

- *Full feature set*: the full feature set that will be discussed in Subsection 3.3;
- *Reduced feature set*: a reduced feature set consisting of only two features, the length of the suggestion and the normalized distance, as will be discussed in Subsection 3.3. This that means that the model has access to similar information to that available to the previous intuitive heuristic method.
- *Winning suggestions*: the set of *winning suggestions*, those that get chosen during the automatic evaluation procedure described in Section 4. Winning suggestions are given a score of 1, and the rest get a score of 0.[8]
- *Usable suggestions*: the set of *usable suggestions*, namely those which, for the current partially translated sentence $\hat{T}$, could be used for advancing the translation, but are not necessarily are part of the suboptimal sequence of actions the greedy automatic evaluation procedure executes. For instance, given $S =$"Un coche rojo", $T =$"A red car", $\hat{T} =$"A r", $p_1 =$"red" and $p_2 =$"red car" both are deemed as usable, even when $p_2$ would be more advantageous. Usable suggestions are given a score of 1, and the rest get a score of 0.

[5] Our automatic testing strategy (see Section 4) only accepts suggestions that exactly match the provided reference. However, professional translators may accept suggestions that slightly differ from their initial translation, editing the suggestion or even adapting their planned translation. However, we lack a formal model that reflects this behaviour; devising one is a requisite for conveying a more human-like automatic evaluation.

[6] Usually, short words such as determinants, prepositions, and non-ambiguous nouns are correctly translated even by low quality resources.

[7] A system that reversed this order picking the longest and shortest suggestions, a second one picking first the longest of each position and a third one picking first the shortest were also tested, but performed worse.

[8] During training, the desired output values will be 0 and 1, but during testing the network output will be a real value between 0 and 1 that correlates with the goodness of the suggestion.

### 3.3 Features

We will use features $f_i, 1 \leq i \leq 30$ for each $p$. As multilayer perceptrons cannot work with nominal features, those will be transformed into multiple one-hot-encoded binary features and then treated as numeric. Likewise, binary features are treated as numeric. The description of this transformation will not be included in the definition of the features to avoid overcomplicating it; when applied, the actual number of features grows to 79.

**Length of the suggestion** The length of the span the suggestion is generated from $f_1 = e_p - b_p$ and the length of the translated sub-segment $f_2 = |t_p|$, both at the word and the character ($f_3, f_4$) level. As discussed in Subsection 3.1, long suggestions are seldom used under our testing conditions.

**Position of the suggestion** A set of features that relate to the position where each suggestion comes from and where it is (potentially) offered. The features include the absolute position in the source sentence $f_5 = b_p$ and the position being currently worked on in the target sentence $f_6 = k$, the position normalized by the length of the sentence $f_7 = b_p/|S|$ and $f_8 = k/|T|$,[9] the distance between source and target positions, both with absolute positions $f_9 = k - b_p$ and their normalized counterparts $f_{10} = k/|T| - b_p/|S|$, and their position ratios $f_{11} = k/b_p$. These features help to determine how far we are from the suggestion source: long sentences will have potentially higher values for the differences, but lower values on the ratio; the opposite stands for short sentences. We also define ($f_{12}...f_{18}$), the equivalent feature set for character level positions and distances.

**Position and length** A set of 3 nominalized features ($f_{19}, f_{20}, f_{21}$) that relate to the position and the length of the suggestion. Each feature takes a value in the $\{\mathrm{short}, \mathrm{long}\} \times \{\mathrm{close}, \mathrm{far}\}$ set. We classify a suggestion as $\mathrm{short}$ if it has 2 or less words ($f_2$) or 10 or less characters ($f_4$), and $\mathrm{long}$ otherwise. We classify a suggestion as $\mathrm{close}$ if it is 3 or less words away ($f_9$), 20 or less characters away ($f_{15}$) or the ratio position ($f_{11}$) is lower than 1.2; $\mathrm{far}$ otherwise. The feature $f_{19}$ uses word-level length and distance, $f_{20}$ uses character-level length and distance, and $f_{21}$ uses word-level length and ratio.

**Distance distribution** Given a training set, we compute the average $\overline{x}$ and standard deviation $\sigma$ values for the distribution of normalized distances and position ratios for the *winning suggestions* set (as described in Subsection 3.2), we define four features:

- $f_{22} = (f_{10} - \overline{x})/\sigma$
- a nominal feature $f_{23}$ that has 4 different classes depending on the relationship between $f_{10}$ and the distribution: less than half $\sigma$ away from $\overline{x}$, a full $\sigma$ away, $2\sigma$ away, or further;
- two more features ($f_{24}$ and $f_{25}$) similar to the two above, but replacing $f_{10}$ with $f_{11}$, using their respective average and deviation.

**Starting letter of the suggestion** As discussed in Section 3, the starting letter of a word is related to the size of the set of compatible suggestions $P^S_{\mathrm{compatible}}$. The nominal feature $f_{26}$ takes the value of the first letter of the suggestion (ignoring the capitalization) if it belongs to the English alphabet, and replaced with a generic $other$ token otherwise.

**Last action taken** The binary feature $f_{27}$ represents the action we took for the previous word $T_{k-1}$: whether we typed it or it was part of an accepted suggestion.

[9] During testing, when the length of $T$ cannot be known, we assume $|T| = |S|$.

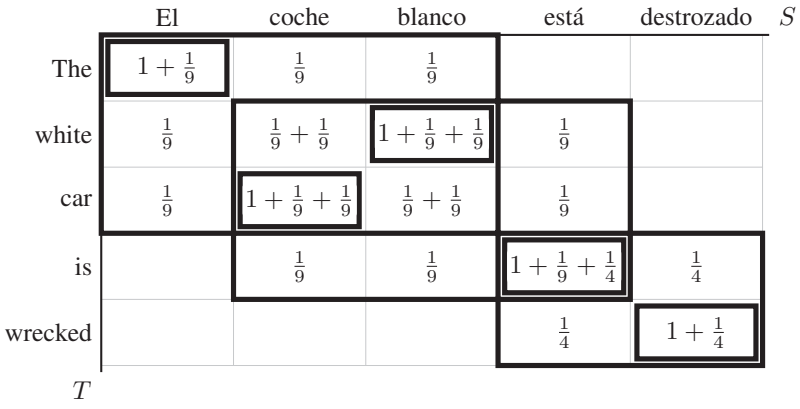



Figure 1: *Alignment strengths* for $S$ =*'El coche blanco está destrozado'*, $T$ =*'The white car is wrecked'*, $P^S_{\text{compatible}}$ = { 'The', 'The white', 'The white car', 'car', 'white car', 'white car is','white', 'white is', 'white is wrecked', 'is', 'is wrecked', 'wrecked'}. Each suggestion is given an *alignment strength* of 1, that gets evenly split along the surface of the suggestion. Some suggestions like 'white is wrecked' cannot be aligned. Each position has a label with the *alignment strength* over it; positions with more *alignment strength* are more likely to be aligned.

**Relationship to the last used suggestion** The nominal feature $f_{28}$ represents the relationship of the current suggestion $p$ with the last used one $p'$. It takes 5 possible values:

- $p$ ends before $p'$, $e_p + 1 < b_{p'}$
- $p$ is contiguous and is placed immediately before $p'$, $e_p + 1 = b_{p'}$
- $p$ and $p'$ overlap,[10] $b_p \in [b_{p'}, e_{p'}] \vee e_p \in [b_{p'}, e_{p'}]$
- $p$ is contiguous and is placed immediately after $p'$, $b_p - 1 = e_{p'}$
- $p$ starts after $p'$, $b_p - 1 > e_{p'}$

At the beginning of the translation, where no suggestion has been used, all the suggestions are deemed as belonging to the last category (starts after $p'$).

**Light alignment model** The light alignment model described in (Esplà-Gomis et al., 2012) performs similarly to other state-of-the-art word-alignment methods using previously existing bilingual resources without needing any training procedure. The model relies on a intuitive idea: each sub-segment that contains $S_j$ (the $j$-th word of $S$) whose translation covers $T_y$ (the $y$-th word of $T$), and vice-versa, increases the likelihood (measured in *alignment strength*) of $S_j$ and $T_y$ to be aligned. An example of how the model works is shown in Figure 1.

In the same way as the ITP system described here, all possible whole-word sub-segments of $S$ and $T$ up to a given length are generated, and then translated using a bilingual resource (MT in (Esplà-Gomis et al., 2012)), although we cannot use the sub-segments that are the product of translating sub-segments of $T$: $T$ only becomes available during the process as the user is

[10]This would mean a given part of $S$ participates on the generation of different parts of $T$, which, in general terms, is unlikely to be desirable.

typing it (which means nothing at all is available at the start); translating these sub-segments as the translator types could degrade the performance of the system to the point that it coud not be effectively used, specially if working with complex MT systems, when the user computer has low processing power or on-line.

We take the original idea one step further: suggestions that cover an area with high *alignment strength* are more likely to be aligned; hence those covering the position currently being worked on ($k$) are more likely to be used. To this end, we analyze the set of suggestions that overlap with the end of the typed prefix $\hat{T}$. Let $\mathrm{Pr}$ and $\mathrm{Suf}$ be the character-level set of prefixes and suffixes of a given string, we define $\mathrm{extender}$, the set of suggestions that overlap with and extend the end of the current typed prefix $\hat{T}$

$$\mathrm{extender}(\hat{T}) = \{p \in P^S : \mathrm{Pr}(t_p) \cap \mathrm{Suf}(\hat{T}) \neq \varnothing\}$$

As discussed by Esplà-Gomis et al. (2012), we operate on the idea that sub-segment alignment applies *alignment pressure*: the larger the surface covered, the weaker the confidence in the alignment. Each sub-segment pair is given an *alignment strength* of 1 unit. This strength is split evenly along the *surface* of the suggestion as measured in *square words*. So, the force exerted on each position is

$$v_p = \frac{1}{(e_p - b_p)|t_p|}$$

To estimate which position $j$ of $S$ is being worked on, we look at how much *alignment strength* is exerted on it. For this mean, we define

$$W(j, \hat{T}) = \sum_{p \in \mathrm{extender}(\hat{T})} \begin{cases} v_p & \text{if } j \in [b_p, e_p] \\ 0 & \text{otherwise} \end{cases}$$

As discussed, we interpret high *alignment strength* as high confidence in an alignment; positions ($j$) of $S$ with higher pressure for the position $k$ of $T$ currently being worked on are more likely to be aligned. Hence, suggestions covering $k$ whose area collects more *alignment strength* have a higher probability of being direct translation of the segment of $S$ being currently translated. Moreover, we do not want to realign already used suggestions: for this mean we define $P^S_{\mathrm{accepted}}$, the set of accepted suggestions. Having this in consideration, we add the combined *alignment strength* pressing the area under each suggestion:

$$f_{29} = \sum_{j \in \mathrm{span}(p)} \begin{cases} W(j, \hat{T}) & \text{if } p \notin P^S_{\mathrm{accepted}} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Conversely, we can exploit this alignment model to discredit suggestions originating in positions $j$ that may have been already covered in $T$. To this end, we calculate the *alignment strength* over the already translated part of the sentence $\overline{T} = T_1 ... T_{k-1}$. Let $\mathrm{occurrences}(t_p, T)$ be the function that returns the number of times $t_p$ appears as a subsequence (substring) of $T$, we define the set of suggestions that overlap with $\overline{T}$,

$$\mathrm{overlap}(\overline{T}) = \{p \in P^S : \mathrm{occurrences}(t_p, \overline{T}) > 0\}$$

As the target text of each suggestion $\bar{t}_p$ may appear more than once in $\hat{T}$, we are unsure of which alignment is the correct one. For addressing this problem, we split the *alignment strength* between the different matches,

$$u_p = \frac{v_p}{\mathrm{occurrences}(t_p, \overline{T})}$$

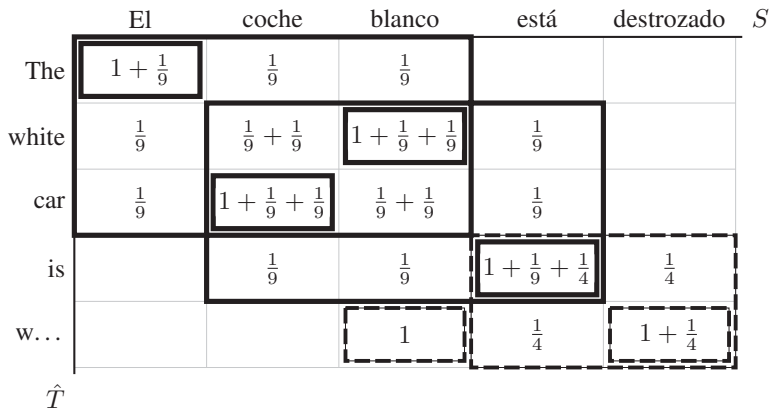



Figure 2: *Alignment strengths* for $S$ ='*El coche blanco está destrozado*', $T$ ='*The white car is wrecked*', $\hat{T}$ ='*The white car is w...*', $P^S_{\text{compatible}}$ = { 'The', 'The white', 'The white car', 'car','white car', 'white car is', 'white', 'white is', 'white is wrecked', 'is', 'is wrecked', 'wrecked'}. Rectangles with solid outlines represent suggestions in overlap, those with dashed outlines represent suggestions in extender. Assuming 'white' $\notin P^S_{\text{accepted}}$ ('white' has not been used when typing $\hat{T}$), 'white' has $f_{26} = 1$, eq. 1 and $f_{27} = 1 - (1 + 6/9)$, eq. 2, 'wrecked' has $f_{26} = f_{27} = 1 + 1/4$

We define the past *alignment strength* function, that includes those suggestions that could have been used for $\overline{T}$, even if those were never offered or used by the translator:

$$W_{\text{past}}(j, \overline{T}) = \sum_{p \in \text{overlap}(\overline{T})} \begin{cases} u_p & \text{if} j \in [b_p, e_p] \\ 0 & \text{otherwise} \end{cases}$$

We define a second feature that discredits suggestions originating from positions that have evidence (in the form of suggestions that overlap with $\overline{T}$) of having already been covered in the translation:

$$f_{30} = \sum_{j \in \text{span}(p)} \begin{cases} W_{\text{past}}(j, \overline{T}) & \text{if } p \notin P^S_{\text{accepted}} \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

An example of how this both features work is discussed in Figure 2.

## 4 Experimental setup and results

As explained in Section 3.2, four different configurations are trained. Training, development and test data are extracted from a corpus with 15,000 English–Spanish sentences extracted from Europarl (Koehn, 2005) version 7, a collection of proceedings from the European Parliament; 11,000 sentences were used as training set, 1,000 as development or validation set and the remaining 3,000 as test set. As a bilingual MT engine capable of providing the translation of the sub-segments we used the free/open-source statistical MT system Moses (Koehn et al., 2007) trained over 155,760 independent sentences from the same corpus, following the standard procedure for training a baseline system.[11] The evaluation was conducted for the translation of texts from English to Spanish.

[11]The corpora is available at `http://www.statmt.org/europarl`. The sentences match those we used in a previous paper (Torregrosa et al., 2014).

To generate the training set, using a source sentence and a reference, an automatic system iteratively considers the first letter of each word and evaluates all the possible suggestions; those that fully match the following words of the reference are tagged as *usable* in that context, and those that would be part of the greedy suboptimal sequence of actions that leads to the best performance are tagged as *winning* in that context.

In order to measure the performance of each configuration, we use the keystroke ratio (KSR), that is, the ratio between the actual number of keys pressed for typing the translation and the length in characters of the translation, as described by Langlais et al. (2000). Accepting a suggestion, no matter its rank, costs one keystroke. It is worth noting that this metric does not measure the time or effort needed to read the suggestions, and does not penalize the offering of inappropriate suggestions in any way. The automatic evaluation system[12] is similar to the one described in the previous work (Pérez-Ortiz et al., 2014): it tries to emulate the behaviour of a user that, given a source sentence $S$, mentally generates a translation $T$,[13] then proceeds to type it, reading every offered suggestion and accepting the longest one that matches exactly $T$, if any. Suggestions need to be full-word translations: if $T_k$ is "thesaurus", the suggestion "the" will not be accepted. The system types the first letter of the next word in $T$, evaluates all the suggestions, and either chooses the longest one that matches with $T$ or types the rest of the word,[14] until $T$ is completed.[15] The models have been tested with different limits for the maximum number of suggestions $M \in [1, 8]$, sorted in descending order according to the multilayer perceptron output value.

All the multilayer perceptron configurations have three layers: input, hidden and output. Those trained with the reduced feature set only have two input neurons; perceptrons dealing with the full feature set have 75 input units. We will explore different values for the number of units in the hidden layer. All configurations have just one output neuron, and are fully connected. All the neurons but the output neuron have logistic activation functions; as we are estimating a regression function, the output neuron has the identity activation function.

For the training procedure, we use backpropagation with mean squared error (MSE) as the error function to optimize, and no momentum or regularization of any kind. As neural networks have trouble dealing with local minima (Gori and Tesi, 1992), each perceptron has been trained five times with different random initializations. Table 1 shows that there is a strong correlation between MSE and KSR; as a result of this we can presume that the systems with lower minimum squared error MSE will do a better job ranking the suggestions, so we select the model that achieves the lowest MSE out of the 5 different initializations. While the weights were updated after computing the error of every event in the training set, the decision to stop the training (also known as convergence condition) is based on the validation set, in order to minimise the risk of overfitting. We will use these algorithms as implemented in the free open source neural network library FANN (Nissen, 2003).

As hyperparameters, we explored different values for the learning rate and the number of neurons in the hidden layer. All the configurations were tested with learning rates $\alpha \in \{10^{-2}, 10^{-3}, 10^{-4}\}$. The configurations using the reduced feature set were trained and tested with $N \in \{2, 4, 8, 16\}$ neurons in the hidden layer; the ones using the full feature set used

[12] While human evaluation is preferable, it is too expensive for systematically testing every model.

[13] During automatic evaluation, a reference translation is provided.

[14] This differs from the previously described approach, where the suggestions were evaluated after every keypress, rather than only evaluating them at the start of each word.

[15] This evaluation model assumes a *greedy* left-to-right longest-match coverage and, as such, produces a suboptimal solution. Experiments using an optimal coverage strategy that find the global optimum sequence of actions rather than the suboptimal one achieved by the the greedy left-to-right longest match strategy, have been conducted, achieving keystroke ratio improvements under $1\%$. As such experiments are much more computationally expensive, the left-to-right longest-match strategy is used.

| Configuration \ $M$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| Full set, Usable | 0.87 | 0.88 | 0.88 | 0.88 | 0.88 | 0.87 | 0.87 | 0.86 |
| Full set, winning | 0.92 | 0.95 | 0.96 | 0.96 | 0.96 | 0.95 | 0.95 | 0.95 |
| Reduced set, Usable | 0.91 | 0.94 | 0.93 | 0.93 | 0.93 | 0.92 | 0.91 | 0.91 |
| Reduced set, winning | 0.90 | 0.90 | 0.89 | 0.87 | 0.87 | 0.86 | 0.85 | 0.84 |

Table 1: Pearson correlation coefficients between minimum squared error (MSE) and keystroke ration (KSR) for each maximum number of suggestions $M$ and configuration. Every configuration shows strong positive correlation between MSE and KSR.

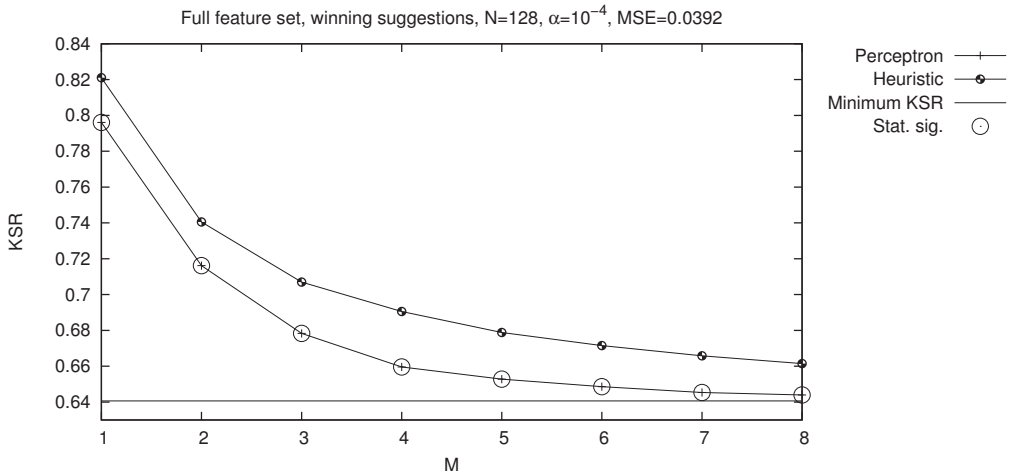



Figure 3: KSR for the multilayer perceptron hyperparameter configuration trained using the *full feature set* and the *winning suggestions* set that got the best MSE. The system beats the baseline in a statistically significant way for every tested value of the maximum number of suggestions $M$.

$N \in \{2, 8, 32, 128\}$ neurons in the hidden layer.

We will also use the previous intuitive heuristic approach, as described in Subsection 3.1 (referred to as *heuristic* onwards) as a baseline. Paired bootstrap resampling (Koehn, 2004) is performed between the different models (including the *heuristic* approach) using 1000 iterations and $p \leq 0.05$; the best statistically significant KSR values achieved will be denoted with a circle.

Results show that the perceptrons trained with the *winning suggestions* and full feature set (Figure 3) perform notably better than the ones trained with the reduced set (Figure 4). In every figure, the line "Minimum KSR" denotes the best KSR we can achieve offering all the suggestions using the current methodology, and circled points denote the best statistically significant KSR. While several systems trained with the full feature set performed statistically significantly better than the heuristic approach, no reduced feature set system manages to outperform the *heuristic* baseline with $M = [1, 2]$. Those trained with the *usable suggestions* perform similarly to the ones trained with the *winning suggestions*: the best performing systems are not statistically significantly better or worse than the best ones using *winning suggestions* for most values of the maximum number of suggestions $M$, but, overall, less configurations manage to beat the baseline. Figure 5 compares the performance of the best system for each configuration.

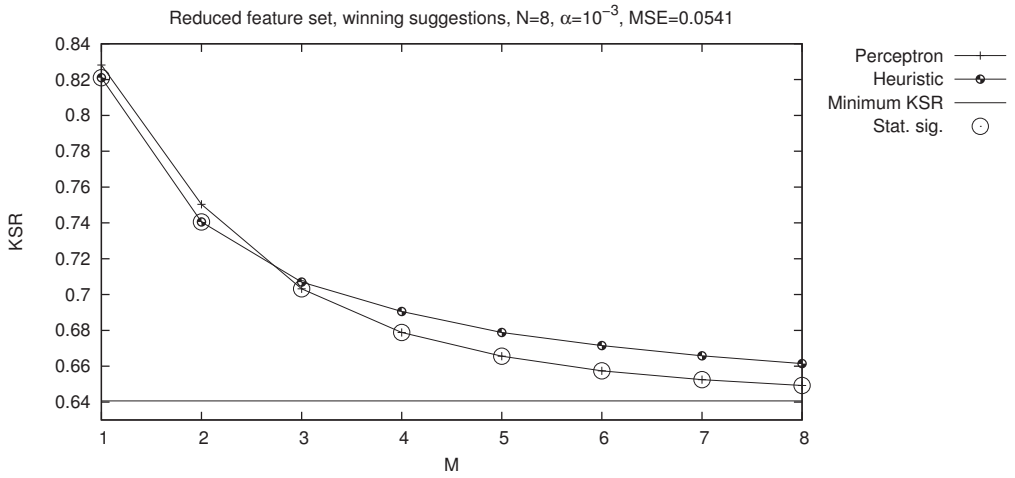



Figure 4: KSR for the multilayer perceptron hyperparameter configuration trained using the *reduced feature set* and the *winning suggestions* set that got the best MSE. The system beats the baseline in a statistically significant way for every tested value of $M$, but for $M = 1, 2$, where it achieves significantly worse KSR.

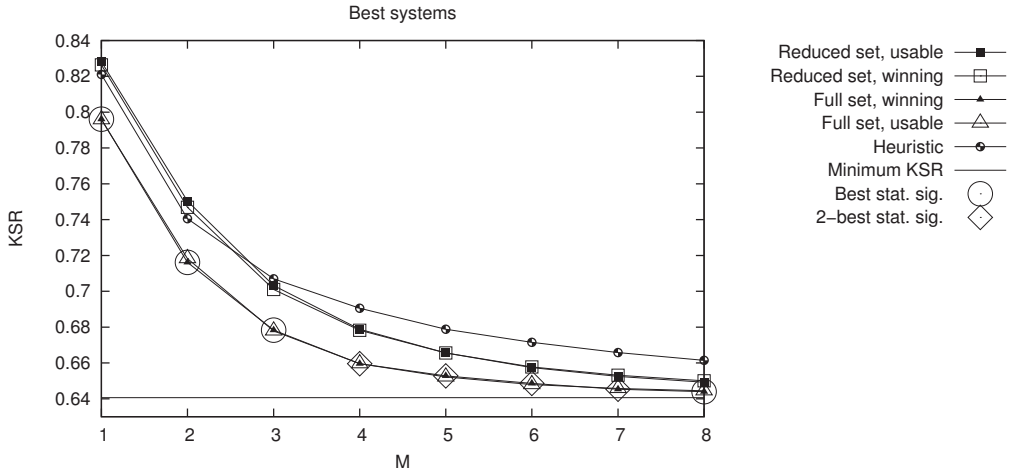



Figure 5: KSR for the best multilayer perceptron hyperparameter configuration for each training set. The best models perform similarly regardless if predicting *winning* or *usable suggestions*, even though the MSE is higher for the ones predicting usable suggestions. Best statistical significance marks those values of the maximum number of suggestions $M$ where the model using the *Full feature set with winning suggestions* is statistically significantly better than the rest; 2-best statistical significance marks those where it is not significantly better or worse than the model using the *Full feature set with usable suggestions*, but both are statistically significantly better than the rest.

## 5 Conclusions and future work

Resource-agnostic ITP is a low-cost approach for computer-aided translation. We aim to provide a competitive alternative to postediting that can easily integrate any bilingual resource the user has access to.

The naïve distance-based ranking described in (Pérez-Ortiz et al., 2014) provided results comparable to glass-box ITP. We managed to significantly improve it employing a sounder machine-learned model that manages to obtain keystroke savings in the range of 25–45% when offering up to 4 suggestions to the user.

It is important to evaluate this model with more language pairs, specially those which present special interest deemed their grammatical or lexical differences. Also, selecting which features are more representative can further improve the performance of the models while reducing the processing power needed to train and test the model.

There are other ITP systems being currently developed, namely Thot. (Ortiz-Martínez and Casacuberta, 2014) Although it is addressing a different problem by using an ad-hoc SMT system, it assists the user in a similar way. A comparison of the performance achieved will be conducted to contextualize our method.

For further improving the results attained, we plan to use a distortion model as the one proposed by Al-Onaizan and Papineni (2006), that can be used to predict which source words will be translated next, integrating this information as a feature for the multilayer perceptron, hopefully complementing our light alignment model described in Section 3.3. We also plan to use a simplified language model to give a rough estimate of the likelihood of the concatenation of $\hat{T}$ and a given suggestion.

We also plan to explore the impact of simultaneously using different black-box bilingual resources. Different strategies will be evaluated in order to integrate the available resources: combining multiple black-box MT systems as described in (Jayaraman and Lavie, 2005); using confidence-based measures in order to select the most promising translations as performed by Blatz et al. (2004); predicting the best candidates for the translation of each particular sub-segment by using only source-language information, thus avoiding the need to consult every available resource, as explored by Sánchez-Martínez (2011); or letting the multilayer perceptron manage the different suggestions, devising new features if needed.

Finally, we also aim to find evaluation metrics that improve the correlation with professionals by mimicking how a professional translator would work with the tool. Currently, we only choose suggestions that perfectly fit what the professionals are going to type; a translator could however accept a partially matching suggestion and then replace the mismatching part, or even accept a suggestion that does not match the planned translation, adapting it to the new prefix.

**Acknowledgments** This work has been partially funded by Generalitat Valenciana through grant ACIF/2014/365 from VALi+d programme.

## References

Al-Onaizan, Y. and Papineni, K. (2006). Distortion models for statistical machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, pages 529–536.

Alabau, V., González-Rubio, J., Ortíz-Martínez, D., Sanchis-Trilles, G., Casacuberta, F., García-Martínez, M., Mesa-Lao, B., Petersen, D. C., Dragsted, B., and Carl, M. (2014). Integrating online and active learning in a computer-assisted translation workbench. In *Proceedings of the First Workshop on Interactive and Adaptive Statistical Machine Translation, page to appear*, pages 1–8.

Barrachina, S., Bender, O., Casacuberta, F., Civera, J., Cubel, E., Khadivi, S., Lagarda, A., Ney, H.,

Tomás, J., Vidal, E., and Vilar, J.-M. (2009). Statistical approaches to computer-assisted translation. *Computational Linguistics*, 35(1):3–28.

Blatz, J., Fitzgerald, E., Foster, G., Gandrabur, S., Goutte, C., Kulesza, A., Sanchis, A., and Ueffing, N. (2004). Confidence estimation for machine translation. In *Proceedings of the 20th International Conference on Computational Linguistics*, COLING '04, pages 315–321, Stroudsburg, PA, USA. Association for Computational Linguistics.

Domingo, M., Peris, A., and Casacuberta, F. (2016). Interactive-predictive translation based on multiple word-segments. *Baltic Journal of Modern Computing*, 4(2):282–291.

Duda, R. O., Hart, P. E., and Stork, D. G. (2000). *Pattern Classification*. John Wiley and Sons Inc., second edition.

Esplà-Gomis, M., Sánchez-Martínez, F., and Forcada, M. L. (2012). A simple approach to use bilingual information sources for word alignment. *Procesamiento del Lenguaje Natural*, 49:93–100.

Foster, G. F., Isabelle, P., and Plamondon, P. (1997). Target-text mediated interactive machine translation. *Machine Translation*, 12(1-2):175–194.

Gori, M. and Tesi, A. (1992). On the problem of local minima in backpropagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(1):76–86.

Green, S., Chuang, J., Heer, J., and Manning, C. D. (2014). Predictive translation memory: A mixed-initiative system for human language translation. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*, pages 177–187.

Hutchins, W. J. and Somers, H. L. (1992). *An introduction to machine translation*. Academic Press.

Jayaraman, S. and Lavie, A. (2005). Multi-engine machine translation guided by explicit word matching. In *Proceedings of the ACL 2005 on Interactive poster and demonstration sessions*, pages 101–104.

Koehn, P. (2004). Statistical significance tests for machine translation evaluation. In *Conference on Empirical Methods on Natural Language Processing*, pages 388–395.

Koehn, P. (2005). Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pages 79–86.

Koehn, P. (2009). A web-based interactive computer aided translation tool. In *Proceedings of the ACL-IJCNLP 2009 Software Demonstrations*, pages 17–20.

Koehn, P., Alabau, V., Carl, M., Casacuberta, F., García-Martínez, M., González-Rubio, J., Keller, F., Ortiz-Martínez, D., Sanchis-Trilles, G., Bonk, U. G. R., and and, C. B. (2015). CASMACAT: Final public report. `http://www.casmacat.eu/uploads/Deliverables/final-public-report.pdf`.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., et al. (2007). Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL: interactive poster and demonstration sessions*, pages 177–180.

Langlais, P., Sauvé, S., Foster, G., Macklovitch, E., and Lapalme, G. (2000). Evaluation of TransType, a computer-aided translation typing system: a comparison of a theoretical-and a user-oriented evaluation procedures. In *Conference on Language Resources and Evaluation (LREC)*.

Macklovitch, E. (2006). TransType2: The last word. In *Proceedings of the 5th International Conference on Languages Resources and Evaluation (LREC 06)*, pages 167–172.

Nissen, S. (2003). Implementation of a fast artificial neural network library (FANN). Technical report, Department of Computer Science University of Copenhagen (DIKU). `http://fann.sf.net`.

Ortiz-Martínez, D. and Casacuberta, F. (2014). The new thot toolkit for fully automatic and interactive statistical machine translation. In *Proc. of the European Association for Computational Linguistics (EACL): System Demonstrations*, pages 45–48, Gothenburg, Sweden.

Pérez-Ortiz, J. A., Torregrosa, D., and Forcada, M. L. (2014). Black-box integration of heterogeneous bilingual resources into an interactive translation system. *EACL 2014 Workshop on Humans and Computer-assisted Translation*, pages 57–65.

Sánchez-Martínez, F. (2011). Choosing the best machine translation system to translate a sentence by using only source-language information. In *Proceedings of the 15th Annual Conference of the European Associtation for Machine Translation*, pages 97–104.

Somers, H. L. (2003). *Computers and Translation: A Translator's Guide*. Benjamins translation library. John Benjamins Publishing Company.

Torregrosa, D., Forcada, M. L., and Pérez-Ortiz, J. A. (2014). An open-source web-based tool for resource-agnostic interactive translation prediction. *The Prague Bulletin of Mathematical Linguistics*, 102(1):69–80.

# Multi-domain Adaptation for Statistical Machine Translation Based on Feature Augmentation

**Kenji Imamura** kenji.imamura@nict.go.jp
**Eiichiro Sumita** eiichiro.sumita@nict.go.jp
National Institute of Information and Communications Technology,
3-5 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-0289, Japan

**Abstract**

Domain adaptation is a major challenge when applying machine translation to practical tasks. In this paper, we present domain adaptation methods for machine translation that assume multiple domains. The proposed methods combine two model types: a corpus-concatenated model covering multiple domains and single-domain models that are accurate but sparse in specific domains. We combine the advantages of both models using feature augmentation for domain adaptation in machine learning.

Our experimental results show that the BLEU scores of the proposed method clearly surpass those of single-domain models for low-resource domains. For high-resource domains, the scores of the proposed method were superior to those of both single-domain and corpus-concatenated models. Even in domains having a million bilingual sentences, the translation quality was at least preserved and even improved in some domains. These results demonstrate that state-of-the-art domain adaptation can be realized with appropriate settings, even when using standard log-linear models.

## 1 Introduction

Machine translation is used for translating a variety of text types, including speech. However, it remains challenging to appropriately translate texts across all domains and only a limited number of domains have been targeted.

The most promising approach to improve translation quality is to train the translator on massive bilingual corpora. However, collecting such corpora is challenging and expensive in several domains. Domain adaptation, which improves target domain quality by using data from another domain, has been proposed as a solution (Foster and Kuhn, 2007; Foster et al., 2010; Axelrod et al., 2011; Bisazza et al., 2011; Sennrich, 2012; Sennrich et al., 2013). This technique is important when applying machine translation to practical tasks.

This paper presents methods of domain adaptation for statistical machine translation (SMT) that assume multiple domains. The proposed methods combine multiple models using log-linear interpolation. These are simple yet effective approaches to take advantage of multiple domains based on feature augmentation (Daumé, 2007), a domain adaptation technique used in machine learning. We propose the following two methods.

1. Simultaneous optimization of multiple domains: this method uses an optimizer extended to multiple domains to optimize an augmented feature space.
2. Optimization of one domain at a time: this method restricts the feature space and regards

this space as that used in the standard log-linear model. This can be realized via a slight modification of existing translation systems.

Both methods use a corpus-concatenated model, which covers multiple domains and contains few unknown words, and single-domain models, which are accurate in their specific domains. In addition, we tune the hyper-parameter of the multiple-model combination. With appropriate settings, state-of-the-art domain adaptation can be realized even when using standard log-linear models.

In this study, we use phrase-based statistical machine translation (PBSMT) (Koehn et al., 2003, 2007) with preordering. The remainder of this paper is organized as follows. Section 2 briefly reviews domain adaptation in machine translation. Section 3 explains our proposed methods in detail. Section 4 discusses the characteristics of our methods through experiments, and Section 5 concludes the paper.

## 2 Domain Adaptation for Statistical Machine Translation

Domain adaptation is applied when the target domain (in-domain) data are insufficient but data from another domain (out-domain) are available in sufficient quantities. Domain adaptation in machine translation aims to improve the translation quality of in-domain texts using both in-domain and out-domain data.

There are two types of domains: those that are predefined, such as "News" and "Web," and those that are artificially created via automatic clustering. Even when using automatic clustering, the translation quality can be improved in some cases (Finch and Sumita, 2008; Sennrich et al., 2013). However, in this study, we have used predefined domains.

**Corpus Concatenation** The simplest approach to achieving domain adaptation for SMT is training the model using a concatenated corpus of in- and out-domain data. We refer to this method as corpus concatenation. The trained model is optimized using development (held-out) data of the in-domain.

In machine learning, a model trained on a concatenated corpus has features that are intermediate between the in- and the out-domains. Therefore, model accuracy is also generally intermediate between models trained individually on the in-domain or the out-domain data (i.e., single-domain models).

In contrast, for machine translation, translation quality achieved with corpus concatenation may be superior to that achieved with a single-domain model because the vocabulary coverage increases. The improvement represents a trade-off between reduction in the number of unknown words and greater inaccuracy of model parameters.

**Linear/Log-linear Interpolation** Statistical machine translation computes translation likelihood using linear or log-linear interpolation of feature values obtained from submodels such as phrase tables, language models, and lexicalized reordering models. The overall likelihood is computed by the following equation:

$$\log P(e|f) \quad \propto \quad \mathbf{w} \cdot \mathbf{h}(e, f) \tag{1}$$

where $\mathbf{h}(e, f)$ is a feature vector and $\mathbf{w}$ is a weight vector of the feature functions.

Then, a domain-specific translation is generated by changing the weight vector $\mathbf{w}$ of each domain. For example, Foster and Kuhn (2007) trained single-domain PBSMT models and translated them while changing the weight vectors of the linear and log-linear interpolations. Although they used perplexities as objective functions to estimate the weights, optimization algorithms, such as minimum error rate training (MERT) (Och, 2003), have been used recently to estimate weight vectors (Foster et al., 2010).

Feature augmentation (Daumé, 2007) is a domain adaptation method used in machine learning that simultaneously optimizes the weight vector of each domain (cf., Section 3.1). Clark et al. (2012) applied it to machine translation as a type of log-linear interpolation; however, they only adapted the weight vectors of a model.

**Model Adaptation** There are basically two approaches to achieve domain adaptation by changing the feature vector $\mathbf{h}(e, f)$. The first is model adaptation, which modifies trained submodels, and the second is corpus filtering, which trains models using adapted corpora. The fill-up method (Bisazza et al., 2011), translation model combination (Sennrich, 2012), and instance weighting (Foster et al., 2010; Matsoukas et al., 2009) are well-known model adaptation methods.

The fill-up method changes feature values. If a phrase is contained in an in-domain phrase table, the feature values in that table are used. Otherwise, the feature values in the out-domain phrase table are used.

Translation model combination generates a new phrase table by combining two translation probabilities of in- and out- domains. The weights of the combination are determined using each feature function to minimize the perplexity on a development set.

Instance weighting modifies each model parameter in the phrase table to discriminate between the in- and the out-domains by additional learning.

These methods reduce the number of unknown words because the candidates for phrase translation are also altered when the phrase tables are modified. However, submodels other than the phrase table must be adapted using other methods.

**Corpus Filtering** The other approach to changing the feature vector $\mathbf{h}(e, f)$ is to train the models using the adapted corpora. Although corpus concatenation is one such approach, it uses all sentences in the out-domain corpora. Training data should be selected for better adaptation. Axelrod et al. (2011) selected training sentences similar to those in the in-domain from the out-domain corpora on the basis of cross-entropy difference (i.e., modified Moore-Lewis filtering). Then, they trained the models using the in-domain corpus with additional sentences.

Corpus filtering adapts not only phrase tables but also all submodels used in the translator. However, the ideal number of additional sentences cannot be estimated in advance.

**Another Approach** Another approach that does not require changing the likelihoods is connecting two translators in series. A translation result generated by the out-domain translator is re-translated by the in-domain translator (Jeblee et al., 2014). This method treats the generation of domain-specific translation as error correction.

## 3 Multi-domain Adaptation

### 3.1 Feature Augmentation

Feature augmentation is used to adapt feature weights to domains in machine learning. The feature space is segmented into the following subspaces: common, out-domain (source domain), and in-domain (target domain). In-domain features are copied to the in-domain and common spaces, and out-domain features are copied to the out-domain and common spaces. The adapted weight vector is obtained by optimizing the entire space. The in- and out-domain features deployed in the common space complement each other to improve likelihood accuracy.

Although feature augmentation is mainly used to adapt out-domain models to the in-domain, it can be easily extended to $D$ domains because it treats the in- and the out-domains equivalently. In this case, the feature space is segmented into $D + 1$ subspaces: common, domain 1 , ..., and domain $D$ (Figure 1), which is expressed as follows:

$$\mathbf{h}(f, e) \quad = \quad \langle \mathbf{h}_c, \mathbf{h}_1, \ldots, \mathbf{h}_i, \ldots, \mathbf{h}_D \rangle \tag{2}$$

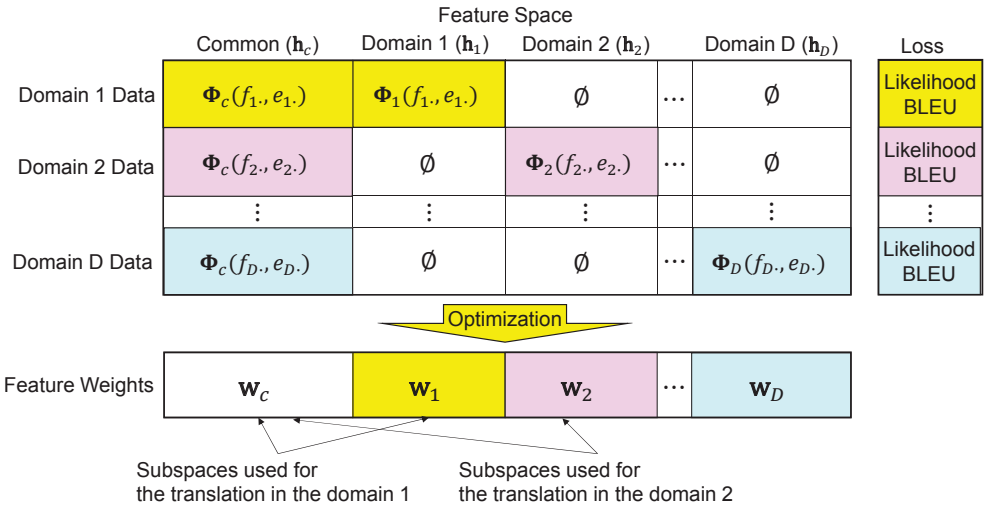



Figure 1: Feature Augmentation Incorporating Corpus-Concatenated Model and Single-Domain Models

where $\mathbf{h}_c$ and $\mathbf{h}_i$ denote the feature vectors of the common and the domain-specific spaces, respectively. All features are deployed to the common space, but only features that match the domain are copied to the domain space.

$$\mathbf{h}_c = \mathbf{\Phi}(f, e) \tag{3}$$

$$\mathbf{h}_i = \begin{cases} \mathbf{\Phi}(f, e) & \text{if domain}(f) = i \\ \emptyset & \text{otherwise} \end{cases} \tag{4}$$

where $\mathbf{\Phi}(f, e)$ denotes the subvector that stores the model scores and so on. It is equal to $\mathbf{h}(f, e)$ if no feature augmentation is applied. We obtain the weight vector by optimizing this feature matrix.

We use the default features of the Moses toolkit (Koehn et al., 2007) (15 dimensions) in the experiments reported in Section 4. The number of dimensions in the augmented feature space is 15 in the common space and 14 in each of the domain spaces[1].

Clark et al. (2012) applied feature augmentation to machine translation from Arabic to English (with News and Web domains) and Czech to English (six domains, e.g., Fiction). Only a corpus-concatenated model was used to obtain features so that feature functions were not changed to reflect the different domains.

### 3.2 Core of Proposed Methods

#### 3.2.1 Corpus-Concatenated Model and Single-domain Models

In machine translation, compared to feature weights, feature functions have a greater effect on translation quality. Therefore, it is natural to change the submodels depending on the space. Similar to the feature deployment, we assign the corpus-concatenated model, which is constructed from all domain data, to the common space and the single-domain models, which are constructed from one domain data, to the domain-specific spaces. Our approach is as follows.

[1] `UnkPenalty`, which indicates the number of unknown words, is only deployed to the common space because it is not tunable.

- For all submodels, the corpus-concatenated model and single-domain models are constructed in advance.
- In feature augmentation, the scores obtained from the corpus-concatenated model are deployed to the common space as the feature function values, while those from the single-domain models are deployed to the domain spaces (Figure 1). Equations 3 and 4 are then rewritten as follows:

$$\mathbf{h}_c = \mathbf{\Phi}_c(f, e) \tag{5}$$

$$\mathbf{h}_i = \begin{cases} \mathbf{\Phi}_i(f, e) & \text{if domain}(f) = i \\ \emptyset & \text{otherwise} \end{cases} \tag{6}$$

  where $\mathbf{\Phi}_c(f, e)$ and $\mathbf{\Phi}_i(f, e)$ denote feature vectors obtained from the corpus-concatenated model and single-domain model $i$, respectively.
- While decoding, phrase pairs are first retrieved from both the corpus-concatenated and single-domain phrase tables. The likelihood of each translation hypothesis is computed using only the common space and domain space of the input sentence.

Use of the corpus-concatenated phrase table reduces the number of unknown words because phrase pairs appearing in other domains can be used to generate hypotheses. In addition, precise values of the feature functions can be obtained if the hypotheses exist in the single-domain models. All submodels used in the translator can be adapted without considering their types (i.e., phrase tables, reordering models, and language models) because adaptation is achieved by optimizing the augmented feature space. Therefore, this method can be easily applied to other translation methods, such as tree-to-tree translation. Moreover, unlike corpus filtering, this method does not need to consider the optimal number of additional data entries.

Note that, in machine translation, language models are sometimes constructed from very large monolingual corpora. Such models are regarded as corpus-concatenated models that cover broad domains. In this case, i.e., when we add models (feature functions) acquired from external knowledge, they are located in the common space, which increases the number of dimensions.

#### 3.2.2 Empty Value

In our method, several phrases appear only in one of the phrase tables of the corpus-concatenated and single-domain models. The feature functions are expected to return appropriate values for these phrases. We refer to these as empty values. Even though an empty value is a type of unknown probability and should be computed from the probability distribution of the phrases, we treat it as a hyper-parameter. In other words, empty values are set experimentally to maximize the BLEU score of a development corpus[2].

### 3.3 Optimization

#### 3.3.1 Joint Optimization

One merit of feature augmentation in machine learning is that conventional algorithms can be used for optimization because feature augmentation operates only in the feature space.

Machine translation uses optimization algorithms such as MERT (Och, 2003), pairwise ranking optimization (PRO) (Hopkins and May, 2011), and $k$-best batch MIRA (KBMIRA) (Cherry and Foster, 2012). We employ KBMIRA in this paper because it is appropriate for high-dimensional optimization[3].

[2] Moses assigns -100 as the empty value (Koehn and Schroeder, 2007; Birch et al., 2007). As we describe in Section 4.2, this is extremely small and produces low BLEU scores.

[3] Another reason is that the BLEU score of a baseline system was the highest in our preliminary experiments.

A major difference between general machine learning and optimization in machine translation is in the loss functions. The loss functions of machine learning algorithms use likelihood output by decoders. In contrast, the optimization algorithms employed in machine translation use both likelihood and automatic evaluation scores, such as BLEU (Papineni et al., 2002). Automatic evaluation scores are computed by comparing system outputs with their reference translations over *the entire document*. In fact, MERT and KBMIRA contain BLEU scores of the development set in their loss functions[4]. This means that BLEU scores must be computed for each domain to optimize multiple domains.

To solve this problem, we modify the KBMIRA algorithm. The modifications to Algorithm 1 proposed by Cherry and Foster (2012) are as follows.

1. The variable $BG$ that maintains BLEU statistics (such as the number of n-gram matches) is extended to the $D$-dimensional array, where $D$ denotes the number of domains.
2. The BLEU score of each translation is computed from $BG_i$, where $i$ is the domain of the input sentence.
3. After the weights are updated, the BLEU statistics of the best translation are added to $BG_i$.

These modifications optimize the feature weights of each domain space to the development set of each domain.

#### 3.3.2 Independent Optimization

Joint optimization is sometimes redundant because it optimizes all domains even when only one domain is to be adapted. To solve this problem, we restrict and optimize the feature space only to subspaces related to the domain that we want to adapt. We refer to this as independent optimization.

Independent optimization restricts the feature space to the common and the domain $i$ spaces, and only the tuning data of domain $i$ are used for optimization. Namely, Equation 2 is replaced with Equation 7.

$$\mathbf{h}(f,e) = \langle \mathbf{h}_c, \mathbf{h}_i \rangle \quad (7)$$

$$\mathbf{h}_c = \mathbf{\Phi}_c(f,e) \quad (8)$$

$$\mathbf{h}_i = \mathbf{\Phi}_i(f,e) \quad (9)$$

This is the same as a standard log-linear model, which can be optimized without joint optimization by using existing optimizers. Furthermore, we can use multiple decoders with slight modifications because they only have to allow for 1) multiple models to be jointly used and 2) setting the empty value.

The common space might not be strictly optimized compared to joint optimization. However, in machine translation, feature functions affect translation quality to a greater extent than feature weights. Therefore, we assume that, in practice, partially rough optimization is not problematic. Note that the following two points are common to joint optimization.

1. Features of the common space are obtained from the corpus-concatenated model.
2. An empty value is set appropriately.

## 4 Experiments

### 4.1 Experimental Settings

**Domains/Corpora** Four domains were used in this paper. The language pairs were English-to-Japanese (En-Ja) and Japanese-to-English (Ja-En). The size of the corpora of each domain

[4]PRO, which was used by Clark et al. (2012) for feature augmentation, employs BLEU scores approximated by sentences.

| Domain | # of Sentences | | | # of Training Words | |
|---|---|---|---|---|---|
| | Training | Dev. | Test | En | Ja |
| MED | 222,945 | 1,000 | 1,000 | 3.1M | 3.3M |
| LIVING | 986,946 | 1,800 | 1,800 | 14.3M | 16.5M |
| NTCIR | 1,387,713 | 2,000 | 2,000 | 48.7M | 52.3M |
| ASPEC | 1,000,000 | 1,790 | 1,784 | 25.9M | 28.7M |

Table 1: Corpus Statistics

is listed in Table 1. The MED corpus is relatively small, whereas the other corpora comprise nearly a million sentences each. Sentences with fewer than 80 words were used for training.

- **MED**: A pseudo-dialogue corpus between patients and medical doctors (or staff) in hospitals. This was developed in-house.
- **LIVING**: A pseudo-dialogue corpus wherein visitors (or residents) from foreign countries talk to local people. This was also developed in-house.
- **NTCIR**: A patent corpus. The training and development sets were provided by the international conference NTCIR-8, and the test set was provided by NTCIR-9[5].
- **ASPEC**: An Asian scientific paper excerpt corpus (Nakazawa et al., 2016)[6]. We used a million sentences of high-confidence translation from ASPEC-JE.

**Translation System** Each source sentence was preordered using an in-house preordering system (Section 4.5 of Goto et al. (2015)) trained for general-purpose. The same preordering system was applied to all domains. In addition, all Japanese sentences, including the test sets, were segmented into words in advance using the MeCab morphological analyzer (Kudo et al., 2004).

The phrase tables and lexicalized reordering models were trained using the default settings in the Moses toolkit. The 5-gram language models were learned from the target side of the training sentences using KenLM (Heafield et al., 2013). Multi-domain KBMIRA, described in Section 3.3.1, was used for optimization.

A clone of the Moses decoder was used for decoding. The settings were the same as the default values in Moses, i.e., `phrase_table_limit` $= 20$, `distortion_limit` $= 6$, and the beam width was 200. When the decoder selected phrase pair candidates, 1) phrase pairs were first obtained from all phrase tables, 2) a likelihood of each phrase was computed in accordance with the augmented feature space, and 3) the highest 20 pairs were selected.

**Empty Value** The empty value described in Section 3.2.1 was set empirically. From integer values of -3 to -20, we selected the empty value that achieved the highest BLEU score in the set in which all development sets were concatenated. The resulting empty values were -7 for En-Ja translation and -6 for Ja-En translation. If we treat these values as probabilities, they are $\exp(-7) \approx 0.0009$ and $\exp(-6) \approx 0.0025$, respectively.

**Evaluation Metrics** We used word BLEU, the translation edit rate (TER) (Snover et al., 2006), Meteor (English only) (Denkowski and Lavie, 2014), and the rank-based intuitive bilingual evaluation score (RIBES; Japanese only) (Isozaki et al., 2010) as the evaluation metrics.

[5] `http://research.nii.ac.jp/ntcir/index-ja.html`
[6] `http://lotus.kuee.kyoto-u.ac.jp/ASPEC/`

The MultEval tool (Clark et al., 2011)[7] was used for statistical testing[8] with the significance level set to $p < 0.05$. The mean scores of five runs were used to reduce instability in optimization. Although we used multiple metrics, for simplicity we will describe the results using BLEU.

**Comparison Methods** We compared various methods using the single-domain model as the baseline. We used the following conventional methods, which have been described in Section 2.

- **Corpus Concatenation**: A corpus-concatenated model was constructed using all domain data. Optimization and testing were performed using the development and test sets of each domain.
- **Feature Augmentation (Clark)**: Feature augmentation was applied to the adaptation; however, all features of the common and domain-specific spaces were obtained from the corpus-concatenated model. This is the same setting as that used by Clark et al. (2012), except that we used multi-domain KBMIRA for optimization.
- **Fill-Up**: The fill-up method (Bisazza et al., 2011) was used for domain adaptation.
- **TM Combination**: Translation model combination (Sennrich, 2012) was applied for adaptation. We used the `tmcombine` program in the Moses toolkit.
- **Corpus Filtering**: The modified Moore-Lewis filtering scheme proposed by Axelrod et al. (2011) was applied. All corpora, except for the target domain, were used as out-domain corpora. The number of additional sentences was determined to maximize the BLEU score of the development set.

As variations of the proposed methods, we tested the following settings.

- **Proposed (Joint)**: The best setting of the proposed method using joint optimization (cf., Section 3.3.1).
- **Proposed (Independent)**: The best setting of the proposed method using independent optimization (cf., Section 3.3.2).
- **Proposed (empty = -100)**: The empty value was set to -100, which is equivalent to the Moses value. We used independent optimization in this setting; however, the same tendency was observed when using joint optimization.
- **Proposed (Out-Domain)**: The common space model was changed from the corpus-concatenated model to the out-domain model learned from the other three domain corpora. In addition, independent optimization was used.

### 4.2 Translation Quality

Tables 2 and 3 show the BLEU scores of the abovementioned methods for En-Ja and Ja-En translations, respectively. Bold values represent the highest scores. The symbols (+) and (-) denote whether the score was significantly improved or degraded compared to that of the single-domain model ($p < 0.05$).

Because the domains used in the experiments are not closely related, many BLEU scores were degraded by conventional adaptations. For instance, the corpus concatenation approach

[7] `https://github.com/jhclark/multeval`.

[8] We incorporated the RIBES script (`http://www.kecl.ntt.co.jp/icl/lirg/ribes/index.html`) into the MultEval tool.

| Method | Domain | | | | | |
|---|---|---|---|---|---|---|
| | MED | | | LIVING | | |
| | BLEU | TER | RIBES | BLEU | TER | RIBES |
| Single Domain Model | 23.23 | 62.46 | 78.34 | **24.56** | 61.33 | 78.78 |
| Corpus Concatenation | 22.65(-) | 64.08(-) | 77.53(-) | 22.99(-) | 63.41(-) | 77.52(-) |
| Feature Augmentation (Clark) | 22.49(-) | 63.78(-) | 77.70(-) | 22.97(-) | 63.40(-) | 77.41(-) |
| Fill-Up | 22.42(-) | 63.63(-) | 77.46(-) | 23.38(-) | 63.16(-) | 77.80(-) |
| TM Combination | 23.81(+) | 61.94(+) | 78.37 | 24.05(-) | 62.05(-) | 78.62(-) |
| Corpus Filtering | **24.02**(+) | **61.61**(+) | 78.43 | 24.50 | 62.08(-) | 78.51(-) |
| Proposed (Joint) | 23.69(+) | 61.79(+) | **78.67**(+) | 24.43 | 61.51 | 78.72 |
| Proposed (Independent) | 23.75(+) | 61.78(+) | 78.56 | 24.43 | **61.09**(+) | **78.80** |
| Proposed (empty=-100) | 23.66(+) | 62.37 | 78.12 | 23.91(-) | 61.84(-) | 78.39(-) |
| Proposed (Out-Domain) | 23.79(+) | 62.19 | 78.46 | 24.29(-) | 61.77(-) | 78.75 |

| Method | Domain | | | | | |
|---|---|---|---|---|---|---|
| | NTCIR | | | ASPEC | | |
| | BLEU | TER | RIBES | BLEU | TER | RIBES |
| Single Domain Model | 38.62 | 47.85 | 80.16 | 32.69 | 54.12 | 78.16 |
| Corpus Concatenation | 38.09(-) | 48.54(-) | 79.88(-) | 30.59(-) | 55.95(-) | 77.22(-) |
| Feature Augmentation (Clark) | 38.09(-) | 48.54(-) | 79.90(-) | 30.65(-) | 55.91(-) | 77.28(-) |
| Fill-Up | 38.37(-) | 48.12(-) | 80.03(-) | 31.50(-) | 55.06(-) | 77.62(-) |
| TM Combination | 38.32(-) | 48.11(-) | 80.09(-) | 31.97(-) | 54.77(-) | 77.84(-) |
| Corpus Filtering | 38.77(+) | 47.82 | 80.17 | 32.57(-) | 54.14 | 78.14 |
| Proposed (Joint) | 38.72(+) | 47.77 | 80.22(+) | 32.69 | 54.20 | 78.10 |
| Proposed (Independent) | **38.83**(+) | **47.64**(+) | 80.22 | **32.76** | 54.13 | 78.15 |
| Proposed (empty=-100) | 38.56 | 47.90 | 80.10(-) | 32.62 | 54.17 | 78.13 |
| Proposed (Out-Domain) | 38.65 | 47.78 | **80.28**(+) | 32.72 | **54.02** | **78.19** |

Table 2: Automatic Evaluation Scores of Various Methods (En-Ja Translation)

and feature augmentation (Clark) were degraded for most domains. This is because the corpus-concatenated models are an average of all the domain models and the precision of the model parameters was degraded for each specific domain.

The BLEU scores of Fill-Up were superior to those of corpus concatenation in several cases but inferior to those of the single-domain models. The TM Combination scores were both improved and degraded depending on the domain, and we could not confirm the effects. In corpus filtering, the scores were improved or were at the same level compared to the single-domain model, except for En-Ja translation of the ASPEC domain. Although only 100k sentences were added in the ASPEC En-Ja domain, they affected translation quality. This shows that corpus filtering is effective; however, determining the ideal number of additional sentences is difficult.

Conversely, all BLEU scores of the proposed methods (joint and independent) were significantly improved or were at the same level as those of the single-domain models. The independent optimization scores tended to be better than those of joint optimization. When the empty value was set to -100, the weight vector could not be optimized because the BLEU score did not converge in some cases (N/A in Table 3). Focusing on the proposed method (out-domain), the scores were degraded from those of the proposed methods (joint and independent) in most cases. This indicates that the corpus-concatenated model is better than the out-domain model for the common space.

| Method | Domain | | | | | |
|---|---|---|---|---|---|---|
| | MED | | | LIVING | | |
| | BLEU | TER | Meteor | BLEU | TER | Meteor |
| Single Domain Model | 17.38 | 71.14 | 25.49 | 19.71 | 67.08 | 27.58 |
| Corpus Concatenation | 17.07 | 70.72 | 25.26(-) | 18.80(-) | 68.50(-) | 26.74(-) |
| Feature Augmentation (Clark) | 16.75(-) | 71.39 | 25.13(-) | 18.95(-) | 68.31(-) | 26.75(-) |
| Fill-Up | 16.56(-) | 71.98(-) | 25.11(-) | 19.06(-) | 68.45(-) | 26.65(-) |
| TM Combination | 17.55 | 71.17 | 25.65(+) | 19.99(+) | 67.22 | 27.31(-) |
| Corpus Filtering | 18.14(+) | 70.14(+) | **26.16**(+) | 19.76 | 66.81(+) | 27.48(-) |
| Proposed (Joint) | 18.14(+) | **69.29**(+) | 25.90(+) | 20.16(+) | **66.61**(+) | 27.45(-) |
| Proposed (Independent) | **18.43**(+) | 69.85(+) | 26.00(+) | **20.17**(+) | 66.94 | 27.53 |
| Proposed (empty=-100) | 17.13 | 71.22 | 25.78(+) | 19.86 | 67.24 | 27.67(+) |
| Proposed (Out-Domain) | 17.32 | 70.93 | 25.34 | 19.66 | 67.31(-) | 27.02(-) |

| Method | Domain | | | | | |
|---|---|---|---|---|---|---|
| | NTCIR | | | ASPEC | | |
| | BLEU | TER | Meteor | BLEU | TER | Meteor |
| Single Domain Model | 33.63 | 52.67 | 35.68 | 21.75 | 64.95 | 31.01 |
| Corpus Concatenation | 33.21(-) | 52.94(-) | 35.33(-) | 20.41(-) | 66.00(-) | 30.36(-) |
| Feature Augmentation (Clark) | 33.24(-) | 53.00(-) | 35.38(-) | 20.39(-) | 66.18(-) | 30.33(-) |
| Fill-Up | 33.14(-) | 53.06(-) | 35.48(-) | 20.98(-) | 65.41(-) | 30.58(-) |
| TM Combination | 33.32(-) | 52.78(-) | 35.54(-) | 21.16(-) | 65.17(-) | 30.77(-) |
| Corpus Filtering | **33.73** | 52.45(+) | **35.71** | 21.72 | 64.71(+) | 31.03(+) |
| Proposed (Joint) | 33.68 | 52.42(+) | 35.70 | 21.75 | 64.79(+) | **31.20**(+) |
| Proposed (Independent) | 33.70 | **52.33**(+) | 35.67 | **21.81** | 64.76(+) | 31.19(+) |
| Proposed (empty=-100) | N/A | N/A | N/A | N/A | N/A | N/A |
| Proposed (Out-Domain) | 33.52(-) | 52.70 | 35.62 | 21.73 | **64.72**(+) | 31.06 |

Table 3: Automatic Evaluation Scores of Various Methods (Ja-En Translation)

In summary, compared to the other methods, the proposed methods achieved the best translation quality. Therefore, state-of-the-art domain adaptation can be realized with the appropriate settings even when using a standard log-linear model such as independent optimization.

### 4.3 Effects as Single-Domain Adaptation

A typical situation wherein domain adaptation is needed would be one in which sufficient training data cannot be collected and new domain data must be translated. In this section, we investigate translation quality when changing training corpus size, focusing only on the En-Ja translation in the MED domain. Note that the other domains are not changed.

Table 4 compares the results obtained using the single-domain model, corpus concatenation, and the proposed method (independent). The symbols (+) and (-) denote scores that were significantly improved or degraded compared to those of the single-domain model. The symbol (†) denotes a score that was significantly improved compared to that of corpus concatenation.

When using one thousand training sentences (1k), the score achieved by the proposed method was significantly higher than that achieved by the single-domain model and equal to that achieved by corpus concatenation. When the size of the training corpus was increased, BLEU scores increased for all methods. However, the improvement in the corpus concatenation score was less than that in the single-domain model score. The single-domain model surpassed corpus

| # of Sentences | Single Domain Model | Corpus Concatenation | Proposed (Independent) |
|---|---|---|---|
| 1k | 6.42 | 17.51 (+) | **17.59** (+) |
| 3k | 8.99 | 17.52 (+) | **17.95** (+)(†) |
| 10k | 12.54 | 18.19 (+) | **19.02** (+)(†) |
| 30k | 16.49 | 19.18 (+) | **20.28** (+)(†) |
| 100k | 20.63 | 20.92 | **22.53** (+)(†) |
| 223k (All) | 23.23 | 22.65 (-) | **23.75** (+)(†) |

Table 4: BLEU Scores for Different Training Sizes (MED Domain, En-Ja Translation)

concatenation when more than 100 thousand (100k) training sentences were used. BLEU scores for the proposed method exceeded those of the single-domain model and corpus concatenation when more than three thousand (3k) training sentences were used. These results demonstrate that the proposed method successfully integrated the merits of the single-domain and corpus-concatenated models.

Here, we refer back to Tables 2 and 3. From these tables, it can be seen that the BLEU scores of the proposed method (joint and independent) in the MED domain improved for both En-Ja and Ja-En. We assume that it was possible to improve the translation quality because the MED corpus was relatively small. In contrast, the translation quality was not necessarily improved in other domains because these were trained using approximately a million sentences. However, it should be noted that the translation quality was not degraded and, in some cases, it was improved, even when the proposed method was applied to domains of very large corpora. The proposed method is, therefore, robust to corpus size.

### 4.4 Unknown Words

The proposed methods take the advantage of the corpus-concatenated and single-domain models. Finally, we analyze the characteristics of the proposed methods from the viewpoint of unknown words.

In this paper, we distinguish between source unknown words (source UNK) and target unknown words (target UNK) by referring to the categories suggested by Irvine et al. (2013)[9]. Source unknown words occur when the source words (or phrases) do not exist in the phrase tables. Target unknown words occur when a reference translation cannot be generated because the target words (or phrases) are not in the phrase tables even though the source words exist. This can be determined by forced decoding (Yu et al., 2013).

Table 5 shows the sentence rates that include unknown words in the En-Ja translation. Although there were differences depending on the domains, the rates of source and target UNK both decreased in corpus concatenation compared to those in the single-domain models. For example, in the MED domain, the rate of source UNK decreased from 9.1% to 1.0%, and that of target UNK decreased from 38.5% to 18.1%. This result proved that words in the other domains were used for translation. However, this result also proved that reduction of unknown words does not directly contribute to translation quality because the quality of the single-domain models was better than that of corpus concatenation. Hence, optimization is important.

The proposed methods further reduced unknown words, except for the NTCIR domain. The proposed methods use phrase tables of the corpus-concatenated and single-domain models. Even though these phrase tables were trained from corpora that partially overlap, the acquired

[9]The source unknown and target unknown words correspond to the SEEN and SENSE errors, respectively, used by Irvine et al. (2013).

| UNK Type | Method | Domain MED | LIVING | NTCIR | ASPEC |
|---|---|---|---|---|---|
| Source UNK | Single-Domain Model | 9.1% | 4.1% | 7.9% | 22.5% |
| | Corpus Concatenation | 1.0% | 2.8% | 6.5% | 21.6% |
| | Proposed (Independent) | 0.9% | 2.4% | 6.4% | 21.2% |
| Target UNK | Single-Domain Model | 38.5% | 26.4% | 21.3% | 26.5% |
| | Corpus Concatenation | 18.1% | 20.1% | 17.1% | 21.6% |
| | Proposed (Independent) | 16.1% | 17.0% | 17.2% | 20.0% |

Table 5: Sentence Rates that Include Unknown Words (En-Ja Translation)

phrases were slightly different. Hence, the coverage of the proposed methods increased.

## 5 Conclusions

In this paper, we presented multi-domain adaptation methods for statistical machine translation. The proposed methods combined the corpus-concatenated model, which has high coverage and few unknown words, with single-domain models, which ensure precision of the feature functions. To take advantage of the benefits of both models, we applied feature augmentation to machine translation. In addition, we tuned the empty value to balance the two models.

In both joint and independent optimization, the translation quality was improved or at the same level compared with the single-domain models in our experiments. The resulting BLEU scores of the proposed method clearly surpassed those of the single-domain models in low-resource domains. Even in domains with a million training sentences, the translation quality remained at least equal, and in some domains, it was improved. These results show that state-of-the-art domain adaptation can be realized with appropriate settings, even when using standard log-linear models.

The proposed methods can be easily applied to other translation strategies, such as tree-to-tree translation. We plan to apply our methods to tree-to-tree translation and evaluate the effects.

## Acknowledgments

This work was supported by "Promotion of Global Communications Plan — Research and Development and Social Demonstration of Multilingual Speech Translation Technology," a program of Ministry of Internal Affairs and Communications, Japan.

## References

Axelrod, A., He, X., and Gao, J. (2011). Domain adaptation via pseudo in-domain data selection. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 355–362, Edinburgh, Scotland, UK.

Birch, A., Osborne, M., and Koehn, P. (2007). CCG supertags in factored statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 9–16, Prague, Czech Republic.

Bisazza, A., Ruiz, N., and Federico, M. (2011). Fill-up versus interpolation methods for phrase-based SMT adaptation. In *Proceedings of the International Workshop on Spoken Language Translation (IWSLT)*, San Francisco, California, USA.

Cherry, C. and Foster, G. (2012). Batch tuning strategies for statistical machine translation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 427–436, Montréal, Canada.

Clark, J. H., Dyer, C., Lavie, A., and Smith, N. A. (2011). Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 176–181, Portland, Oregon, USA.

Clark, J. H., Lavie, A., and Dyer, C. (2012). One system, many domains: Open-domain statistical machine translation via feature augmentation. In *Proceedings of the 10th biennial conference of the Association for Machine Translation in the Americas (AMTA 2012)*, San Diago, California, USA.

Daumé, III, H. (2007). Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 256–263, Prague, Czech Republic.

Denkowski, M. and Lavie, A. (2014). Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the 9th Workshop on Statistical Machine Translation*, pages 376–380, Baltimore, Maryland, USA.

Finch, A. and Sumita, E. (2008). Dynamic model interpolation for statistical machine translation. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 208–215, Columbus, Ohio, USA.

Foster, G., Goutte, C., and Kuhn, R. (2010). Discriminative instance weighting for domain adaptation in statistical machine translation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 451–459, Cambridge, Massachusetts, USA.

Foster, G. and Kuhn, R. (2007). Mixture-model adaptation for SMT. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 128–135, Prague, Czech Republic.

Goto, I., Utiyama, M., Sumita, E., and Kurohashi, S. (2015). Preordering using a target-language parser via cross-language syntactic projection for statistical machine translation. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 14(3):13:1–13:23.

Heafield, K., Pouzyrevsky, I., Clark, J. H., and Koehn, P. (2013). Scalable modified Kneser-Ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 690–696, Sofia, Bulgaria.

Hopkins, M. and May, J. (2011). Tuning as ranking. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1352–1362, Edinburgh, Scotland, UK.

Irvine, A., Morgan, J., Carpuat, M., III, H. D., and Munteanu, D. (2013). Measuring machine translation errors in new domains. *Transactions of the Association for Computational Linguistics*, 1:429–440.

Isozaki, H., Hirao, T., Duh, K., Sudoh, K., and Tsukada, H. (2010). Automatic evaluation of translation quality for distant language pairs. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 944–952, Cambridge, Massachusetts, USA.

Jeblee, S., Feely, W., Bouamor, H., Lavie, A., Habash, N., and Oflazer, K. (2014). Domain and dialect adaptation for machine translation into Egyptian Arabic. In *Proceedings of the EMNLP 2014 Workshop on Arabic Natural Language Processing (ANLP)*, pages 196–206, Doha, Qatar.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., and Herbst, E. (2007). Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic.

Koehn, P., Och, F. J., and Marcu, D. (2003). Statistical phrase-based translation. In *HLT-NAACL 2003: Main Proceedings*, pages 127–133, Edmonton, Alberta, Canada.

Koehn, P. and Schroeder, J. (2007). Experiments in domain adaptation for statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 224–227, Prague, Czech Republic.

Kudo, T., Yamamoto, K., and Matsumoto, Y. (2004). Applying conditional random fields to Japanese morphological analysis. In Lin, D. and Wu, D., editors, *Proceedings of EMNLP 2004*, pages 230–237, Barcelona, Spain.

Matsoukas, S., Rosti, A.-V. I., and Zhang, B. (2009). Discriminative corpus weight estimation for machine translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 708–717, Singapore.

Nakazawa, T., Yaguchi, M., Uchimoto, K., Utiyama, M., Sumita, E., Kurohashi, S., and Isahara, H. (2016). ASEPC: Asian scientific paper excerpt corpus. In *Proceedings of the Tenth Edition of the Language Resources and Evaluation Conference (LREC-2016)*, Portoroz, Slovenia.

Och, F. J. (2003). Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167, Sapporo, Japan.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 311–318, Philadelphia, Pennsylvania, USA.

Sennrich, R. (2012). Perplexity minimization for translation model domain adaptation in statistical machine translation. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 539–549, Avignon, France.

Sennrich, R., Schwenk, H., and Aransa, W. (2013). A multi-domain translation model framework for statistical machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 832–840, Sofia, Bulgaria.

Snover, M., Dorr, B., Schwartz, R., Micciulla, L., and Makhoul, J. (2006). A study of translation edit rate with targeted human annotation. In *Proceedings of the 7th Biennial Conference of the Association for Machine Translation in the Americas (AMTA-2006)*, pages 223–231, Cambridge, Massachusetts, USA.

Yu, H., Huang, L., Mi, H., and Zhao, K. (2013). Max-violation perceptron and forced decoding for scalable MT training. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1112–1123, Seattle, Washington, USA.

# Bilingual Methods for Adaptive Training Data Selection for Machine Translation

**Boxing Chen** Boxing.Chen@nrc-cnrc.gc.ca
**Roland Kuhn** Roland.Kuhn@nrc-cnrc.gc.ca
**George Foster** George.Foster@nrc-cnrc.gc.ca
**Colin Cherry** Colin.Cherry@nrc-cnrc.gc.ca
National Research Council Canada, Ottawa, ON, Canada

**Fei Huang** feihuang@fb.com
Facebook, New York, NY, USA

**Abstract**

In this paper, we propose a new data selection method which uses semi-supervised convolutional neural networks based on bitokens (Bi-SSCNNs) for training machine translation systems from a large bilingual corpus. In earlier work, we devised a data selection method based on semi-supervised convolutional neural networks (SSCNNs). The new method, Bi-SSCNN, is based on bitokens, which use bilingual information. When the new methods are tested on two translation tasks (Chinese-to-English and Arabic-to-English), they significantly outperform the other three data selection methods in the experiments. We also show that the Bi-SSCNN method is much more effective than other methods in preventing noisy sentence pairs from being chosen for training. More interestingly, this method only needs a tiny amount of in-domain data to train the selection model, which makes fine-grained topic-dependent translation adaptation possible. In the follow-up experiments, we find that neural machine translation (NMT) is more sensitive to noisy data than statistical machine translation (SMT). Therefore, Bi-SSCNN which can effectively screen out noisy sentence pairs, can benefit NMT much more than SMT.We observed a BLEU improvement over 3 points on an English-to-French WMT task when Bi-SSCNNs were used.

## 1 Introduction

When building a statistical machine translation (SMT) system, it is important to choose bilingual training data that are of high quality [1] and that are typical of the domain in which the SMT system will operate. In previous work, these two goals of data selection, i.e., picking high-quality data and picking data that ensure the SMT system is well-adapted to a given domain, have often been achieved separately. For instance, the papers (Munteanu and Marcu, 2005; Khadivi and Ney, 2005; Okita et al., 2009; Jiang et al., 2010; Denkowski et al., 2012) focus on reducing the noise in the data. They use different scoring functions, such as language model perplexity, word alignment score, or IBM model 1 score, to score each sentence pair, top scored sentence pairs are selected. While the papers (Zhao et al., 2004; Lü et al., 2007; Yasuda et al., 2008; Moore and Lewis, 2010; Axelrod et al., 2011; Duh et al., 2013; Axelrod et al., 2015) focus on domain adaptation. They all select monolingual or bilingual data that

[1] Within each sentence pair, the target-language sentence is a good translation of the source sentence

are similar to the in-domain data according to some criterion. These state-of-the-art adaptive data selection approaches (Axelrod et al., 2011; Duh et al., 2013; Axelrod et al., 2015) search for bilingual parallel sentences using the difference in language model perplexity between two language models trained on in-domain and out-domain data, respectively. Furthermore, (Duh et al., 2013) extends these approaches from $n$-gram models to recurrent neural network language models (Mikolov et al., 2010). While some previous work considers achieving the two goals simultaneously, such as (Mansour et al., 2011) which uses IBM model 1 and a language model to do data selection, (Durrani et al., 2015) uses a neural network joint model to select the in-domain data.

In a recent paper (Chen and Huang, 2016), we describe one type of neural network for carrying out data selection: a semi-supervised Convolutional Neural Network (SSCNN) that is trained on the in-domain set to score one side of each sentence in a general-domain bilingual corpus (either the source side or the target side) for its suitability as training material for an SMT system. The highest-scoring sentence pairs are chosen to train the SMT system. Experiments described in that paper, covering three different types of test domain and four language directions, show that this SSCNN method yields significantly higher BLEU scores for the resulting SMT system than for three state-of-the-art data selection methods when the amount of training data selected is held constant. The advantage of the SSCNN over the earlier methods is especially dramatic when the amount of in-domain data used to train the selection model is small (less than 800 sentence pairs): the in-domain set for the SSCNN can be as few as 100 sentences, which makes fine-grained topic-dependent translation adaptation possible. In some cases, the SSCNN is so effective at selecting good training data that it is possible to greatly reduce the amount of training data for the SMT system without negative impact on translation quality: this reduces the footprint of the system, which can be advantageous for many practical applications.

In the experiments for (Chen and Huang, 2016), we found that the best variant of the SSCNN method was one in which we trained two CNN models - one that scores source-language sentences and one that scores target-language sentences - and sum the scores of these two models to get the overall score of each sentence pair in the bilingual corpus. Thus, the SSCNN variant we used assigns high scores to sentence pairs where both the source-language sentence and its target-language partner resemble sentences in the in-domain corpus. Note what this method does **not** do: it does not check that the target sentence is a good translation of the source sentence. Essentially, it scores the extent to which both the source and target sentence are in-domain, but does not in any way penalize bad translations. We say that such a method is "symmetric": it incorporates equal amounts of information from the source and the target language, but it is not "bilingual": it does not incorporate information about the quality of translations.

The main motivation for this paper is to explore CNN-based data selection techniques that are bilingual. It is based on semi-supervised CNNs that use bitokens as units instead of source or target words (Marino et al., 2006; Niehues et al., 2011). For the bitoken semi-supervised CNN, we should use the abbreviation "Bi-SSCNN". We also experiment with the bilingual method that combines IBM model 1 and language model (LM) scores and neural network joint model.

In this paper, we carried out experiments reported on two language pairs: Chinese-to-English and Arabic-to-English. We fix the number of training sentences to be chosen for the data selection techniques so that they can be fairly compared, and measure the BLEU score on test data from the resulting MT systems. It turns out that three techniques have roughly the same performance in terms of BLEU: the symmetric but non-bilingual word-based SSCNN method, and two symmetric, bilingual techniques - the simple IBM-LM method, and the NNJM method. The Bi-SSCNN method, on the other hand, outperforms all other methods: by +0.5 BLEU for

Chinese-to-English task, and by +0.3 BLEU for Arabic-to-English task.

Because the main motivation for the paper is exploration of bilingual methods for SSCNN-based data selection, we perform another set of experiments to see how good each method is at rejecting sentence pairs whose target side is not a translation of the source side. We do this by permuting 50% of the pairs in the bilingual corpus. We then count the proportion of pairs chosen by each method that are mismatched. In these experiments, all three bilingual methods - IBM-LM, NNJM, and Bi-SSCNN outperform the other methods (they chose a smaller proportion of mismatched sentence pairs) and Bi-SSCNN is even more effective than the other two bilingual methods.

## 2 Four Data Selection Methods

In this section, we focus on four methods for data selection: the IBM-LM method, the NNJM method, the original word-based SSCNN method, and the Bi-SSCNN method. The IBM-LM method is similar to Mansour et al. (2011), which is a simple bilingual method that is a good baseline for other bilingual methods. The NNJM-based data selection (Durrani et al., 2015) is the first bilingual NN method. The word-based SSCNN method is described in (Chen and Huang, 2016). The SSCNN method is symmetrical but not bilingual. The Bi-SSCNN method is a newly proposed method, which is symmetrical as well as bilingual.

### 2.1 Data Selection with IBM1 and Language Models

The IBM-LM method is straightforward, since state-of-the-art methods use IBM models to measure the mutual translation quality, and language models to select in-domain data. We combine length-normalized scores from these models into a global score, i.e., target-given-source IBM model 1 score, source-given-target IBM model 1 score, source language model cross entropy difference, and target language model cross entropy difference, are normalized by the corresponding sentence length and then averaged to obtain one score. The IBM models are trained on whole general-domain data plus the in-domain set, while the language models are trained on the in-domain set or a small (equal size to the in-domain set), random subset of the general-domain corpus (excluding the small in-domain set).

To select a subset of data to be used for training an SMT system from a bilingual corpus, the user must specify the number $N$ of sentence pairs to be chosen. The $N$ sentence pairs with the highest global scores $S(s,t)$ will be selected. This method is symmetrical - the roles of the source-language and target-language sides of the corpus are the same - and bilingual, because the IBM model 1 measures the degree to which each target sentence $t$ is a good translation of its partner $s$, and vice versa.

### 2.2 Data Selection with Neural Net Joint Model (NNJM)

The Neural Network Joint Model (NNJM), as described in (Devlin et al., 2014), is a joint language and translation model based on a feedforward neural net (NN). It incorporats a wide span of contextual information from the source sentence, in addition to the traditional $n$-gram information from preceding target-language words. Specifically, when scoring a target word $w_i$, the NNJM inputs not only the $n-1$ preceding words $w_{i-n+1}, ..., w_{i-1}$, but also $2m+1$ source words: the source word $s_i$ most closely aligned with $w_i$ along with the $m$ source words $s_{i-m}, ..., s_{i-1}$ to the left of $s_i$ and the $m$ source words $s_{i+1}, ..., s_{i+m}$ to the right of $s_i$.

The NNJMs used in our experiments input 4-grams on the target side and windows of size 11 on the source side (the aligned word, along with 5 words to the left of it and 5 words on its right). Training is done in two passes: a pass over general-domain data plus the in-domain set, followed by a pass over the in-domain set. While in the second pass, the source and target word embeddings are fixed. For a particular language pair, for instance, Chinese-to-English,

four NNJMs are trained. There are two NNJMs informed by a wide-source language (Chinese) context that act as LMs for the target language (English): a positive one NNJM(+,Chinese-to-English) modeling in-domain target-language (English) sentences, and a negative one NNJM(-,Chinese-to-English) modeling out-of-domain target-language (English) sentences. Both are initialized with parameters and fixed source and target word embeddings which are learned on the entire general-domain corpus plus the in-domain set. The second phase of training for NNJM(+,Chinese-to-English) is on the in-domain set. It would be nice if we could train NNJM(-,Chinese-to-English) on sentence pairs that are known to be out-of-domain, but there is no easy way of obtaining such sentence pairs, so we simply carry out the second phase of training for this negative NNJM on a small, random subset of the general-domain corpus (excluding the small in-domain set). The difference between the two scores, score(NNJM(+,Chinese-to-English)) minus score(NNJM(-,Chinese-to-English)), as calculated on a sentence pair is an indicator of how close to the in-domain set the sentence pair is.

We train in similar manner a positive and a negative NNJM that act as LMs for the source language (Chinese) while consulting a wide context in the target language (English): NNJM(+,English to Chinese) and NNJM(-,English to Chinese). The global, symmetrical NNJM score $S$ for a sentence pair is made up of equal contributions from these four models.

Since this metric contains information about the translation relationship between each source sentence and its target counterpart, and since the ways in which the source and target languages are used are mirror images of each other, the NNJM data selection method is both bilingual and symmetrical.

### 2.3 Data Selection with Semi-Supervised CNN

As described in more detail in (Chen and Huang, 2016), we were inspired by the success of convolutional neural networks (CNNs) applied to image and text classification (Krizhevsky et al., 2012; Kim, 2014; Johnson and Zhang, 2015a,b) to use CNNs to classify training sentences in either the source language or the target language as in-domain or out-of-domain.

Convolutional neural networks (CNNs) (LeCun and Bengio, 1998) are feed-forward neural networks that exploit the internal structure of data through convolutional and pooling layers; each computation unit of the convolutional layer processes a small region of the input data. When CNNs are applied to text input, the convolution layers process small regions of a document, i.e., a sequence of words. CNNs are now used in many text classification tasks (Kalchbrenner et al., 2014; Johnson and Zhang, 2015b; Wang et al., 2015). Chen and Huang (2016) use CNNs to classify sentence pairs to in-domain and out-of-domain sentence pairs.

In many of these studies, the first layer of the network converts words to word embeddings using table lookup; the embeddings are sometimes pre-trained on an unnlabeled data. The embeddings remain fixed during subsequent model training. A CNN trained with small number of labled data and pre-trained word embeddings on large unlabeled data is termed "semi-supervised". Because we were interested in data selection scenarios where only small amounts of in-domain data are available, we chose to use semi-supervised CNNs (SSCNNs) with $word2vec$ embeddings (Mikolov et al., 2013) pre-trained on a large general-domain, monolingual corpus. The input region vector that represents a segment of data can be either a concatenation of word vectors, in which the order of concatenation is the same as the word order in the sentence, or it can be a bag-of-word/$n$-gram vector. The bag-of-word (BOW) representation loses word order information but is more robust to data sparsity. A CNN whose input being BOW representation is called $bow$-CNN while input with concatenation of vectors is called $seq$-CNN.

On the other hand, the input sentence can also be represented with one-hot vectors, where each vector's length is the vocabulary size, value 1 at index $i$ indicates word $i$ appears in the

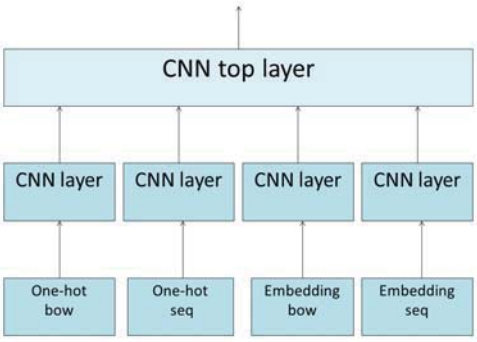



Figure 1: Semi-supervised CNN structure.

sentence, and 0 indicates its absence. The one-hot vector can be either one-hot "BOW" or one-hot "seq" too. We input all four kinds of representations, i.e., bag-of-word one-hot vectors (one-hot BOW), concatenation of one-hot vectors (one-hot seq), bag-of-word embedding vectors (embedding BOW), concatenation of embedding vectors (embedding seq), to the CNN layers to train the classification model, as shown in Figure 1.

To train the CNN itself, we take the in-domain set as the positive training sample and randomly select the same number of sentences from the general-domain training data as the negative training sample. The positively and negatively labeled data and the word embeddings are fed to the convolution layer to train the final classification model. The resulting SSCNN is then used to score each sentence in the general-domain corpus. As with the other methods described above, the SSCNN was applied symmetrically - two SSCNNs were trained, one on the source-language half of the in-domain set, the other on the target-language half, and their scores were summed to obtain a global score for each sentence pair in the big general-domain corpus. The top $N$ sentence pairs are selected to train the SMT system. Note that though this SSCNN method is symmetrical, it is not bilingual: there is no evaluation of whether the two halves of each sentence pair are good translations of each other.

The experimental results given in (Chen and Huang, 2016) for the SSCNN method on four different language directions and a variety of genres (SMS, tweets, Facebook posts, etc) show that the resulting SMT systems typically outperform baseline systems trained with all the general-domain data, and (for a fixed amount of selected training data) previously state-of-the-art data selection methods. The advantage of the SSCNN method over other data selection techniques is especially strong when the size of the initial in-domain data (the dev set) is small. Therefore, if we wish to build a large scale topic-specific MT system with hundreds of topics, we only need to collect a few hundreds sentence pairs for each topic. This makes fine-grained topic-dependent translation adaptation possible.

### 2.4 Data Selection with a Bitoken Semi-supervised CNN

Despite the excellent performance of the SSCNN method described in the previous paragraphs, we believed that further improvement might be possible if we devised a method based on SSCNNs that was not only symmetrical, but also bilingual. The problem with SSCNN is that unlike the IBM-LM method or the NNJM method nothing about it filters out sentence pairs whose source and target halves are bad translations of each other. We decided to experiment with SSCNNs that take as input the bitokens of (Marino et al., 2006; Niehues et al., 2011).

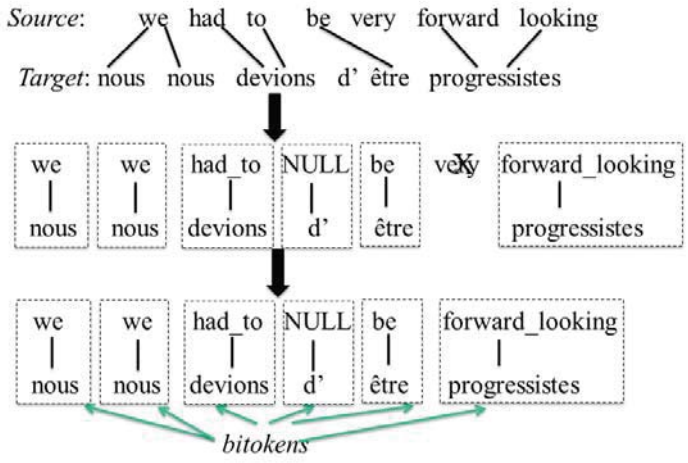



Figure 2: Bitoken sequence.

The paper (Niehues et al., 2011) describes a "bilingual language model" (biLM): the idea that SMT systems would benefit from wider contextual information from the source sentence. BiLMs provide this context by aligning each target word in the training data with source words to create bitokens. An $n$-gram bitoken LM for the sequence of target words is then trained.

Figure 2 (taken from (Stewart et al., 2014)) shows how a bitoken sequence is obtained from a word-aligned sentence pair for the English to French language pair. Unaligned target words (e.g., French word "d'" in the example) are aligned with NULL. Unaligned source words (e.g., "very") are dropped. A source word aligned with more than one target word (e.g., "we") aligned with two instances of "nous" is duplicated: each target word aligned with it receives a copy of that source word.

The word embeddings for bitokens are learned directly by word2vec, treating each bitoken as a word. For instance, in the French sentence shown in Figure 2, there are two occurrences of the new target-language word *nous/we*, one occurrence of *devions/had_to*, etc. To reduce the size of the bitoken vocabulary, we exclude low-frequency bitokens from consideration: to be taken into account, a Chinese-English bitoken must occur at least 10 times in the training data, and an Arabic-English bitoken must occur at least 5 times.

After this embedding has been performed, the Bi-SSCNN is trained similarly as was described above for the original word-based SSCNN. The only difference is the pooling strategy. For the word-based SSCNN, we use maximum-pooling, while for the bitoken-based SSCNN, we use average-pooling. This is because if a sentence contains one or more in-domain words, it is very likely an in-domain sentence, so we use maximum pooling to highlight those typical in-domain words for the word-based SSCNN. The reason for choosing average-pooling for Bi-SSCNN is that if a sentence pair contains one or more in-domain bitokens, it is not necessarily a good quality in-domain sentence pair. It is only when a sentence pair contains some in-domain bitokens, and most of the bitokens are good translations, that it is considered to be a high quality in-domain sentence pair.

| language | zh2en | ar2en |
|---|---|---|
| test domain | webforum | nw&wl |
| train size | 12.20M | 5.29M |
| dev size | 2,748 | 1,360 |
| test size | 1,224 | 5,812 |

Table 1: Summary of the data. Data is given as the number of sentence pairs, "M" represents "million". "nw" stands for "newswire", "wl" stands for "weblog".

As with the other methods, this data selection method is applied in a bilingual manner: the gobal score used to evaluate sentence pairs is the unweighted sum of a score for two bi-SSCNNs, one of which reverses the polarity of the source language (Chinese or Arabic) and the target language (English).

## 3 Experiments on SMT

Our goal is to adapt the MT system when only a small amount of in-domain data is available. So in most of our experiments, we ignored domain information about the training data, such as the source of each corpus. What we have is a small development set (dev) and one or more test sets (test) which are in the same domain.

### 3.1 Data setting

We carried out experiments in two different data settings. The first setting is the Chinese-to-English "webforum" task from the BOLT project (zh2en); the test domain is a combination of news and web posts. The training data from LDC[2] are a mixture of newswire, web crawls, UN proceedings, etc. The second setting is the NIST 2012 Arabic-to-English task (ar2en). Again, the training data are available from LDC, and the test domain is a combination of newswire and weblog. We use the NIST 2008 test data as the dev set and the NIST 2012 test set as the test.

Table 1 summarizes the statistics of the training, dev, and test data for both tasks. Note that for both, the test domain includes newswire data, and the training data include a proportion of newswire data are extracted from comparable data, around 10% for the Chinese-to-English task and 20% for the Arabic-to-English task. We used all the comparable data available from LDC. Some of the comparable data are quite noisy, making the task of data selection more challenging both for two tasks.

Once a subset of the large in-domain, bilingual corpus has been selected by one of the methods described above, that subset is used as training data for a standard phrase-based SMT system.

### 3.2 Experimental setup

We employ the dev set as in-domain data. All the supervised CNN models (both the SSCNN ones and the Bi-SSCNN ones) are trained with the in-domain dev data as positive examples and an equal number of randomly selected general-domain sentences as negative examples. All the meta-parameters of the CNN are tuned on held-out data; we generate one-hot based $bow$-regions and $seq$-regions, word-embedding-based $bow$-regions and $seq$-regions and input them to the CNN. We set the region size to 5 and stride size to 1. The non-linear function we chose is "ReLU", the number of weight vectors or neurons is 500. We use the online available CNN

[2]https://catalog.ldc.upenn.edu/

toolkit $conText$[3]. To train the general domain word embedding, we used $word2vec$[4]. The size of the vector was set to 300.

The baseline SMT system for each language direction, "alldata" is trained using all general-domain data. All other systems are trained with a subset of the general-domain data of fixed size: 1.8 million sentence pairs (about 15% of the available training data) for the Chinese-to-English task, and 1.4 million sentence pairs (about 26% of the available training data) for the Arabic-to-English task. We experimented with the following five data selection methods. Note that the last three methods are bilingual - they measure not only the quality of the source and target sides of a sentence pair, but also the degree to which one is a good translation of the other - but the first two are not:

1. LM: Data selection by 3-gram LMs with Witten-Bell [5] smoothing, using bilingual cross entropy difference as the criterion. This is considered to be a state-of-the-art data selection method for domain adaptation (Axelrod et al., 2011). The "sum LM" variant uses the sum of the source and target LM scores for a sentence pair.

2. SSCNN: Data selection by semi-supervised CNN based on monolingual tokens (Section 2.3)

3. IBM-LM: Data selection by both IBM and language models (Section 2.1)

4. NNJM: Data selection by neural network joint models (Section 2.2)

5. Bi-SSCNN: Data selection by bitoken based semi-supervised CNN (Section 2.4)

### 3.3 Experimental results

We evaluated the system using the BLEU (Papineni et al., 2002) score on the test set. Following (Koehn, 2004), we apply the bootstrap resampling test to do significance testing. Table 2 summarizes the results for each task. The number of selected sentence pairs for each language pair (1.8 million pairs for Chinese-to-English, and 1.4 million pairs for Arabic-to-English) was decided on the basis of tests on held-out data using the IBM-LM method. That is, 1.8 million was the value of $N$ that maximized the BLEU score of the final SMT system when IBM-LM was used to select $N$ sentence pairs as training data for Chinese-to-English, and 1.4 had the same property for Arabic-to-English.

In the table, the bilingual methods are the ones below the horizontal line. All methods shown were applied in their symmetrical version, where the global score is obtained by adding a source-based score to a target-based score.

It can be seen from the table that the three NN-based data selection methods - the original word-based SSCNN, NNJM, and bitoken-based SSCNN - outperform the other methods. Among these three, Bi-SSCNN is significantly better than the other two, outperforming the original SSCNN by +0.5 BLEU on Chinese-to-English and +0.3 BLEU on Arabic-to-English.

What about the original motivation for devising data selection methods based on IBM-LM, NNJMs and Bi-SSCNN: that methods employing bilingual information will do a better job of screening out noisy sentence pairs, i.e., sentence pairs where each side is a bad translation of the other? The BLEU results above do not make this aspect of the behaviour of the various methods clear. For instance, the bilingual IBM-LM method obtains lower scores for both the Chinese-to-English and the Arabic-to-English task than the non-bilingual original SSCNN method.

[3] http://riejohnson.com/cnn_download.html

[4] https://code.google.com/archive/p/word2vec/

[5] For small amounts of data, Witten-Bell smoothing performed better than Kneser-Ney smoothing in our experiments

| | symmetrical | bilingual | zh2en | ar2en |
|---|---|---|---|---|
| alldata | – | – | 24.6 | 45.7 |
| LM | yes | no | 24.7 | 45.2 |
| SSCNN | yes | no | 25.1** | 45.9 |
| IBM-LM | yes | yes | 24.9 | 45.4 |
| NNJM | yes | yes | 25.0* | 45.8 |
| Bi-SSCNN | yes | yes | 25.6**++ | 46.2**+ |

Table 2: Summary of BLEU results. */** means result is significantly better than the "alldata" baseline at $p < 0.05$ or $p < 0.01$ level, respectively. +/++ means result is significantly better than the "SSCNN" method at $p < 0.05$ or $p < 0.01$ level, respectively.

We therefore decided to test the ability of the methods above to screen out noisy sentence pairs directly. Inspired by the experimental approach of (Goutte et al., 2012), we deliberately corrupted a randomly chosen 50% of the sentence pairs in the two large general-domain corpora for Chinese-to-English and Arabic-to-English by permuting the order of the target-language (English) sentences, while leaving the rest of each general-domain corpus (and the dev set on which data selection methods are trained) untouched. We can then see what percentage of the sentence pairs chosen by each data selection method have a mismatched source-language and target-language side. Because the NNJM-based and bitoken SSCNN-based approaches use word alignment information, we re-ran word alignment on all the general-domain data (using the IBM and HMM models trained on the original version of the data).

Table 3 shows the proportion of mismatched sentence pairs in the top $N$ selected sentence pairs. For each language direction, two different values of $N$ are tried: $N = 1.8M$ and $N = 200K$ for Chinese-to-English, and $N = 1.4M$ and $N = 200K$ for Arabic-to-English. Data selection based on only a source LM ("src LM") or target LM ("tgt LM") is completely ineffective at screening out mismatched sentence pairs: their proportion in the selected subset is around 50%, just as it was in the large corpus the subset was selected from. Symmetrizing LM data selection by adding the source LM and target LM yields an improvement that is particularly noticeable when only 200K sentence pairs are selected for each language direction: 38% of the highest-scoring 200K Chinese-English pairs and 39% of the highest-scoring 200K Arabic-English pairs are mismatched. The variants of the original SSCNN method show a similar pattern, with the symmetrical version performing better than the versions that rely only on the source-side or target-side scores.

A definite improvement in performance is seen when we consider the three bilingual data selection methods: IBM-LM, NNJM, and Bi-SSCNN. By far the best performer of the three is Bi-SSCNN. When it ranks sentence pairs in the Chinese-English corpus, of the 1.8M pairs with the highest scores, about 29% are mismatched; of the 200K pairs with the highest scores, about 11% are mismatched. The mismatch proportions for the highest-scoring Arabic-English pairs are about 28% for the 1.4M subset and 10% for the 200K subset.

These results show that the three bilingual methods - IBM-LM, NNJM, and Bi-SSCNN - are more effective at screening out noisy sentence pairs than the non-bilingual methods. It may seem surprising that all three of these methods do not therefore also have the highest BLEU scores in Table 2. A possible reason is that the general-domain data we used to train the system are not too noisy. Moreover, (Goutte et al., 2012) showed that phrase-based MT is highly robust to the sentence alignment errors: "performance is hardly affected when the misalignment rate is below 30%, and introducing 50% alignment error brings performance down less than 1 BLEU point." Thus, the proportion of misaligned sentence pairs in a training corpus and the BLEU

| | symmetrical | bilingual | zh2en | ar2en | zh2en | ar2en |
|---|---|---|---|---|---|---|
| #selected | | | 1.8M | 1.4M | 200K | 200K |
| src LM | no | no | 0.501 | 0.502 | 0.497 | 0.489 |
| tgt LM | no | no | 0.496 | 0.505 | 0.495 | 0.492 |
| sum LM | yes | no | 0.461 | 0.454 | 0.376 | 0.389 |
| src SSCNN | no | no | 0.487 | 0.495 | 0.481 | 0.481 |
| tgt SSCNN | no | no | 0.488 | 0.498 | 0.476 | 0.488 |
| sum SSCNN | yes | no | 0.453 | 0.455 | 0.365 | 0.401 |
| IBM-LM | yes | yes | 0.411 | 0.417 | 0.355 | 0.312 |
| NNJM | yes | yes | 0.428 | 0.402 | 0.376 | 0.298 |
| Bi-SSCNN | yes | yes | 0.292 | 0.280 | 0.113 | 0.100 |

Table 3: The proportion of mismatched sentence pairs in the top $N$ sentence pairs selected from the 50% sentence pairs permuted corpora.

| language | zh2en | ar2en |
|---|---|---|
| alldata | 24.6 | 45.7 |
| Bi-SSCNN | 25.6 | 46.2 |
| clean-in | 25.4 | 46.0 |

Table 4: BLEU Results for Manually Selected Data.

score obtained from a system on that corpus are not highly correlated.

For most practical purposes, we care more about a data selection method ability to produce a training corpus that will yield an SMT system with a high BLEU score than its ability to screen out noisy sentence pairs. Nevertheless, it is reassuring that the method which yields the highest BLEU score in our experiments, Bi-SSCNN, is also the one that is far better than the other methods at screening out noisy pairs.

In another experiment, we manually selected subsets made up of data that were likely to be in-domain (because they came from newswire and weblogs, just like the test set) and clean (we excluded comparable data). We selected around 1.4M sentence pairs for Chinese-to-English task and 1.0M for Arabic-to-English task.

As Table 4 shows, the resulting "clean-in" training data set yielded an SMT system that performed about as well, or slightly worse, than an SMT system trained on data selected by "Bi-SSCNN". This is a strong argument in favour of using Bi-SSCNN, which is fully automatic and doesn't require any outside knowledge about the sentence pairs in the large general-domain corpus.

### 3.4 Discussion

When we examined a random selection of sentence pairs in the big general-domain corpora and looked at their scores as assigned by the Bi-SSCNN method, we made an interesting observation. The sentence pairs with lowest scores are out-of-domain, as expected, but also tend to have very good translation quality. The overall ordering (from highest to lowest score) tends to be 1. clean and in-domain pairs 2. noisy and in-domain pairs 3. noisy and out-of-domain pairs 4. clean and out-of-domain pairs.

This tendency, which is surprising at first glance, is understandable. Both the positive samples in the dev set and the sampled negative examples used to train the Bi-SSCNN classification model are of good quality. Thus, the Bi-SSCNN has learned two top priorities: selecting **for**

clean, in-domain sentence pairs, and selecting **against** clean, out-of-domain sentence pairs. It thus scores the former type of sentence pair highest, and the latter type lowest. Noisy sentence pairs thus receive intermediate scores. This behaviour may be desirable from a practical point of view: clean, out-of-domain sentence pairs are dangerous in the sense that they will populate the phrase table with phrase translations that are likely to compete with the correct translations for the given domain. Noisy out-of-domain sentence pairs will have a more random effect, sprinkling the phrase table with low-frequency, unlikely translations, thus doing less harm (and noisy in-domain pairs will probably have a mildly positive effect on the performance of the resulting SMT system).

In our experiments, NNJM-based data selection did not stand out from other methods either in terms of its impact on BLEU or in terms of its ability to screen out noisy sentence pairs. Study of the NNJMs we trained suggests that part of the problem is their limited vocabulary size: 32K words for both the source and the target language. All other words are mapped into a small number of clusters. Unfortunately, many sentence pairs of poor quality end up with several occurrences of particular cluster on both the source and the target side, which may mislead the NNJMs into thinking that these sentence pairs are of good quality.

Note also that the SMT systems trained on the selected data did not contain an NNJM feature (or, indeed, any neural component). It is possible that the NNJM-based method will work better when applied to the task of selecting data from a bilingual corpus to train a second, larger NNJM. We intend to explore this possibility in our future work.

## 4 Follow-up experiments on NMT

After we submitted this paper, we did some experiments with a neural machine translation (NMT) system (Sutskever et al., 2014; Bahdanau et al., 2015) using Bi-SSCNN. [6] The experiments were carried out with an open source system called Nematus (Sennrich et al., 2016), which is an attention-based NMT system (Bahdanau et al., 2015). We carried out experiments on English-to-French (en2fr) WMT task [7]. The training data contain 12 million sentence pairs; the dev set is a concatenation of newstest2012 and 2013, which contains 6,003 sentence pairs; the test set is newstest2014, which contains 3,003 sentence pairs.

Table 4 summarizes our experiments. If we introduce 30% sentence alignment error to the original data, we lost over 3 BLEU score (35.4 vs 32.0 on the 12.0M data set; 33.7 vs 30.6 on the 6.4M sampled data set.) Then, if we apply Bi-SSCNN to the 12.0M data which contains 30% sentence alignment error, by carefully manipulating the negative training samples of the Bi-SSCNN, we can turn on the noise reduction and domain adaptation separately. Experiment 5 shows that when the data contain 30% alignment error, Bi-SSCNN can screen out the noise in the data and improve the performance to 33.9 BLEU, from 30.6 BLEU for the experiment 4 baseline. Experiment 6 showed that if we only use Bi-SSCNN for domain adaptation on a data with 30% sentence alignment error, the improvement is smaller, only 0.4 BLEU. Finally, if we turn on both noise reduction and domain adaptation for Bi-SSCNN, we obtain the best result, which is 34.2: a 3.6 BLEU point improvement over the baseline in experiment 4. This series of experiments shows that 1. neural machine translation is more sensitive to noise than SMT; 2. Bi-SSCNN is effective in carrying out both noise reduction and domain adaptation.

## 5 Conclusions

We proposed a new method for data selection from a large bilingual corpus for the purpose of training an SMT system. The new method is based on a bitoken semi-supervised convolutional

[6] Due to the time limit, we did not finish the experiments with other data selection methods.

[7] The data is available at http://www-lium.univ-lemans.fr/ schwenk/nnmt-shared-task/

| id | data size | data description | BLEU |
|---|---|---|---|
| 1 | 12.0M | all original data | 35.4 |
| 2 | 12.0M | 30% alignment error | 32.0 |
| 3 | 6.4M | sampled from data 1: original data | 33.7 |
| 4 | 6.4M | sampled from data 2: with 30% alignment error | 30.6 |
| 5 | 6.4M | Bi-SSCNN noise reduction on data 2 | 33.9 |
| 6 | 6.4M | Bi-SSCNN domain adaptation on data 2 | 31.0 |
| 7 | 6.4M | Bi-SSCNN noise reduction and domain adaptation on data 2 | 34.2 |

Table 5: BLEU Results for English-to-French WMT task with neural machine translation system.

neural networks. It outperformed its nearest competitor, a method that uses a word-based SSCNN, by +0.5 BLEU on a Chinese-to-English task and by +0.3 BLEU on an Arabic-to-English task. Since one of the motivations underlying the creation of the Bi-SSCNN method (and two other data selection methods, those based on IBM-LM and NNJM) was the ability to screen out noisy sentence pairs, we carried out another type of experiment in which the methods were explicitly tested for the ability to do this when half the pairs in the bilingual corpus have been deliberately corrupted. According to this criterion, too, the Bi-SSCNN outperformed all other methods.

In the follow-up experiments, we find that neural machine translation is more sensitive to noisy data than statistical machine translation. Therefore, Bi-SSCNN, which can effectively screen out noisy sentence pairs, can benefit NMT much more than SMT. For instance, given a potential training corpus with 30% sentence alignment error, data selected with Bi-SSCNN yields a system with a performance gain of over 3 BLEU points above the baseline.

## References


Axelrod, A., He, X., and Gao, J. (2011). Domain adaptation via pseudo in-domain data selection. In *EMNLP 2011*.

Axelrod, A., Resnik, P., He, X., and Ostendorf, M. (2015). Data selection with fewer words. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 58–65, Lisbon, Portugal.

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *ICLR*.

Chen, B. and Huang, F. (2016). Semi-supervised convolutional networks for translation adaptation with tiny amount of in-domain data. In *Proceedings of the SIGNLL Conference on Computational Natural Language Learning*, Berlin, Germany.

Denkowski, M., Hanneman, G., and Lavie, A. (2012). The cmu-avenue french-english translation system. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 261–266, Montréal, Canada. Association for Computational Linguistics.

Devlin, J., Zbib, R., Huang, Z., Lamar, T., Schwartz, R., and Makhoul, J. (2014). Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1370–1380, Baltimore, Maryland. Association for Computational Linguistics.

Duh, K., Neubig, G., Sudoh, K., and Tsukada, H. (2013). Adaptation data selection using neural language models: Experiments in machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 678–683, Sofia, Bulgaria.

Durrani, N., Sajjad, H., Joty, S., Abdelali, A., and Vogel, S. (2015). Using joint models for domain adaptation in statistical machine translation. In *Proceedings of the Fifteenth Machine Translation Summit (MT Summit XV)*.

Goutte, C., Carpuat, M., and Foster, G. (2012). The impact of sentence alignment errors on phrase-based machine translation performance. In *Tenth Biennial Conference of the Association for Machine Translation in the Americas (AMTA-2012)*, San Diego, USA.

Jiang, J., Way, A., and Carson-Berndsen, J. (2010). Lattice score based data cleaning for phrase-based statistical machine translation. In *14th Annual Conference of the European Association for Machine Translation*, Saint-Raphael, France.

Johnson, R. and Zhang, T. (2015a). Effective use of word order for text categorization with convolutional neural networks. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 103–112, Denver, Colorado.

Johnson, R. and Zhang, T. (2015b). Semi-supervised convolutional neural networks for text categorization via region embedding. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems 28*, pages 919–927. Curran Associates, Inc.

Kalchbrenner, N., Grefenstette, E., and Blunsom, P. (2014). A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 655–665, Baltimore, Maryland.

Khadivi, S. and Ney, H. (2005). Automatic filtering of bilingual corpora for statistical machine translation. In *Natural Language Processing and Information Systems, 10th International Conference on Applications of Natural Language to Information Systems*, pages 263–274, Alicante, Spain.

Kim, Y. (2014). Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882.

Koehn, P. (2004). Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Barcelona, Spain.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc.

LeCun, Y. and Bengio, Y. (1998). Convolutional networks for images, speech, and time series. In Arbib, M. A., editor, *The Handbook of Brain Theory and Neural Networks*, pages 255–258. MIT Press, Cambridge, MA, USA.

Lü, Y., Huang, J., and Liu, Q. (2007). Improving Statistical Machine Translation Performance by Training Data Selection and Optimization. In *Proceedings of the 2007 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Prague, Czech Republic.

Mansour, S., Wuebker, J., and Ney, H. (2011). Combining translation and language model scoring for domain-specific data filtering. In *Proceedings of IWSLT*.

Marino, J. B., Banchs, R. E., Crego, J. M., de Gispert, A., Lambert, P., Fonollosa, J. A. R., and Costa-jussà, M. R. (2006). N-gram-based machine translation. *Computational Linguistics*, 32(4):527–549.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Mikolov, T., Karafiát, M., Burget, L., Černocký, J., and Khudanpur, S. (2010). Recurrent neural network based language model. In *Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH2010)*, pages 1045–1048. International Speech Communication Association.

Moore, R. C. and Lewis, W. (2010). Intelligent selection of language model training data. In *ACL 2010*.

Munteanu, D. S. and Marcu, D. (2005). Improving machine translation performance by exploiting non-parallel corpora. *Computational Linguistics*, 31(4):477–504.

Niehues, J., Herrmann, T., Vogel, S., and Waibel, A. (2011). Wider context by using bilingual language models in machine translation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 198–206, Edinburgh, Scotland. Association for Computational Linguistics.

Okita, T., Naskar, S., and Way, A. (2009). Noise reduction experiments in machine translation. In *the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, Bled, Slovenia.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). BLEU: A method for automatic evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 311–318, Philadelphia. ACL.

Sennrich, R., Haddow, B., and Birch, A. (2016). Edinburgh neural machine translation systems for wmt 16. In *Proceedings of the First Conference on Machine Translation*, pages 371–376, Berlin, Germany. Association for Computational Linguistics.

Stewart, D., Kuhn, R., Joanis, E., and Foster, G. (2014). Coarse 'split and lump' bilingual language models for richer source information in smt. In *Eleventh Biennial Conference of the Association for Machine Translation in the Americas (AMTA-2014)*, Vancouver, Canada.

Sutskever, I., Vinyals, O., and Le, Q. V. V. (2014). Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27 (NIPS)*, pages 3104–3112.

Wang, P., Xu, J., Xu, B., Liu, C., Zhang, H., Wang, F., and Hao, H. (2015). Semantic clustering and convolutional neural network for short text categorization. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 352–357, Beijing, China.

Yasuda, K., Zhang, R., Yamamoto, H., and Sumita, E. (2008). Method of selecting training data to build a compact and ef?cient translation model. In *International Joint Conference on Natural Language Processing.*

Zhao, B., Eck, M., and Vogel, S. (2004). Language model adaptation for statistical machine translation with structured query models. In *Proceedings of the International Conference on Computational Linguistics (COLING) 2004*, Geneva.

# Neural Interactive Translation Prediction

**Rebecca Knowles** rknowles@jhu.edu
**Philipp Koehn** phi@jhu.edu
Department of Computer Science, Center for Language and Speech Processing
Johns Hopkins University, Baltimore, MD, 21218, USA

**Abstract**

We present an interactive translation prediction method based on neural machine translation. Even with the same translation quality of the underlying machine translation systems, the neural prediction method yields much higher word prediction accuracy (61.6% vs. 43.3%) than the traditional method based on search graphs, mainly due to better recovery from errors. We also develop efficient means to enable practical deployment.

Interactive translation prediction (also called interactive machine translation) is an editing mode for translators who interact with machine translation output. In this mode, the machine translation system makes suggestions for how to complete the translation ("auto-complete"), and the translator either accepts suggested words or writes in their own translation. When the suggestion is rejected, the machine translation system recomputes its prediction for how to complete the sentence from the given prefix and presents the corrected version to the translator.

In prior work, phrase-based machine translation systems have been used for interactive translation prediction, and suggestions were made either by re-decoding constrained by the prefix (Green et al., 2014) or by searching for the prefix in the original search graph (Och et al., 2003; Barrachina et al., 2009). Recently, neural translation models have been proposed and in some cases have shown superior performance over phrase-based models (Jean et al., 2015; Sennrich et al., 2016). We propose to use such models for interactive translation prediction. Parallel to this work, Wuebker et al. (2016) also explore a similar approach to using neural MT for interactive translation prediction.

The decoding mechanism for neural models provides a natural way of doing interactive translation prediction. We show that neural translation models can provide better translation prediction quality and improved recovery from rejected suggestions. We also develop efficient methods that enable neural models to meet the speed requirements of live interactive translation prediction systems.

## 1 Interactive Translation Prediction

Interactive translation prediction leaves the translator in charge of writing the translation and places the machine translation system in an assisting role. Rather than having a translator post-edit machine translated output, the system actively makes suggestions as the translator writes their translation. This modality is similar to an auto-complete function. Whenever the translator diverges from the suggestion (by typing a word that differs from the model's suggestion), the system recalculates (taking the translator's input into account) and generates new suggestions. Implementations of interactive translation can be found in the CASMACAT[1] (see Figure 1) and

[1] http://www.casmacat.eu


Figure 1: Interactive translation prediction in CASMACAT: The system suggests to continue the translation with the words *mehr als 18*, which the user can accept by pressing the TAB key.

Lilt[2] computer aided translation tools. This interaction mode is preferred by translators over post-editing (Koehn, 2009).

The goal of interactive translation prediction is to offer suggestions that the translator will accept. Existing approaches with statistical machine translation use the static search graph produced by the machine translation system. The system attempts to match the partial translator input (called the *prefix*) to the search graph, using approximate string matching techniques (minimal string edit distance) when an exact match cannot be found. As a baseline, we use a statistical machine translation system for interactive translation prediction that closely follows Koehn (2009) and Koehn et al. (2014). The prefix could be matched by constraint decoding, however, at a much higher computational cost.

Initial work on interactive translation prediction can be found in the TransType and TransType2 projects (Langlais et al., 2000; Foster et al., 2002; Bender et al., 2005; Barrachina et al., 2009). Our current work focuses on how to produce suggestions; for various approaches to interaction modalities, see Sanchis-Trilles et al. (2008) (mouse actions), Alabau et al. (2011) (hand-writing) and Cubel et al. (2009) (speech).

## 2 Neural Machine Translation

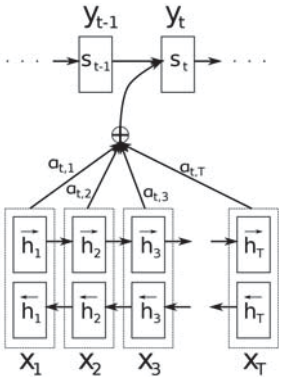
The use of neural network methods in machine translation has followed their recent success in computer vision and automatic speech recognition. Motivations for their use include better generalization of the statistical evidence (such as the use of word embeddings that have similar representations for related words), and more powerful non-linear inference.

The current state-of-the-art neural machine translation approach (Bahdanau et al., 2015) consists of:

- an **encoder** stage where the input sentence is processed by two recurrent neural networks, one running left-to-right, the other right-to-left, resulting in hidden states for each word that encode it with its left and right context,
- a **decoder** stage where the output sentence is produced left-to-right, by conditioning on previous output words in the form of a hidden state (roughly corresponding to a language model in traditional statistical machine translation) and on the input encoding (roughly corresponding to a translation model), and
- an **attention mechanism** that conditions the prediction of each output word on a distribution over input words (roughly corresponding to an alignment function).

We describe a fairly general neural machine translation approach in order to motivate its use in the interactive translation prediction setting, illustrated in Figure 2. For more details, see Bahdanau et al. (2015), whose notation this section follows, or Edinburgh's WMT 2016 submission (Sennrich et al., 2016), whose system we use in our experiments.

[2] https://lilt.com/




Target words $y_t$ are emitted from hidden states $s_t$.

The hidden state $s_t$ is informed by the input sequence, weighted by an attention mechanism $\alpha_{t,1}, ..., \alpha_{t,T}$.

The source language sequence $x_1, ..., x_T$ is encoded as the hidden states $h$ of two recurrent neural networks.

Figure 2: Neural MT model used in this paper (figure from Bahdanau et al. (2015)).

At each time step $t$, the standard decoder computes the conditional probability of generating a word $y_t$ given the input sentence $\vec{x}$. This is defined to be:

$$p(y_t|\{\hat{y}_1, \cdots, \hat{y}_{t-1}\}, \vec{x}) = g(\hat{y}_{t-1}, c_t, s_t) \quad (1)$$

where $g$ is a non-linearity, $\hat{y}_{t-1}$ is the word produced by the previous decoding step, $c_t$ is a context vector, and $s_t$ is the hidden state for time $t$.

During encoding, annotations $h_t$ were produced for each word $x_t$ in the input sentence $\vec{x} = (x_1, \cdots, x_T)$. These $h_t$ were produced by concatenating the forward and backward hidden states produced for each word by the forward and backward RNNs, respectively. The context vector $c_t$ in Equation 1 is a weighted average of the annotations. First, weights $\alpha_{tj} = \exp(e_{tj}) / \sum_{k=1}^{T} \exp(e_{tk})$ are computed, where $e_{tj} = a(s_{t-1}, h_j)$ can be thought of as an alignment model (parameterized as a neural network and jointly trained with the rest of the system). The weight $\alpha_{tj}$ can be interpreted as the probability that $y_t$ is aligned to $x_j$, resulting in soft alignments used by the system's attention mechanism to weight the focus of the context vector. The context vector is then computed as $c_t = \sum_{j=1}^{T} \alpha_{tj} h_j$.

As indicated above, decoding in this attention-based neural machine translation approach proceeds word by word. At each step of the decoding process, a probability distribution over possible next words is computed. This is conditioned on the previous word, the context vector, and the hidden state. The highest scoring word is selected and used in the conditioning context for the next step. Alternatively, similar to beam search in traditional statistical machine translation decoding, the top $n$ next words may be considered and competing hypotheses with different output words maintained. Each of the hypotheses (consisting of a word sequence and a hidden state, and ranked by the combined word translation probabilities) is extended at the next decoding step.

There are various choices for the exact design of the recurrent neural networks used in the encoder and decoder. Going beyond basic approaches that use a simple hidden layer, more complex designs such as long short term memory (LSTM) cells or gated recurrent units (GRU) may be employed.

## 3 Neural Interactive Translation Prediction

The decoding process for neural translation models points to a straightforward implementation of interactive translation prediction. Instead of using the model's predictions in the conditioning context for the next step, the words in the prefix provided by the translator can be used. Hence,

the next word prediction is conditioned on the choice of the translator, rather than the prediction of the model.

During decoding for translation (as described above), the model's predictions $\{\hat{y}_1, \cdots, \hat{y}_{t-1}\}$ are fed back into the model to produce the next predicted word. In order to do interactive prediction, we instead feed the true prefix $\{y^*_1, \cdots, y^*_{t-1}\}$ produced by the translator back into the model. Thus we redefine the conditional probability of generating a word $y_t$ to be:

$$p(y_t|\{y^*_1, \cdots, y^*_{t-1}\}, \vec{x}) = g(y^*_{t-1}, c_t, s_t) \tag{2}$$

In this work, we present two variations on neural interactive translation prediction:

- The **no beam search** method produces the single best hypothesis for each new word, given the prefix provided by the translator, which is fed into the model during decoding (as described above).

- The **beam search** method conducts beam search and selects the most probable full translation of the sentence. If and when the translator diverges from this full translation, a new beam search is conducted from the translator-generated prefix through to the end of the sentence. We show results for beam size 12, but note that a beam size of 2 provides most of the improvement (a similar observation was made by Sutskever et al. (2014) with respect to standard MT evaluation).

While beam search is known to produce better BLEU scores than models without beam search, it is also more computationally expensive. We demonstrate that it performs well on the interactive translation prediction task, but note that full beam search is too slow for use in a live system.

Passing the translator prefix into the system (when the translator diverges from the predicted sequence) may produce subsequent errors in the translation, for instance by causing the attention mechanism to be misfocused. We show that the system is often able to recover from these errors, but that it occasionally results in incoherent sequences of suggestions. However, we show that the sequences of rejected suggestions produced by the neural systems tend to be shorter than those produced by the traditional search graph based systems.

## 4 Experimental Setup

Since a user study that collects sufficient data for the various settings of our methods would be too costly, we resort to a simulation where a preexisting human translation is used in place of the translator's input to the interactive translation system. We do this by treating the preexisting human translation as though it is being typed live, one word (or letter) at a time, by a translator interacting with our prediction system. While we expect that in practical use, the human translator may match the machine translation's suggestions more closely (for example, by accepting synonyms which we score as "wrong" as they are not exact matches), we can nevertheless compare methods based on their prediction accuracy against the human translation relative to one another.

**Data** Our experiment is carried out on the German–English data sets[3] made available for the shared news translation task of the Conference for Machine Translation (WMT). The data consists of a 115 million word parallel corpus (Europarl, News Commentary, CommonCrawl),

[3] http://www.statmt.org/wmt16/

| System | Configuration | BLEU |
|---|---|---|
| **Neural** | **no beam search** | **34.5** |
| | beam size 12 | 36.2 |
| | + ensemble | 37.5 |
| | + r2l reranking | 38.6 |
| **Phrase-based** | | **34.5** |

Table 1: Quality measured by BLEU scores (case-sensitive) of the systems used in this paper on the WMT 2016 news test set (German-English).

and about 75 billion words of additional English monolingual data (LDC Gigaword, monolingual news, monolingual CommonCrawl). We use the official 2999 sentence test set (average sentence length 23 tokens) to measure the accuracy of our methods.

**Neural Translation Model** The neural machine translation model uses Nematus[4], a fork of the DL4MT toolkit[5] by Cho (2015). It was trained on all the available parallel data and a similar amount of synthetic parallel data that was generated by translating part of the monolingual news data into German (Sennrich et al., 2015a). It uses byte pair encoding (Sennrich et al., 2015b) for a vocabulary of 90,000 words. Training this model on GPU takes about three weeks. We use the publicly available model[6] that matches the training settings of Edinburgh's WMT submission (Sennrich et al., 2016).

**Phrase-Based Model** The phrase based model, which we use as a baseline to produce search graph based predictions, uses all available parallel and monolingual training data. The system matches Johns Hopkins University's submission to the WMT shared task (Ding et al., 2016). This version does not use the byte pair encodings used by the neural models.

**System Quality** Since we are concerned with the translation speed, we consider a few simplifications of the neural translation model. We do not use ensemble decoding ("ensemble") or a reranking stage ("r2l reranking")[7]. Each of these simplifications makes decoding several times faster at a cost to quality of 1-2 BLEU points. See Table 1 for a comparison of quality scores for the different settings. Without beam search, the neural system used has the same BLEU score as the phrase-based system. This has the nice advantage that we are comparing methods based on systems of similar quality. It also shows the potential for improvement if computational concerns are removed. For a longer discussion of the speed of the methods, see Section 6.

## 5 Results

See Figure 3 for an example of the neural interactive translation prediction model's output for one sentence. The figure displays the correct word choices (taken from the reference translation), the model's prediction (using the prefix of the reference translation as conditioning context), and the most probable word choices according to the model's probability distribution.

Some of the failures are near-synonyms (*numbers* instead of *levels*, or *increasing* instead of *rising*) that we might expect would be accepted by real human users of the system. For the purposes of our evaluation, we count even these near-synonyms as incorrect (as they are not exact string matches). However, it is worth noting the prevalence of these near-synonyms: in the neural versions, we find that 21.0% of incorrect predictions (22.4% with beam search) are

[4] `https://github.com/rsennrich/nematus/`
[5] `https://github.com/nyu-dl/dl4mt-tutorial/`
[6] `https://github.com/rsennrich/wmt16-scripts/`
[7] For more on these methods, please see Sennrich et al. (2016)

**Input:** *Das Unternehmen sagte, dass es in diesem Monat mit Bewerbungsgesprächen beginnen wird und die Mitarbeiterzahl von Oktober bis Dezember steigt.*

| | Correct | Prediction | Prediction probability distribution |
|---|---|---|---|
| ✓ | the | the | **the (99.2)** |
| ✓ | company | company | **company (90.9)**, firm (7.6) |
| ✓ | said | said | **said (98.9)** |
| ✓ | it | it | **it (42.6)**, this (14.0), that (13.1), job (2.0), the (1.7), ... |
| ✓ | will | will | **will (77.5)**, is (4.5), started (2.5), 's (2.0), starts (1.8), ... |
| ✓ | start | start | **start (49.6)**, begin (46.7) |
| | inter@@ | job | job (16.1), application (6.1), en@@ (5.2), out (4.8), ... |
| ✘ | viewing | state | state (32.4), related (5.8), **viewing (3.4)**, min@@ (2.0), ... |
| ✘ | applicants | talks | talks (61.6), interviews (6.4), discussions (6.2), ... |
| ✓ | this | this | **this (88.1)**, so (1.9), later (1.8), that (1.1) |
| ✓ | month | month | **month (99.4)** |
| ✘ | , | and | and (90.8), **, (7.7)** |
| ✘ | with | and | and (42.6), increasing (24.5), rising (6.3), **with (5.1)**, ... |
| ✓ | staff | staff | **staff (22.8)**, the (19.5), employees (6.3), employee (5.0), ... |
| ✘ | levels | numbers | numbers (69.0), **levels (3.3)**, increasing (3.2), ... |
| ✘ | rising | increasing | increasing (40.1), **rising (35.3)**, climbing (4.4), rise (3.4), ... |
| ✓ | from | from | **from (97.4)** |
| ✓ | October | October | **October (81.3)**, Oc@@ (12.8), oc@@ (2.9), Oct (1.2) |
| ✘ | through | to | to (73.2), **through (15.6)**, until (8.7) |
| ✓ | December | December | **December (85.6)**, Dec (8.0), to (5.1) |
| ✓ | . | . | **. (97.5)** |

Figure 3: Example with about average prediction accuracy. Note the good recovery from failure and that several of the correct choices rank highly in the probability distribution of predicted words (values in parentheses indicate percent of probability mass assigned to words; only words with $\geq$1% shown). Tokens containing @@ are an artifact of byte pair encoding.

synonyms[8] of the correct answer (in the phrase-based system, this drops to 17.7%). Were these to be accepted by a real translator, overall system accuracy scores would improve.

The neural method copes well with failure, and typically resumes with plausible predictions. One exception is the prediction of *talks* after having seen ... *will start interviewing*. This may be due to the attention mechanism being thrown off after a sequence of low-probability prefix words.

### 5.1 Word Prediction Accuracy

Figure 3 also illustrates the evaluation metric we use to measure the quality of the prediction methods. It measures how many words are predicted correctly (see the first column in the figure). Note that we measure on the level of tokens, so we score the split word *inter@@ viewing* (an artifact of byte pair encoding) as a single token, rather than two tokens.

Table 2 shows the prediction accuracy for the three methods. The neural systems clearly outperform the method based on the search graph of the phrase-based model (with over 60% prediction accuracy for the neural systems and just 43.3% for the phrase-based). We discuss more reasons for this improvement in Section 5.3.

[8]Here we define words to be synonyms if their Wu-Palmer similarity (Wu and Palmer, 1994) in WordNet (Fellbaum, 1998) is equal to 1.

| System | Configuration | Word Prediction Accuracy |
|---|---|---|
| Neural | no beam search | 61.6% |
| | beam size 12 | 63.6% |
| Phrase-based | - | 43.3% |

Table 2: **Word prediction accuracy:** Ratio of words predicted by the interactive translation prediction system that matched the human reference translation exactly.

| System | Configuration | Letter Prediction Accuracy |
|---|---|---|
| Neural | no beam search | 86.8% |
| | beam size 12 | 87.4% |
| Phrase-based | - | 72.8% |

Table 3: **Letter prediction accuracy:** Ratio of letters predicted correctly.

### 5.2 Letter Prediction Accuracy

A useful feature of interactive translation prediction is the ability to react to single key strokes of the user. Consider an example from Figure 3. The system has a preference for *numbers* over *levels*. The latter is preferred by the human translator, hence the system fails to make the right prediction. If the user types the letter *l*, the system should quickly update its prediction to *levels*, the most likely word (from the probability distribution that generated the original hypothesis) that starts with this letter. In general, when the user types the initial letters of a word, the system should predict the most probable word with this letter prefix. In the beam search setting, the system first runs through the first word of each of the hypotheses in the beam (from most to least probable) to see if any match the the translator's letter prefix, before falling back to the probability distribution over the full vocabulary. With a beam size of 12, the correct word appears in the beam (but not as the predicted word) 25.2% of the time.

To measure the accuracy of system predictions for word completion, we count the number of incorrectly-predicted characters. To give a more complex example, suppose that the human translator wants to use the word *increased*. The system first predicts *rising*, but after seeing the letter *i*, it updates its prediction to *increasing*. It predicts all letters correctly until it comes to the final *e*. When the user enters *increase*, the system updates its prediction to *increased*. We count this as two wrongly predicted letters: **i***ncreas***e***d*. Table 3 shows the scores for our methods. Again, the neural methods clearly outperform the phrase-based method.

Note that this measure is not as clearly tied to user actions as word prediction accuracy. In the user interface shown in Figure 1, correctly predicted words are accepted by pressing TAB, while incorrectly predicted words have to typed in completely (assuming no word completion). So, the percentage of correctly predicted words reflects the ratio of words that do not have to be typed in. The effort savings for word completion are less clear, since there are various ways the user could interact with the system. In our example, when the user sees the prediction *increasing* but wants to produce *increase*, there are several choices even within the CASMACAT user interface. The user could accept the system's suggestion, and then delete the suffix *ing* and type in *ed*. Or, she could type in the entire prefix *increase* until the system makes the correct prediction — which in this example does not yield any savings at all: the user may accept the prediction with TAB or type in *d* herself.

| System | Configuration | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| Neural | no beam search | 55.9% | 61.8% | 61.3% | 62.2% | 61.1% |
| | beam size 12 | 58.0% | 62.9% | 62.8% | 64.0% | 61.5% |
| Phrase-based | - | 28.6% | 45.5% | 46.9% | 47.4% | 48.4% |

Table 4: Ratio of words correct after first failure.

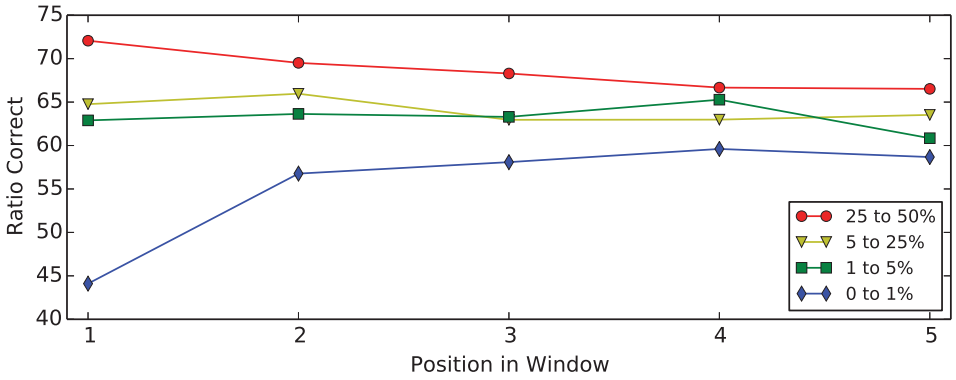


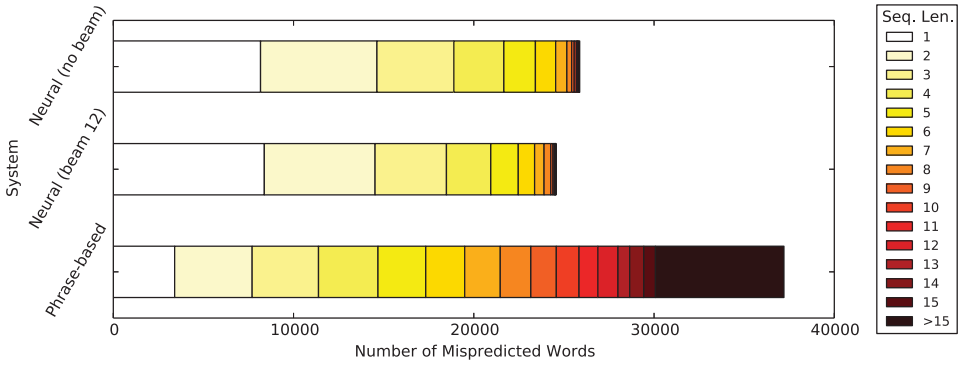

Figure 4: Neural (no beam) recovery from first failure at each position in the window of 5 words following the first failure, binned by probability assigned to correct solution (see legend).

### 5.3 Recovering from Failure

To get a more detailed picture of the performance of the neural prediction method, we explore how it recovers from failure. First, how well does the method predict the words following its first failure? We look at a window of up to five words following the first failure in a sentence (note that if the first failure is near the end of the sentence, the window will be truncated at the end of the sentence). See Table 4 for numbers for the methods.

The neural system predicts the first word in the window correctly 55.9% of the time, after it fails on a word. The second word in the window is predicted correctly 61.8% with similar numbers for the following words. So, failing on a word does impact the prediction of the word immediately following the failure, but only slightly words further down the sentence. The phrase-based method only correctly predicts 28.6% of the first words immediately after failing on a word. This suggests that the phrase-based method has a harder time recovering initially.

Interestingly, not all failures have an equal impact on the predictability of the subsequent words. Figure 4 shows the prediction accuracy for the neural method (without beam search) in more detail for the same five word window following the system's first error. We examine the way that the probability assigned to the correct word (which the model failed to predict) influences recovery from errors. When the model assigns extremely low probability (below 1%) to the correct answer, it performs very poorly on the next word, getting it correct only 44.1% of the time. On the other hand, when the model assigns relatively high probability to the correct word (25% to 50%), the probability of correctly guessing the next word rises to 72.1%. We can think of the probability assigned to the correct word as approximating how close the model was to being correct when it made the first error. When its prediction is far from correct, it has difficulty recovering, but when it is close to correct, it does not suffer a drop in performance in predicting the next words.




| System | Config. | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10+ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Neural | no beam | 8168 | 3229 | 1422 | 694 | 350 | 187 | 89 | 33 | 16 | 25 |
| | beam 12 | 8378 | 3072 | 1320 | 615 | 305 | 151 | 75 | 46 | 14 | 15 |
| Phrase | - | 3403 | 2150 | 1227 | 825 | 530 | 360 | 282 | 212 | 157 | 774 |

Figure 5: The graph shows number of mispredicted *words*, categorized by lengths of the sequence of mispredicted words to which they belong. The table gives a breakdown of the number of *sequences* of each length.

We observe examples of this phenomenon (and its ties to near-synonyms) in Figure 3. When the model assigns low probability to the correct answer (e.g. *interviewing*), there are sequences of incorrect predictions. In the case of *rising*, the model predicts *increasing*, a near-synonym, and assigns the highest probability to *increasing*, *rising*, and *climbing* (in descending order). As we saw in Section 5.2, the neural system predicts more synonyms than the search graph system. We hypothesize that this is partly due to the neural system having additional information about the semantics of words (as represented by their embeddings), while the search graph system treats synonyms and non-synonyms alike.

### 5.4 Length of Sequences of Mispredicted Words

Another revealing set of statistics is the length of sequences of word prediction failures. If the method fails on one word, and predicts the next word correctly, we have a 1-word failure sequence. However, if it misses the next word also and only recovers after that, we have a 2-word failure sequence, and so on. Shorter failure sequences indicate better models (and a better user experience). Figure 5 visualizes the sequences of word prediction failures by showing how many mispredicted words can be accounted for by each failure sequence length (mispredictions in shorter sequences are represented by light colors while mispredictions in long sequences are shown in darker red).

The methods show a stark contrast. The neural methods have a much higher number of 1-word failure sequences (8168 and 8378 vs. 3403) and 2-word failure sequences (3229 and 3072 vs. 2150, comprised of 6458 and 6144 vs. 4300 mispredicted words) but comparably very few long failure sequences. For instance, only a small fraction of the neural systems' mispredicted words occur in sequences of greater than 15 errors in a row (93 or 16 words total), while 7129 of the phrase-based system's word prediction errors occur in misprediction sequences of length greater than 15 words. This is not simply a consequence of the greater word prediction accuracy of the neural systems; in particular, the phrased based model shows far more long misprediction sequences than one would expect were those errors distributed uniformly randomly (the neural systems also have more long misprediction sequences than would be expected if the errors

| **Length** | 1-4 | 5-9 | 10-14 | 15-19 | 20-24 | 25-29 | 30-34 | 35-39 | 100-104 |
|---|---|---|---|---|---|---|---|---|---|
| CPU | 108.6 | 115.7 | 122.7 | 127.0 | 131.3 | 136.1 | 140.7 | 145.2 | 184.4 |
| GPU | 7.0 | 7.2 | 7.4 | 7.4 | 7.4 | 7.4 | 7.6 | 7.6 | 7.6 |

Table 5: Decoding speed per word in milliseconds (neural model, no beam search) for different sentence lengths.

occurred randomly, but to a lesser extent).

These numbers suggest that the neural method recovers much better from failures, while the phrase-based system has more difficulty. Since the neural method considers every word in the vocabulary at any point in the sequence, it can always place the user's choice in a word prediction sequence, and does not have to resort to string edit distance to match up with the user's translation.

## 6 Speed Considerations

We have shown that the neural method delivers superior prediction accuracy, but is it fast enough? To be used in an interactive user interface, the method has to quickly produce alternative sentence completion predictions. A common time limit in human computer interaction is 100 milliseconds for the system's response. Any longer feels sluggish and annoying to the user.

### 6.1 Speed Measurements

In a basic setup, the neural machine translation decoder has to step through the user's prefix, and then produce predicted words until the end of the sentence. In other words, it has to translate the entire sentence for a response to a user interaction. Table 5 gives numbers for decoding speed, running on a multi-core CPU (32 core 3.20GHz Intel Xeon CPU E5-2667 v3, although only 2-3 cores are utilized on average) and a GPU (Tesla K80).

Decoding time is spent mostly on the matrix multiplications to compute hidden and output layer vectors. The computational cost of the argmax operation to pick the best word is negligible, hence the computational cost is essentially the same for matching words in the user prefix and predicting new words.

To predict a single word, the CPU requires over 100 milliseconds, which is clearly too slow. The time it takes to translate a single word slightly increases with the length of the sentence, since the attention mechanism has to sum over a larger context of source language words.

On a GPU, the cost to predict one word drops to 7 milliseconds. For a 20-word sentence, this means ($7 \times 20$) 140 milliseconds which is also beyond our 100 millisecond time limit.

### 6.2 Fast Recovery

However, we can employ the following optimizations:

- We can precompute the initial translation when the document is uploaded. This gives us the entire prediction sequence.
- The user will diverge from the suggested translation at some point. We do not need to match the accepted user prefix again, since we can use the cached prediction sequence up to this point.
- We can produce a limited sequence of predicted words rather than a completely new full translation. In the user interface illustrated in Figure 1, only three words are shown. We may initially respond to the user interface with a small sequence of words and deliver the remaining words later.

- To have a full sentence completion at any time, we may initially patch together a limited prediction with the original sentence completion, since later words are less likely to be influenced by the local user choice.

| | |
|---|---|
| (1) Initial hypothesis | *A sovereign prel@@ ate of the Champions League season .* |
| (2) Translator prefix | A confident |
| (3) New prediction (3 words) | `start to the` |
| (4) Alignment | `start to the` → *prel@@ ate of* ***the*** *Champions* |
| (5) New hypothesis | A confident `start to the` *Champions League season .* |

Figure 6: Example of patching a 3-word prediction into the original sentence completion.

We propose a method (without beam search) that patches together a limited (say, 3 word) new prediction with the existing sentence completion each time that the translator diverges from the predicted translation. An example of this is shown in Figure 6 (we reference the row numbers parenthetically in the following description). We begin by precomputing a full translation when the document is uploaded (row 1). If and when the translator diverges from this (row 2), we compute predictions for the next 3 words only (row 3) and attempt to patch them together with the original translation.

We find the patch position by computing the KL divergence between the probability distribution that produced the last of the 3 new words and the stored probability distributions that produced the words in a 5-word window (following the position of the the last word in the translator prefix). This results in an alignment between the last of the 3 new words and the index of some word in the existing translation (row 4). The new translation hypothesis consists of concatenating the translator prefix, the 3 newly predicted words, and any words following the position of the index in the existing translation hypothesis that minimized the KL divergence (row 5).

By patching together earlier predictions with a short sequence of predictions based on new input from the translator, we can guarantee that we can serve the translator new predictions quickly. The new prediction and patching combined takes an average of 54.3 milliseconds to compute. This approach yields a word prediction accuracy of 56.4% and a letter prediction accuracy of 84.2% (a drop from the full search neural model by 5.2% and 2.4%, respectively, but still vastly outperforming the phrasal search graph system).

In a real-life setting, we may sometimes have enough time to recompute the full sentence in the background, rather than relying on patching together different predictions, so we could expect performance closer to the performance noted earlier. Additionally, we could use beam search (or other improvements to the neural model) in order to precompute better initial sequences, which we expect would also improve performance.

### 6.3 Analysis

In the example in Figure 6, the new hypothesis is in fact the correct translation. If the initial error by the system is a single-token error (for example a synonym), we might expect the last of the 3 newly translated words to align to the word at the center of the window. In this case it (correctly) aligns one position to the right of this and produces the desired hypothesis.

When the alignment is close to the center of the window, this suggests that the sentence does not require much reordering. The patching heuristic is somewhat imprecise and has difficulty handling sentences with long-range reordering. In Figure 7 we compute the failure

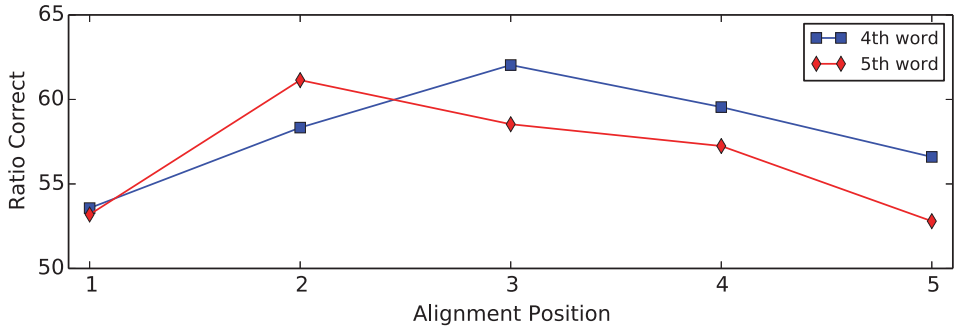


Figure 7: Ratio correct after first failure for the 4th and 5th words in the window.

recovery ratios for the 4th and 5th words in the window following the first error,[9] conditioned on the alignment position.

An alignment position of 3 indicates that the 3rd newly translated word aligned with the 3rd word in the window (as would be expected if no reordering were needed). Similarly, an alignment position of 1 indicates that the 3rd newly translated word aligned to the 1st word following the failure, and so on. We see that when a longer-distance alignment occurs, the ratio drops, demonstrating either an error of alignment or the system's difficulty in handling long-distance reordering.

## 7 Conclusion

In this paper we demonstrate that neural machine translation systems can effectively be applied to interactive translation prediction, improving upon the performance of traditional methods. We show that they recover well from errors, have shorter sequences of incorrect predictions, and produce more near-synonyms of the correct answers. Finally, we demonstrate a method that allows for practical deployment given real-life time constraints.

## Acknowledgments

This work was supported, in part, by the Human Language Technology Center of Excellence (HLTCOE) through the 2016 SCALE workshop CADET, as well as by a National Science Foundation Graduate Research Fellowship under Grant No. DGE-1232825 (to the first author). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsors.

## References

Alabau, V., Sanchis, A., and Casacuberta, F. (2011). Improving on-line handwritten recognition using translation models in multimodal interactive machine translation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Techologies*, pages 389–394, Portland, Oregon, USA. Association for Computational Linguistics.

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *ICLR*.

Barrachina, S., Bender, O., Casacuberta, F., Civera, J., Cubel, E., Khadivi, S., Lagarda, A., Ney, H.,

[9] We show only performance for the 4th and 5th words in the window; performance on the first three is identical to the no-beam-search values reported in Table 4, as the patching occurs after this sequence of 3 new predictions.

Toms, J., Vidal, E., and Vilar, J.-M. (2009). Statistical approaches to computer-assisted translation. *Computational Linguistics*, 35(1).

Bender, O., Hasan, S., Vilar, D., Zens, R., and Ney, H. (2005). Comparison of generation strategies for interactive machine translation. In *Proceedings of the 10th Conference of the European Association for Machine Translation (EAMT)*, Budapest.

Cho, K. (2015). Neural machine translation tutorial. Technical report.

Cubel, E., Khadivi, S., Lagarda, A., Ney, H., Toms, J., Vidal, E., and Vilar, J.-M. (2009). Statistical approaches to computer-assisted translation. *Computational Linguistics*, 35(1).

Ding, S., Duh, K., Khayrallah, H., Koehn, P., and Post, M. (2016). The JHU machine translation systems for WMT 2016. In *Proceedings of the First Conference on Machine Translation (WMT)*.

Fellbaum, C., editor (1998). *WordNet: An Electronic Lexical Database*. MIT Press.

Foster, G., Langlais, P., and Lapalme, G. (2002). User-friendly text prediction for translators. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 148–155, Philadelphia. Association for Computational Linguistics.

Green, S., Wang, S. I., Chuang, J., Heer, J., Schuster, S., and Manning, C. D. (2014). Human effort and machine learnability in computer aided translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1225–1236, Doha, Qatar. Association for Computational Linguistics.

Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95.

Jean, S., Firat, O., Cho, K., Memisevic, R., and Bengio, Y. (2015). Montreal neural machine translation systems for wmt15. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 134–140, Lisbon, Portugal. Association for Computational Linguistics.

Koehn, P. (2009). A process study of computer-aided translation. *Machine Translation*, 23(4):241–263.

Koehn, P., Tsoukala, C., and Saint-Amand, H. (2014). Refinements to interactive translation prediction based on search graphs. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 574–578, Baltimore, Maryland. Association for Computational Linguistics.

Langlais, P., Foster, G., and Lapalme, G. (2000). Transtype: a computer-aided translation typing system. In *Proceedings of the ANLP-NAACL 2000 Workshop on Embedded Machine Translation Systems*.

Och, F. J., Zens, R., and Ney, H. (2003). Efficient search for interactive statistical machine translation. In *Proceedings of Meeting of the European Chapter of the Association of Computational Linguistics (EACL)*.

Sanchis-Trilles, G., Ortiz-Martínez, D., Civera, J., Casacuberta, F., Vidal, E., and Hoang, H. (2008). Improving interactive machine translation via mouse actions. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 485–494, Honolulu, Hawaii. Association for Computational Linguistics.

Sennrich, R., Haddow, B., and Birch, A. (2015a). Improving neural machine translation models with monolingual data. *CoRR*, abs/1511.06709.

Sennrich, R., Haddow, B., and Birch, A. (2015b). Neural machine translation of rare words with subword units. *CoRR*, abs/1508.07909.

Sennrich, R., Haddow, B., and Birch, A. (2016). Edinburgh neural machine translation systems for WMT 16. In *Proceedings of the First Conference on Machine Translation (WMT)*.

Sutskever, I., Vinyals, O., and Le, Q. V. V. (2014). Sequence to sequence learning with neural networks. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K., editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Wu, Z. and Palmer, M. (1994). Verbs semantics and lexical selection. In *Proceedings of the 32nd Annual Meeting on Association for Computational Linguistics*, pages 133–138, Stroudsburg, PA, USA. Association for Computational Linguistics.

Wuebker, J., Green, S., DeNero, J., Hasan, S., and Luong, M.-T. (2016). Models and inference for prefix-constrained machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Berlin, Germany. Association for Computational Linguistics.

# Guided Alignment Training for Topic-Aware Neural Machine Translation

**Wenhu Chen** hustchenwenhu@gmail.com
RWTH Aachen University, Ahornstr. 55, Aachen, Germany

**Evgeny Matusov** ematusov@ebay.com
eBay, Inc. Kasernenstr. 25, Aachen, Germany

**Shahram Khadivi** skhadivi@ebay.com
eBay, Inc. Kasernenstr. 25, Aachen, Germany

**Jan-Thorsten Peter** peter@cs.rwth-aachen.de
RWTH Aachen University, Ahornstr. 55 Aachen, Germany

**Abstract**

In this paper, we propose an effective way for biasing the attention mechanism of a sequence-to-sequence neural machine translation (NMT) model towards the well-studied statistical word alignment models. We show that our novel guided alignment training approach improves translation quality on real-life e-commerce texts consisting of product titles and descriptions, overcoming the problems posed by many unknown words and a large type/token ratio. We also show that meta-data associated with input texts such as topic or category information can significantly improve translation quality when used as an additional signal to the decoder part of the network. With both novel features, the BLEU score of the NMT system on a product title set improves from 18.6 to 21.3%. Even larger MT quality gains are obtained through domain adaptation of a general domain NMT system to e-commerce data. The developed NMT system also performs well on the IWSLT speech translation task, where an ensemble of four variant systems outperforms the phrase-based baseline by 2.1% BLEU absolute.

## 1 Introduction

NMT systems were shown to reach state-of-the-art translation quality on tasks established in MT research community such as IWSLT speech translation task Cettolo et al. (2012). In this paper, we also apply NMT approach to e-commerce data: user-generated product titles and descriptions for items put on sale. Such data are very different from newswire and other texts typically considered in the MT research community. Titles in particular are short (usually fewer than 15 words), contain many brand names which often do not have to be translated, but also product feature values and specific abbreviations and jargon. Also, the vocabulary size is very large due to the large variety of product types, and many words are observed in the training data only once. At the same time, these data are provided with additional meta-information about the item (e.g. product category such as clothing or electronics), which can be used as context to perform topic/domain adaptation for improved translation quality.

At first glance, established phrase-based statistical MT approaches are well-suited for e-commerce data translation. In a phrase-based approach, singleton, but unambiguous words and phrases are usually translated correctly. Also, since the alignment between source and

target words is available, it is possible to transfer certain entities from the source sentence to the generated target sentence "in-context" without translating them. Such entities can include numbers, product specifications such as "5S" or brand names such as "Samsung" or "Lenovo". In training, these entities can be replaced with placeholders to reduce the vocabulary size.

However, NMT approaches are more powerful at capturing context beyond phrase boundaries and were shown to better exploit available training data. They also successfully adapt themselves to a domain, for which only a limited amount of parallel training data is available Luong and Manning (2015). Also, previous research Mathur et al. (2015) has shown that it is difficult to obtain translation quality improvements with topic adaptation in phrase-based SMT because of data sparseness and a large number of topics (e. g. corresponding to product categories), which may or may not be relevant for disambiguating between alternative translations or solving other known MT problems. In contrast, we expected NMT to better solve the topic adaptation problem by using the additional meta-information as an extra signal in the neural network. To the best of our knowledge, this is the first work where the additional information about the text topic is embedded into the vector space and used to directly influence NMT decisions.

In an NMT system, the attention mechanism introduced in Luong et al. (2014) is important both for decoding as well as for restoration of placeholder content and insertion of unknown words in the right positions in the target sentence. To improve the estimation of the soft alignment, we propose to use the Viterbi alignments of the IBM model 4 Brown et al. (1993) as an additional source of knowledge during NMT training. The additional alignment information helps the current system to bias the attention mechanism towards the Viterbi alignment.

This paper is structured as follows. After an overview of related NMT work in Section 2, we propose a novel approach in Section 3 on using statistical word alignemt to bias the training of neural MT attention mechanism, we call it *guided alignment training*. In Section 4, we describe in more detail how topic information can benefit NMT. Section 5 and Section 6 describes our domain adaptation approach. Experimental results are presented in Section 7. The paper is concluded with a discussion and outlook in Section 8.

## 2 Related Work

Neural machine translation is mainly based on using recurrent neural networks to grasp long term dependencies in natural language. An NMT system is trained on end-to-end basis to maximize the conditional probability of a correct translation given a source sentence Sutskever et al. (2014), Bahdanau et al. (2014), Cho et al. (2014b). When using attention mechanism, large vocabularies Jean et al. (2014), and some other techniques, NMT is reported to achieve comparable translation quality to state-of-art phrase-based translation systems. Most NMT approaches are based on the encoder-decoder architecture Cho et al. (2014a), in which the input sentence is first encoded into a fixed-length representation, from which the recurrent neural network decoder generates the sequence of target words. Since fixed-length representation cannot give enough information for decoding, a more sophisticated approach using attention mechanism is proposed by Bahdanau et al. (2014). In this approach, the neural network learns to attend to different parts of source sentence to improve translation quality. Since the source and target language vocabularies for a neural network have to be limited, the rare words problem deteriorates translation quality significantly. The rare word replacement technique using soft alignment proposed by Luong et al. (2014) gives a promising solution for the problem. Both encoder-decoder architecture and insertion of unknown words into NMT output highly rely on the quality of the attention mechanism, thus it becomes the crucial part of NMT. Some research has been done to refine it by Luong et al. (2015), who proposed global and local attention-based models, and Cohn et al. (2016), who used biases, fertility and symmetric bilingual structure to improve the

attention model mechanism. The most recent work done by Mi et al. (2016) is highly parallel with our *guided alignment training*, Section 3. They use statistical alignment to supervise the NMT in a similar fashion as we do, the difference is that they smooth the statistical alignment and apply Euclidean distance directly to the objective function, while we try with different divergence function and also re-weight it before adding to the overall objective function.

Research on topic adaptation most closely related to our work was performed by Hasler et al. (2014), but the features proposed there were added to the log-linear model of a phrase-based system. Here, we use the topic information as part of the input to the NMT system. Another difference is that we primarily work with human-labeled topics, whereas in Hasler et al. (2014) the topic distribution is inferred automatically from data.

When translating e-commerce content, we are faced with a situation when only a few product titles and descriptions were manually translated, resulting in a small in-domain parallel corpus, but a large general-domain parallel corpus is available. In such situations, domain adaption techniques have been used both in phrase-based systems Koehn and Schroeder (2007) and NMT Luong and Manning (2015). In addition, while diverse NMT models using different features and techniques are trained, an ensemble decoder can be used to combine them together to make a more robust model. This approach was used by Luong et al. (2015) to outperform the state-of-art phrase-based system with their NMT approach in the WMT 2015 evaluation.

## 3 Guided Alignment Training

When using the attention-based NMT Bahdanau et al. (2014), we observed that the attention mechanism sometimes fails to yield appropriate soft alignments, especially with increasing length of the input sentence and many out-of-vocabulary words or placeholders. In translation, this can lead to disordered output and word repetition.

In contrast to a statistical phrase-based system, the NMT decoder does not have explicit information about the candidates of the current word, so at each recurrent step, the attention weights only rely on the previously generated word and decoder/encoder state, as depicted in Figure 1. The target word itself is not used to compute its attention weights. If the previous word is an out-of-vocabulary (OOV) or a placeholder, then the information it provides for calculating the attention weights for the current word is neither sufficient nor reliable anymore. This leads to incorrect target word prediction, and the error propagates to the future steps due to feedback loop. The problem is even larger in the case of e-commerce data where the number of OOVs and placeholders is considerably higher.

To improve the estimation of the soft alignment, we propose to use the Viterbi alignments of the IBM model 4 as an additional source of knowledge during the NMT training. Therefore, we first extract Viterbi alignments using GIZA++ toolkit Och and Ney (2003), then we use them to bias the attention mechanism. Our approach is to optimize on both the decoder cost and the divergence between the attention weights and the alignment connections generated by statistical alignments. The multi-objective optimization task is then expressed as a single-objective function, which is a linear combination of two loss functions: original and new guided-alignment.

### 3.1 Decoder Cost

NMT proposed by Bahdanau et al. (2014) maximizes the conditional log-likelihood of target sentence $y_1, \ldots, y_T$ given the source sentence $x_1, \ldots, x'_T$:

$$H_D(y, x) = -\frac{1}{N} \sum_{n=1}^{N} \log p_\theta(y_n | x_n) \tag{1}$$

where $(y_n, x_n)$ refers to $n_{th}$ training sentence pair, and $N$ denotes the total number of sentence pairs in the training corpus. In the paper, we name the negative log-likelihood as decoder cost

to distinguish from alignment cost. When using encoder-decoder architecture by Cho et al. (2014b), the conditional probability can be written as:

$$p(y_1 \dots y_T | x_1 \dots x_{T'}) = \prod_{t=1}^{T} p(y_t | y_{t-1} \cdots y_1, c) \tag{2}$$

with $p(y_t|y_{t-1} \cdots y_1, c) = g(s_t, y_{t-1}, c)$, where $T$ is the length of the target sentence and $T'$ is the length of source sentence, $c$ is a fixed-length vector to encode source sentence, $s_t$ is a hidden state of RNN at time step $t$, and $g(\cdot)$ is a non-linear function to approximate word probability. If attention mechanism is used, the vector $c$ in each sentence is replaced by time-variant representation $c_t$ that is a weighted summary over a sequence of annotations $(h_1, \cdots, h_{T'})$, and $h_i$ contains information about the whole input sentence, but with a strong focus on the parts surrounding the $i_{th}$ word Bahdanau et al. (2014). Then, the context vector can be defined as:

$$c_t = \sum_{i}^{T'} \alpha_{ti} h_i \quad \text{where} \quad \alpha_{ti} = \frac{exp(e_{ti})}{\sum_{k=1}^{T'} exp(e_{tk})}. \tag{3}$$

This means, $\alpha_{ti}$ for each annotation $h_i$ is computed by normalizing the score function with the softmax. Also, $e_{ti} = a(s_{t-1}, h_i)$ is the function to calculate the score of $t$-th target word aligning to $i$-th word in the source sentence. The alignment model $a(\cdot, \cdot)$ is used to calculate similarity between previous state $s_{t-1}$ and bi-directional state $h_i$. In our experiments, we took the idea of the dot global attention model of Luong et al. (2015), but we still keep the order $h_{t-1} \rightarrow a_t \rightarrow c_t \rightarrow h_t$ as proposed by Bahdanau et al. (2014). We calculate the dot product of encoder state $h_i$ with the last decoder state $s_{t-1}$ instead of the current decoder state. We observe that this dot attention model (Equation 4) works better than concatenation in our experiments.

$$a(s_{t-1}, h_i) = (W_s s_{t-1})^T (W_h h_i) \tag{4}$$

### 3.2 Alignment Cost

We introduce alignment cost to penalize attention mechanism when it is not consistent with statistical word alignment. We represent the pre-trained statistical alignments by a matrix $A$, where $A_{ti}$ refers to the probability of the $t_{th}$ word in the target sentence of being aligned to the $i_{th}$ word in the source sentence. In case of multiple source words aligning to the same target word, we normalize to make sure $\sum_i A_{ti} = 1$, in the case of non-aligned target words, we do not add any penalty. In attention-based NMT, the matrix of attention weights $\alpha$ has the same shape and semantics as $A$. We propose to penalize NMT based on the divergence of the two matrices during the training, the divergence function can e. g. be cross entropy $G_{ce}$ or mean square error $G_{mse}$ as in Equation 5. As shown in Figure 1, $A$ comes from statistical alignment and is fed into our guided-alignment NMT as an additional input to penalize the attention mechanism.

$$G_{ce}(A, \alpha) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{i=1}^{T'} A_{ti} \log \alpha_{ti} \qquad G_{mse}(A, \alpha) = \frac{1}{T} \sum_{t=1}^{T} \sum_{i=1}^{T'} (A_{ti} - \alpha_{ti})^2 \tag{5}$$

We combine decoder cost and alignment cost to build the new loss function $H(y, x, A, \alpha)$:

$$H(y, x, A, \alpha) = w_1 G(A, \alpha) + w_2 H_D(y, x) \tag{6}$$

During training, we optimize the new compound loss function $H(y, x, A, \alpha)$ with regard to the same parameters as before. The guided-alignment training influences the attention mechanism to generate alignment closer to Viterbi alignment and has the advantage of unchanged parameter

space and model complexity. When training is done, we assume that NMT can generate robust alignment by itself, so there is no need to feed an alignment matrix as input during evaluation. As indicated in Equation 6, we set $w_1$ and $w_2$ for weights of decoder cost and alignment cost to balance their weight ratio. We performed further experiments (see section 7) to analyze the impact of different weight settings on translation quality.

## 4 Topic-aware Machine Translation

In the e-commerce domain, the information on the product category (e.g., "mens' clothing", "mobile phones", "kitchen appliances") often accompanies the product title and description and can be used as an additional source of information both in the training of a MT system and during translation. In particular, such meta-information can help to disambiguate between alternative translations of the same word that have different meaning. The choice of the right translation often depends on the category. For example, the word "skin" has to be translated differently in the categories "mobile phone accessories" and "make-up". Outside of the e-commerce world, similar topic information is available in the form of e.g. tags and keywords for a given document (on-line article, blog post, patent, etc.) and can also be used for word sense disambiguation and topic adaptation. In general, a document may belong to multiple topics.

Here, we propose to feed such meta-information into the recurrent neural network to help generate words which are appropriate given a particular category or topic.

### 4.1 Topic Representation

The idea is to represent topic information in a $D$-dimensional vector $l$, where $D$ is the number of topics. Since one sentence can belong to multiple topics (possibly with different probabilities/weights), we normalize the topic vector so that the sum of its elements is 1. It is fed into the decoder to influence the proposed target word distribution. The conditional probability given the topic membership vector can be written as (cf. Equations 2 and 3):

$$p(y_t|y_{<t-1}, c_t, s_{t-1}, l) = p(y_t|y_{t-1}, c_t, s_{t-1}, l) \approx g(y_{t-1}, s_{t-1}, c_t, l)$$

where $g(\cdot)$ is used to approximate the probability distribution. In our implementation, we introduce an intermediate readout layer to build the function $g(\cdot)$, which is a feed-forward network as depicted in Figure 2.

### 4.2 Topic-aware Decoder

In the NMT decoder, we feed the topic membership vector to the readout layer in each recurrent step to enhance word selection. As shown in Figure 1 and Figure 2, topic membership vector $l$ is fed into the NMT decoder as an additional input besides source and target sentences:

$$p(y_t|y_{<t-1}, c_t, s_{t-1}, l) = p(y_t|r_t) \quad \text{where} \quad r_t = W_r[c_t; f_{t-1}; s_{t-1}; l] + b_r \tag{7}$$

Here, $W_r$ is the concatenation of original transformation matrix and $l$, $r_t$ is the output from readout layer and $f_t$ is the embedding of the last target word $y_{t-1}$; $s_{t-1}$ refers the last decoder state. $W_r$ and $b_r$ are weights and bias for the linear transformation, respectively. We can rearrange the formula as:

$$\begin{aligned} r_t &= [W'_r, W_c][c_t; f_{t-1}; s_{t-1}; l] + b_r \\ &= [W'_r[c_t; f_{t-1}; s_{t-1}] + b_r] + W_c l \\ &= r'_t + E_c \end{aligned} \tag{8}$$

where $W_r$ is concatenation of original transformation matrix $W'_r$ and topic transformation matrix $W_c$. Then adding topic into readout layer input is equivalent to adding an additional topic

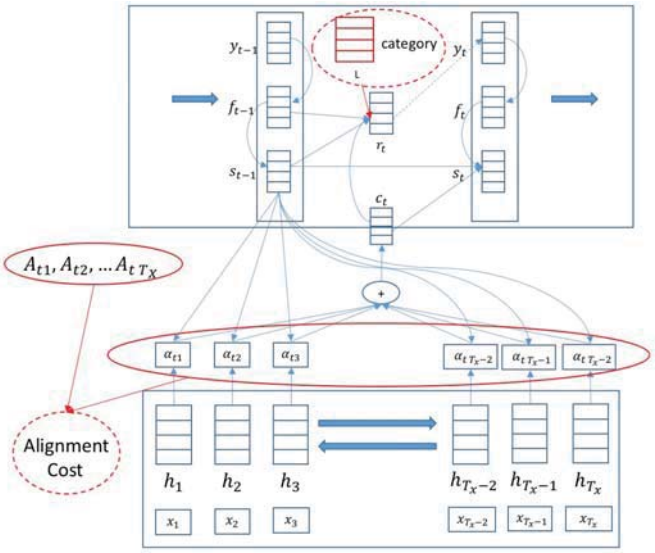



Figure 1: Topic-aware, alignment-guided encoder-decoder model. Topic information $l$ is added to the decoder as an additional input, influencing every decoding step; statistical alignment $A$ is added to the attention layer to supervise the learning of the attention mechanism.

vector $E_c$ into the original readout layer output. Assuming $l$ is a one-hot category vector, then $W_c l$ is equivalent to retrieving a specific column from the matrix $W_c$. Hence, we can name this additional vector $E_c$ as topic embedding, regarded as a vector representation of topic information. It is quite similar to word embedding by Mikolov et al. (2013), we will further analyze the similarity between different topics in Figure 3.

The readout layer depicted in Figure 2 merges information from the last state $s_{t-1}$, previous word embedding $f_{t-1}$ (coming from word index $y_{t-1}$, which is sampled w.r.t. the proposed word distribution), as well as the current context $c_t$ to generate output. It can be seen as a shallow network, which consists of a max-out layer Goodfellow et al. (2013), a fully-connected layer, and a softmax layer.

## 5 Bootstrapping

When trained on small amounts of data, the attention-based neural network approach does not always produce reliable soft alignment. The problem gets worse when the sentence pairs available for training are getting longer. To solve this problem, we extracted bilingual sub-sentence units from existing sentence pairs to be used as additional training data. These units are exclusively aligned to each other, i. e. all words within the source sub-sentence are aligned only to the words within the corresponding target sub-sentence and vice versa. The alignment is determined with the standard approach (IBM Model 4 alignment trained with the GIZA++ toolkit Och and Ney (2003)). As boundaries for sub-sentence units, we used punctuation marks, including period, comma, semicolon, colon, dash, etc. To simplify bilingual sentence splitting, we used the standard phrase pair extraction algorithm for phrase-based SMT, but set the minimum/maximum source phrase length to 8 and 30 tokens, respectively. From all such long phrase pairs extracted by the algorithm, we only kept those which are started or ended with a punctuation mark or started/ended a sentence; both on the source and on the target side.

For the bootstrapped training, we merged the original training data with the extracted sub-

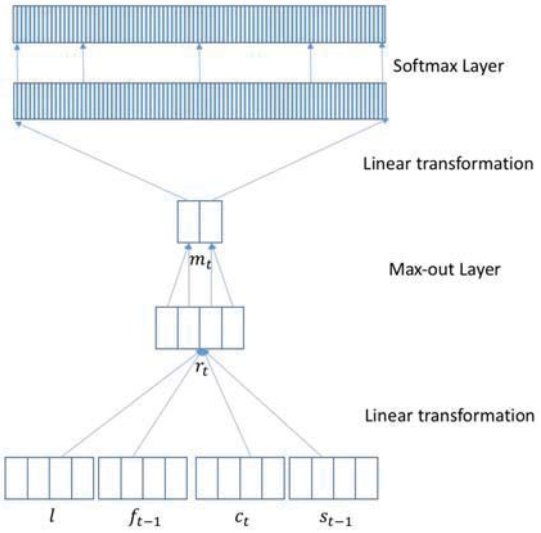



Figure 2: Topic-aware readout layer. The topic information vector, $l$, is fed to the readout layer in each recurrent step to influence target word selection.

sentence units and ran the neural training algorithm on this extended training set. Since the extracted bilingual sub-sentence units generally showed good correspondence between source and target due to the constraints described above, the expectation was that having such units repeated in the training data as stand-alone training instances would guide the attention mechanism to become more robust and make it easier for the neural training algorithm to find better correspondences between more difficult source/target sentence parts. Also, having both short and long training instances was expected to make neural translation quality less dependent on the input length.

## 6 E-commerce Domain Adaptation

For the e-commerce English-to-French translation task, we only have a limited amount of in-domain parallel training data (item titles and descriptions). To benefit from large amounts of general-domain training data, we follow the method described by Luong and Manning (2015). We first train a baseline NMT model on English-French WMT data (common-crawl, europarl v7, and news commentary corpora) for two epochs to get the best result on a development set, and then we continue training the same model on the in-domain training set for a few more epochs. In contrast to Luong and Manning (2015), however, we use the vocabularies of the most frequent 52K source/target words in the in-domain data (instead of the out-of-domain data vocabularies). This causes NMT to focus on translation of the most relevant in-domain words.

## 7 Experimental Results

### 7.1 Data Sets and Preprocessing

We performed MT experiments on the German-to-English IWLST 2015 speech translation task Cettolo et al. (2012) and on an in-house English-to-French e-commerce translation task. As part of data preprocessing, we tokenized and lowercased the corpora, as well as replaced numbers, product specifications, and other special symbols with placeholders such as $num. We only keep these placeholders in training, but preserve their content as XML markups in the dev/test sets, which we try to restore using attention mechanism. This content is inserted for the

generated placeholders on the target side based on the attention mechanism (see Luong et al. (2014)). In the beam search for the best translation, we make sure that each placeholder content is used only once. Using the same mechanism, we also pass OOV words to the target side "as is" (without using any special unknown word symbol).

On both tasks, we evaluate all systems and system variants using case-insensitive BLEU Papineni et al. (2002) and TER Snover et al. (2006) scores on held-out development and test data using a single human reference translation.

| Data-set | | IWSLT | | e-commerce | |
|---|---|---|---|---|---|
| Language | | German | English | English | French |
| Training | Sentences | 165 201 | | 516 000 | |
| | Running words | 3 873 816 | 3 656 038 | 2 592 202 | 2 895 089 |
| | Full vocabulary | 103 390 | 45 068 | 119 607 | 129 848 |
| Dev | Sentences | 567 | | 910 | |
| | Running words | 9 812 | 10 695 | 10 339 | 11 283 |
| Test | Sentences | 1100 | | 910 | |
| | Running words | 19 019 | 22 895 | 10 817 | 11 016 |
| Source OOV rate w.r.t. full/NMT Voc. | | 5.16/6.12 % | | 2.56/5.76 % | |

Table 1: Corpus statistics for the IWSLT and e-commerce translation tasks. OOV rate is calculated after preprocessing, placeholders like $num, $url, etc. largely decrease the OOV rate in the e-commerce dev and test sets.

#### 7.1.1 IWSLT TED Talk Data

For the IWSLT German-to-English task (translation of transcribed TED talks), we map the topic keywords of each TED talk in the 2015 training/dev/test evaluation campaign release to ten general topics such as politics, environment, education, and others. All sentences in the same talk share the same topic, and one talk can belong to several topics. Instead of using the official IWSLT dev/test data, we set aside 81/159 talks for development/test set, respectively. Out of these talks, we used 567 dev and 1100 test sentences which have the highest probability of relating to a particular topic (bag-of-words classification using the remaining 1365 talks as the training data). The corpus statistics of the data sets obtained this way are given in Table 1[1].

#### 7.1.2 E-commerce Data

For the e-commerce English-to-French task, we used the product category such as "fashion" or "electronics" as topic information (a total of 80 most widely used categories plus the category "other" that combined all the less frequent categories). The training set contained both product titles and product descriptions, while dev and test set only contained product titles. Each title or description sentence was assigned to only one category. The statistics of the e-commerce data sets are given in Table 1.

### 7.2 Model Training

We implemented our neural translation model in Python using the Blocks deep learning library van Merriënboer et al. (2015) based on the open-source MILA translation project. We compared our implementation of NMT baseline system with Bahdanau et al. (2014) on the WMT 2014 English-to-French machine translation task and obtained a similar BLEU score on the official test set as they reported in Bahdanau et al. (2014). Then we implemented the topic-aware

[1]This data set with topic labels is publicly available at https://github.com/wenhuchen/iwslt-2015-de-en-topics.

| E-commerce En→Fr | BLEU [%] | TER [%] |
|---|---|---|
| Baseline NMT | 18.6 | 68.5 |
| +prefixed human-labeled categs | 18.3 | 69.3 |
| +readout human-labeled categs | 19.7 | 65.3 |
| +readout LDA topics | 14.5 | 74.9 |

Table 2: Comparison of different approaches for topic-aware NMT.

algorithm (section 4), guided alignment training (section 3), and the bootstrapped training (section 5) into the NMT model. We trained separate models with various feature combinations. We also created an ensemble of different models to obtain the best NMT translation results.

In our experiments, we set the dimension of both source and target word embeddings to 620 and use a bi-directional GRU encoder and attention-based GRU decoder, the cell dimension of both are set to 1000. We selected the 50k most frequent German words and top 30k English words as vocabularies for the IWSLT task, and most frequent 52k English/French words for the e-commerce task. The optimization of the objective function was performed by using AdaDelta algorithm Zeiler (2012). We set the beam size to 10 for dev/test set beam search translation.

For training implementation, we use stochastic gradient descent with batch size of 100, saving model parameters after a certain number of epochs. We saved around 30 consecutive model parameters. We selected the best parameter set according to the sum of the established MT evaluation measures BLEU Papineni et al. (2002) and 1-TER Snover et al. (2006) on the development set. After model selection, we evaluated the best model on the test set. We report the test set BLEU and TER scores in Table 5 and Table 7.

We use TITAN X GPUs with 12GB of RAM to run experiments on Ubuntu Linux 14.04. The training converges in less than 24 hours on the IWSLT talk task and around 30 hours on the e-commerce task. The beam search on the test set for both tasks takes around 10 minutes, the exact time depends on the vocabulary size and beam size.

### 7.3 Effect of Topic-aware NMT

We tested different approaches to find out where topic information fits best into NMT, since topic information can affect alignment, word selection, etc. The most naive approach is to insert a pseudo topic word in the beginning of a sentence to bias the context of the sentence to a certain topic. We also tried topic vectors of different origin in the read-out layer of the network. We used both topics predicted automatically with the Latent Dirichlet Analysis (LDA) and human-labeled topics to feed into the network as shown in Figure 1.

The results on the e-commerce task in Table 2 show that category information as a pseudo topic word does not carry enough semantic and syntactic meaning in comparison to real source words to have a positive effect on the target words predicted in the decoder. The BLEU score of such system (18.3%) is even below the baseline (18.6%). In contrast, the human-labeled categories are more reliable and are able to positively influence word selection in the NMT decoder, significantly (19.7% BLEU) outperforming the baseline.

Replacing the human-labeled topic one-hot vectors of size 80 with the LDA-predicted topic distribution vectors of the same dimension in the read-out layer of the neural network deteriorated the BLEU and TER scores significantly. We attribute this to data sparseness problems when training the LDA of dimension 80 on product titles.

On the German-to-English task, we also observed MT quality improvements when using human-labeled topic information as described in Figure 1. Here, we extracted the topic embedding $E_c$ from different experiments and show their cosine distance in Figure 3. It's straightforward that in different experiments, the same topic tends to share similar representation in

| | |
|---|---|
| SRC | ich möchte Ihnen heute Morgen gerne von meinem Projekt, Kunst Aufräumen, erzählen. |
| NMT | I want to clean you this morning, from my project, to say Art. |
| +topics | I would like to talk to you today by my project, Art clean. |
| REF | I would like to talk to you this morning about my project, Tidying Up Art. |
| SRC | … unsere Kollegen an Tufts verbinden Modelle wie diese mit durch Tissue Engineering erzeugten Knochen, um zu sehen, wie Krebs sich von einem Teil des Körpers zum nächsten verbreiten könnte. |
| NMT | … our NOAA colleagues combined models of models like this with tissue generated bones from bones to see how cancer could spread from one part of the body, to the next distribution. |
| +topics | … our colleagues at Tufts are using models like this with tissue-based engineered bones to see how cancer could spread from a part of the body to the next part. |
| REF | … our colleagues at Tufts are mixing models like these with tissue-engineered bone to see how cancer might spread from one part of the body to the next. |

Table 3: Example of improved translation quality when topic information is used as input in the neural MT system (German-to-English IWSLT test set).

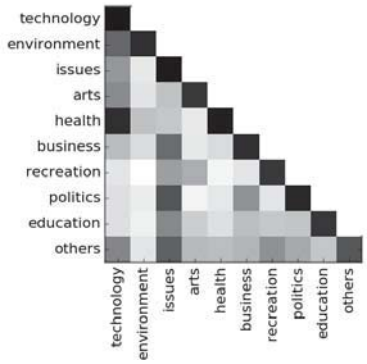



Figure 3: Topic embedding cosine distance.

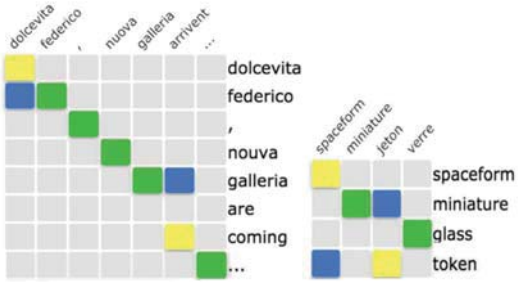



Figure 4: Refined alignment examples using guided-alignment learning (green blocks refer to the identical alignments from Baseline NMT and guided-alignment NMT, blue blocks refer to the alignment from baseline NMT, yellow blocks refer to guided-alignment NMT).

continuous embedding space. At the same time, closer topic pairs like "politics" and "issues" tend to have shorter distance from each other. Examples of improved German-to-English NMT translations when human-labeled topic information is used are shown in Table 3.

### 7.4 Implementation of Guided Alignment

To balance decoder cost and the attention weight cost, we experimented with different weights for these costs. We analyzed the relation between weight ratio and the final result in Table 4. Besides fixing the cost ratios during training, we also apply a heuristic to adjust the ratio as the training is progressing. One approach is to set a high value for the alignment cost in the beginning, then decay the weight to 90% after every epoch, finally eliminating the influence of the alignment after some time. This approach helps for the IWSLT task, but not on the e-commerce task. We assume that the alignment for the TED talk sentences seems to be easier for NMT to learn on its own than the alignment between product titles and their translations. We also analyzed the effect of using different loss functions for calculating alignment divergence (see Section 3.2). The difference between the squared error and cross-entropy is not so large as

| En→Fr | BLEU % | TER % |
|---|---|---|
| Baseline NMT | 18.6 | 68.3 |
| +ce (decay) | 20.5 | 65.8 |
| +ce (1:2) | 20.6 | 65.5 |
| +ce (1:1) | 20.2 | 65.0 |
| +ce (2:1) | 20.9 | 65.7 |
| +mse (1:1) | 20.8 | 64.5 |

Table 4: Comparison of different loss functions and weight ratios for guided alignment (cf. Equation 5).

| En→Fr systems | | BLEU % | TER % |
|---|---|---|---|
| 1. NMT in-domain (ID) | | 18.6 | 68.5 |
| 2. 1) + topic vectors | | 19.7 | 65.3 |
| 3. 1) + bootstrapping | | 20.1 | 66.2 |
| 4. 1) + guided alignment | | 20.9 | 65.7 |
| 5. NMT with 2) and 4) | | 21.3 | 64.3 |
| 6. NMT with 2) and 3) and 4) | | 20.7 | 66.2 |
| 7. NMT out-of-domain (OOD) | | 13.8 | 77.4 |
| 8. 7) + guided alignment | | 16.3 | 74.5 |
| 9. 8) + domain adaptation | | 25.0 | 60.1 |
| Ensemble NMT ID | system 4) | 24.5 | 60.9 |
| | system 5) | | |
| | system 6) | | |
| | NMT w. 3) and 4) | | |
| Ensemble NMT OOD | system 9) | 25.6 | 58.6 |
| | 9) with DW | | |
| | 9) w. topic vectors | | |

Table 5: Translation results on the En→Fr e-commerce task. (DW: decaying weight for the statistical alignment).

| | |
|---|---|
| SRC | Vintage Ollech & Wajs Early Bird Diver watch, Excellent! |
| SMT | Vintage Ollech & Wajs début oiseau montre de plongée, excellent! |
| NMT | Montre de plongée vintage Ollech & Wajs early bird, excellent! |
| REF | Montre de Plongée Vintage Ollech & Wajs Early Bird, Excellent ! |

Table 6: Example of improved translation quality of the NMT ensemble system vs. phrase-based baseline system (English-to-French title test set).

shown in Table 4. Since the cross-entropy function has a consistent form as decoder cost, we decided to use it in further experiments. We extracted the NMT attention weights and marked the connection with the highest score as hard alignment for each word. We drew the alignment in Figure 4 to compare baseline NMT and alignment-guided NMT. It can be seen from the graph that the guided alignment training truly improves the alignment correspondence.

### 7.5 Overall Results

The overall results on the e-commerce translation task and IWSLT task are shown in Table 5 and Table 7, respectively. We observed consistency on both tasks, in a sense that a feature that improves BLEU/TER results on one task is also beneficial for the other.

For comparison, we trained phrase-based SMT models using the Moses toolkit Koehn et al. (2007) on both translation tasks. We used the standard Moses features, including a 4-gram LM trained on the target side of the bilingual data, word-level and phrase-level translation probabilities, as well as the distortion model with the maximum distortion of 6. Our stronger phrase-based baseline included additional 5 features of a 4-gram operation sequence model – OSM Durrani et al. (2015).

On the e-commerce task, which is more challenging due to a high number of OOV words and placeholders, we observed that NMT translation output had many errors related to incorrect attention weights. To improve the attention mechanism, we applied guided alignment and

| # | De→En systems | | BLEU % | TER % |
|---|---|---|---|---|
| 1 | Phrase-based system | | 24.7 | 55.4 |
| 2 | Phrase-based system + OSM | | 25.7 | 55.1 |
| 3 | NMT | | 23.4 | 60.1 |
| 4 | NMT + topic vectors | | 23.7 | 59.6 |
| 5 | NMT + bootstrapping | | 24.1 | 58.6 |
| 6 | NMT + guided alignment | | 23.8 | 60.8 |
| 7 | NMT + topic vectors + bootstrapping | | 24.2 | 59.4 |
| 8 | NMT + topic vectors + bootstrapping + guided alignment | | 24.6 | 57.7 |
| 9 | Ensemble | NMT + topic vectors | 27.8 | 55.4 |
| | | NMT + topic vectors + guided alignment | | |
| | | NMT + topic vectors + bootstrapping | | |
| | | NMT + topic v. + guided alignment + bootstrapping | | |

Table 7: Overview of the translation results on the German-to-English IWSLT task.

bootstrapping. Both boosted the translation performance. Adding topic information increased the BLEU score to 21.3%. We selected the four best model parameters from various experiments to make an ensemble system, this improved the BLEU score to 24.5%. For the following experiment, we had pre-trained a model on WMT15 parallel data with the guided alignment technique, and then continued training on the e-commerce data for several epochs as described in section 6, performing domain adaptation. This approach proved to be extremely helpful, giving an increase of over 3.0% absolute in BLEU. Finally, we also applied ensemble methods on variants of the domain-adapted models to further increase the BLEU score to 25.6, which is 7.0 BLEU higher than the NMT baseline system, and only 0.6% BLEU behind the BLEU score of 26.2% for the state-of-the-art phrase-based baseline. Table 6 shows examples where the ensemble NMT system is better than the phrase-based system despite the slightly lower corpus-level BLEU score. In fact, a more detailed analysis of the sentence-level BLEU scores showed that the NMT translation of 386 titles out of 910 was ranked higher than the SMT translation, the reverse was true for 460 titles. In particular, the word order of noun phrases was observed to be better in the NMT translations.

On the IWSLT task (Table 7), the baseline NMT was not as far behind the phrase-based system as on the e-commerce task, and thus the obtained improvements were smaller than for product title translations. We observed that topic information is less helpful than bootstrapping and guided alignment learning. When we combined them, we reached the same BLEU score as the phrase-based system (see Table 7). Finally, we combined four variant systems to create an ensemble, which resulted in the BLEU score of 27.8%, surpassing the phrase-based translation with the OSM model by 2.1% BLEU absolute.

## 8 Conclusion

We have presented a novel guided alignment training for a NMT model that utilizes IBM model 4 Viterbi alignments to guide the attention mechanism. This approach was shown experimentally to bring consistent improvements of translation quality on e-commerce and spoken language translation tasks. Also on both tasks, the proposed novel way of utilizing topic meta-information in NMT was shown to improve BLEU and TER scores. We also showed improvements when using domain adaptation by continuing training of an out-of-domain NMT system on in-domain parallel data. In the future, we would like to investigate how to effectively make use of the abundant monolingual data with human-labeled product category information that we have available for the envisioned e-commerce application.

## References


Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Brown, P. F., Pietra, S. A. D., Pietra, V. J. D., and Mercer, R. L. (1993). The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19:263–311.

Cettolo, M., Girardi, C., and Federico, M. (2012). Wit$^3$: Web inventory of transcribed and translated talks. In *Proceedings of the 16$^{th}$ Conference of the European Association for Machine Translation (EAMT)*, pages 261–268, Trento, Italy.

Cho, K., van Merriënboer, B., Bahdanau, D., and Bengio, Y. (2014a). On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014b). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Cohn, T., Hoang, C. D. V., and Vymolova, E. (2016). Incorporating structural alignment biases into an attention neural translation model. *arXiv preprint arXiv:1601.01085*.

Durrani, N., Schmid, H., Fraser, A., Koehn, P., and Schütze, H. (2015). The operation sequence model-combining n-gram-based and phrase-based statistical machine translation. *Computational Linguistics*.

Goodfellow, I. J., Warde-Farley, D., Mirza, M., Courville, A., and Bengio, Y. (2013). Maxout networks. *arXiv preprint arXiv:1302.4389*.

Hasler, E., Blunsom, P., Koehn, P., and Haddow, B. (2014). Dynamic topic adaptation for phrase-based MT. In *Proceedings of EACL*, pages 328–337.

Jean, S., Cho, K., Memisevic, R., and Bengio, Y. (2014). On using very large target vocabulary for neural machine translation. *CoRR*, abs/1412.2007.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., et al. (2007). Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, pages 177–180. Association for Computational Linguistics.

Koehn, P. and Schroeder, J. (2007). Experiments in domain adaptation for statistical machine translation. In *Proceedings of the second workshop on statistical machine translation*, pages 224–227. Association for Computational Linguistics.

Luong, M.-T. and Manning, C. D. (2015). Stanford neural machine translation systems for spoken language domain.

Luong, M.-T., Pham, H., and Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Luong, M.-T., Sutskever, I., Le, Q. V., Vinyals, O., and Zaremba, W. (2014). Addressing the rare word problem in neural machine translation. *arXiv preprint arXiv:1410.8206*.

Mathur, P., Federico, M., Köprü, S., Khadivi, S., and Sawaf, H. (2015). Topic adaptation for machine translation of e-commerce content. *Proceedings of MT Summit XV*, page 270.

Mi, H., Wang, Z., and Ittycheriah, A. (2016). Supervised attentions for neural machine translation. *arXiv preprint arXiv:1608.00112*.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Och, F. J. and Ney, H. (2003). A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Snover, M., Dorr, B., Schwartz, R., Micciulla, L., and Makhoul, J. (2006). A study of translation edit rate with targeted human annotation. In *Proceedings of association for machine translation in the Americas*, pages 223–231.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

van Merriënboer, B., Bahdanau, D., Dumoulin, V., Serdyuk, D., Warde-Farley, D., Chorowski, J., and Bengio, Y. (2015). Blocks and fuel: Frameworks for deep learning. *arXiv preprint arXiv:1506.00619*.

Zeiler, M. D. (2012). Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

# Improving Neural Machine Translation on resource-limited pairs using auxiliary data of a third language

**Ander Martínez** ander.martinez.zy4@is.naist.jp
**Yuji Matsumoto** matsu@is.naist.jp
Graduate School of Information Science, Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara, 630-0192, Japan

**Abstract**

In the recent years interest in Deep Neural Networks (DNN) has grown in the field of Natural Language Processing, as new training methods have been proposed. The usage of DNN has achieved state-of-the-art performance in various areas. Neural Machine Translation (NMT) described by Bahdanau et al. (2014) and its successive variations have shown promising results. DNN, however, tend to over-fit on small data-sets, which makes this method impracticable for resource-limited language pairs. This article combines three different ideas (splitting words into smaller units, using an extra dataset of a related language pair and using monolingual data) for improving the performance of NMT models on language pairs with limited data. Our experiments show that, in some cases, our proposed approach to subword-units performs better than BPE (Byte pair encoding) and that auxiliary language-pairs and monolingual data can help improve the performance of languages with limited resources.

## 1 Introduction

In the recent years interest in Deep Neural Networks (DNN) has grown in the field of Natural Language Processing (NLP), as new training methods (Blunsom and Kalchbrenner, 2013; Sutskever et al., 2014) have been proposed. The encoder-decoder approach for Neural Machine Translation (NMT) consists in encoding the source sentence into an intermediate vector representation and then generating (decoding) the target sentence from this representation. Cho et al. (2014) is an example of this approach.

The NMT approach of jointly training alignment and translation models described by Bahdanau et al. (2014) and its successive variations have shown promising results. Its attention mechanism deals with the problem of having a fixed length vector for sentences of varying length by encoding the source sentence into a set of vectors, one vector for each of the tokens in the source sentence.

NMT doesn't need complex feature engineering, which is convenient when dealing with resource-limited languages. However, a large parallel corpus is still needed in order to get competitive performance and avoid overfitting. As an example, Bahdanau et al. (2014) and Jean et al. (2015) use a dataset of about 12 million parallel sentences.

Two main problems arise when using a small dataset for training a MT model.

One of those problems is that the vocabulary exposed by a small dataset is inherently small. Also, even if a word shows up in the data it may occur too few times for learning a reliable representation. One strategy for minimizing this problem is subdividing words into subword

units, like syllables. Doing so reduces the total vocabulary size and increases the hit-rate of each symbol in the dataset. This has been explored in Sennrich et al. (2015b).

Another problem concerns large NMT models. When the number of parameters is too large compared to the data size, the model may optimize for memorizing all or a large part of the dataset instead of modeling the translation; overfitting in practice. A solution to this problem is to reduce the model size to better match the amount of data. However, many relevant features may not be modeled with a smaller number of parameters. Another approach is to artificially increase the number of samples by counterfeiting or using data of a third language.

This research explores how much of an improvement using auxiliary parallel sentences from a third language to the target language (A $\rightarrow$ T) in modeling MT from of a resource-limited language pair (S $\rightarrow$ T) brings. We also explore the effect of using an auxiliary language on the decoder side.

For the case of phrase-based Statistical Machine Translation, similar ideas have been explored before with varied results. For example, for closely related pairs (Nakov and Ng, 2012) or through lexical triangulation (Crego et al., 2010; Dholakia and Sarkar, 2014). For NMT, a couple of authors have also explored this possibility. Dong et al. (2015) modeled translation to different targets from a common source with shared representation. Firat et al. (2016) also explored the case of a common target language for different source languages. Both papers claimed to get higher translation quality over individually trained models. A comparison to these papers follows in Section 2.

In order to assist the learning of the target language pair, MT for the auxiliary pair is trained jointly. We argue that doing so prevents the language pair with the small dataset from overfitting and leads to more robust models. This problem could also be addressed through Transfer Learning, as explored in Yosinski et al. (2014) and Zoph et al. (2016), but that approach falls outside the scope of this article.

A third solution to parallel data scarcity is using monolingual data in addition in order to make the Language Model (LM) at the target side stronger. A strong LM at the decoder can increase performance, as already tried by Sennrich et al. (2015a).

We perform experiments that are analogous to the ones described in Firat et al. (2016) with focus on the more resource-limited pairs and use their results as a baseline for comparison.

This article has the following sections: Section 2 summarizes previous related work; in Section 3, the proposed approach is described; Section 4 presents some experiments with their results and analysis; finally, in Section 5, we draw some conclusions.

## 2 Related work

### 2.1 Neural Machine Translation

The Neural Machine Translation method proposed in Bahdanau et al. (2014) on which this work is based is briefly described in this section.

NMT models, like SMT models, are trained to maximize the conditional log-probability of every translation in the training set $Y^{(i)}$ w.r.t. their corresponding source sentence $X^{(i)}$ and model's parameters $\theta$.

$$\theta^* = \arg\max_{\theta} \sum_{i:Y^{(i)} \in Y} \log p(Y^{(i)}|X^{(i)}, \theta). \tag{1}$$

In order to compute the probability of a translation, the sequence $x = (x_1, \ldots, x_T)$ is first encoded into a sequence of annotations $h = (h_1, \ldots, h_T)$, by a bidirectional recurrent neural network.

$$h_t = [\overleftarrow{h}_t; \vec{h}_t] = [f_1(x_t, \overleftarrow{h}_{t+1}); f_2(x_t, \vec{h}_{t-1})], \tag{2}$$

where $f_1$ and $f_2$ are both Gated Recurrent Units (GRU) as described in Cho et al. (2014).

This annotations are used by the decoder to estimate the probability of the translation, one word at a time, based also on the previous words in the sentence. A convex sum of the annotations, $c_\tau$, is computed each time according to their contribution to the upcoming word. The weight of each annotation is computed in the following way:

$$c_\tau = \sum_{i=1}^{T_x} \alpha_{\tau,i} h_i, \tag{3}$$

$$\alpha_{\tau,i} = \frac{\exp\{a(h_i, z_{\tau-1}, E_y[\tilde{y}_{\tau-1}])\}}{\sum_{j=1}^{T_x} \exp\{a(h_j, z_{\tau-1}, E_y[\tilde{y}_{\tau-1}])\}}, \tag{4}$$

where $z_{\tau-1}$ is the previous hidden state of the decoder GRU and $E_y[\tilde{y}_{\tau-1}]$ is the word embedding of the previously produced word. During training, the embedding of the expected previous word is used instead. The function $a$ scores the relevance of the annotation in the current context. It is defined as a projection of the sum of the projections of each of its parameters.

A softmax layer can be applied on a projection of $c_\tau$, $z_\tau$ and the embedding of the previous work to compute the probability of each candidate word in the target vocabulary.

$$p(y_\tau | y_{<\tau}, x) \propto exp\{q(y_{\tau-1}, z_\tau, c_\tau)\}. \tag{5}$$

When generating translations of new sentences beam-search is used based on the trained probability model.

The hidden state of the decoder $z_\tau$ is updated with respect to the convex sum $c_\tau$, as described in Cho et al. (2014).

$$z_\tau = g(y_{\tau-1}, z_{\tau-1}, c_\tau). \tag{6}$$

In this article, the part of the network that produces $h$ is referred as *encoder* and the rest of the network as *decoder*. In articles by other authors, the parameters used exclusively to compute $\alpha_\tau$ may not be considered part of the decoder.

### 2.2 Subword-units

Out-of-vocabulary words and low word hits are an inherent problem to small datasets. Instead of using words as translation unit, sub-word elements can be used. By doing so, we get more symbols from the same dataset and thus a bigger percentage of all possible symbols will appear and the number of appearances of each of these symbols will become higher. Also, by making the symbols shorter the number of possible symbols also reduces.

Sennrich et al. (2015b) tried using subword units for improving translation of rare words. For doing so, they first applied BPE (Byte pair encoding) (Gage, 1994) to the word list at the character level. BPE consists in replacing the most common symbol pairs with a new symbol consecutively until the symbol table has a certain size. They didn't merge symbols across words.

In order to have more similar subword units at the source and target languages, they transliterated the Cyrillic characters into Latin characters and trained the merging jointly. This works for the use case explored in that article of translating proper names and other words that can be mapped phonetically. However, it doesn't match well with morphemes (the smallest meaningful unit of a language), which are usually short, and it minimizes word hits, which isn't desirable in the case of small datasets.

They tried two vocabulary sizes: one of 60,000 words and another one of 90,000 words for the joint case.

### 2.3 Multilingual Neural Machine translation

The idea of jointly training additional language pairs to improve the translation quality of a model has already been tried.

The method explained in Dong et al. (2015) consists in translating to a set of languages from English using the same English encoder for each pair. They only investigate the case where the source language is shared. This approach mainly improves the quality of the encoder side as it studies more datapoints, but the quality of the decoder doesn't improve that much because its amount of data remains the same.

The model described by Bahdanau et al. (2014) has 85,967,240 floating point parameters when using a vocabulary size of 30,000 at both ends. Of these parameters 32.95% are related to the encoder and 67.05% to the decoder and soft-alignment system. This suggests that the decoder will overfit more easily under data-scarce conditions.

Another article investigating the use of additional language pairs is Firat et al. (2016). They train five language pairs in both directions, which makes ten individual models. For each of the six languages, the same encoder and decoder parameters are used when repeated in a pair and the parameters of the attention mechanism (function $a$ in Equation (4) in this article) are shared for all pairs.

In the encoder, the hidden states, called *annotations*, obtained from the forward and backward RNN are projected into a new vector. This allows for the annotations to be more language-independent, as it doesn't make a difference whether a feature is extracted by the network iterating over the source sentence forwards or backwards.

$$h_t^n = W_{adp}^n[\overleftarrow{h}_t; \vec{h}_t] \tag{7}$$

For each of the language pairs, they feed a minibatch to the corresponding model and update its weights accordingly; one language-pair at a time.

They obtained significant improvement for small datasets and when the repeated language (English) was in the decoder.

They applied this method to datasets of varying sizes, starting on 100K parallel sentences for English-French translation and 210K for German-English.

### 2.4 Monolingual data

Sennrich et al. (2015a) introduced a method for integrating pre-trained LMs with NMT models in order to improve the translation quality. In their Deep-Fusion approach, they trained a LM on monolingual data and a NMT model on parallel data, and then integrated the LM prediction before the Softmax layer to contribute in the selection of the next word.

The LM was based on the RNNLM (Deoras, 2011) approach using GRUs in the decoder, which in effect, is very similar to the model described in Section 2.1 but without the attention mechanism. That is, the decoder only depends on one monolingual $y$ sentence, without any encoder. The equivalents to Equations 5 and 6 are:

$$p(y_\tau|y_{<\tau}) \propto exp\{q(y_{\tau-1}, z_\tau^{\mathrm{LM}})\}. \tag{8}$$

$$z_\tau^{\mathrm{LM}} = g(y_{\tau-1}, z_{\tau-1}^{\mathrm{LM}}). \tag{9}$$

In order to integrate this LM with the Translation Model (TM), they first scale the hidden state of the LM and then concatenate it to the hidden state of the TM before computing the softmax. The scale factor $g_\tau$ for the LM is given by:

$$g_\tau = \sigma(v_g^T z_\tau^{\mathrm{LM}} + b_g), \tag{10}$$

where $v_g$ and $b_g$ are learned weights and bias, respectively. After merging, the equivalent to Equation 5 is

$$p(y_\tau|y_{<\tau}, x) \propto exp\{q(y_{\tau-1}, z_\tau, z_\tau^{\mathrm{LM}}, c_\tau)\}. \quad (11)$$

The model from Bahdanau et al. (2014) already trains something close to a LM from the parallel data, as can be seen in Equation 5. This approach increases the complexity of the model, as two states for two decoders need to be updated for every timestep.

## 3 Proposed solution

The method proposed here consists in jointly training various models with shared parameters; either the encoder or the decoder parameters. By jointly training the models they will co-adapt and benefit from each other. In the experiments we only explore the possibility of using one extra auxiliary language, either as the source language or as the target language, in addition to the intended source and target languages. More than one extra language could be used in a similar fashion.

In addition to using parallel data from an auxiliary language we also experiment with using monolingual data to obtain improved results.

We used subword units instead of words as the translation unit as a method to deal with large vocabulary sizes and differences in morpheme-per-word ratios (synthetic vs isolating).

### 3.1 Subword units

We pre-processed the data to split words into subword units for training.

Any word in the dataset was split into a number of subword units equal to its number of vowels. Each of the subword units consists of a vowel with all its surrounding consonants. Numbers aren't split. Word-boundary marks were included into the subwords. As an example, the sentence "the 54 polychromatic mats" is split into the sequence ["_the_", "_54_", "_pol", "lychr", "chrom", "mat", "tic_", "_mats_"].

The motivation behind this approach is that the subword-units generated by this approach are similar in shape to syllables and syllables map relatively well to morphemes in many languages. By using these short subword-units the model could learn some kind of morphological derivation.

In our experiments we used a vocabulary size of 30k symbols. Using only these symbols would result in too many out-of-vocabulary symbols. In order to alleviate this problem we tried to fit the infrequent symbols into the vocabulary by trimming one consonant at a time from the beginning or the end of the symbol until they matched a symbol in the vocabulary. We didn't trim consonants from word-boundaries and when for the inner symbols we trimmed from the side with more consonants. This reduced the number of unknown symbols considerably but didn't eliminate them. Note that sometimes excessive trimming can happen to the point that the original word cannot be restored, but this is still preferable to an unknown symbol.

The subword units are merged after translation before computing the BLEU score in order to be comparable to other authors' results. The subword units are merged using a simple regular expression of the form "`s/([:consonant:]+) \1/\1/g`".

We notice two problems with this approach. For languages with long consonants clusters like Czech the vocabulary size will grow faster. We observed this problem with German when compared to English. The number of out-of-vocabulary symbols in our datasets can be seen in Table 1.

The other problem is related to sound changes. For languages with a lot of sound changes, like consonant gradation in Finnish, morphological changes can produce a different symbol, which increases the need of data. As an example, the word *poika* will use the symbols `_poik` and `ka_` but in genitive it changes to *pojan* resulting in `_poj` and `jan_`, two different symbols.

This approach produces more symbols for each sentence than the BPE approach. As an example, Firat et al. (2016) got 43.67M Finnish tokens from a 2.03M sentence dataset, i.e. 21.5 symbols per sentence, while our method produces 52.0 symbols. For English and German they got 26.9 and 28.3 respectively, while we get 47.9 and 45.2. This increased number of symbols helps specially with small datasets and languages with many one-syllable morphemes.

### 3.2 Auxiliary language parallel data

We train a model on an auxiliary language pair which shares one of its languages with the target language pair together with the main model. Both models, the target translation model and the auxiliary translation model, are trained to minimize the Negative Log Likelihood (NLL) on their corresponding datasets. In order to train both of them jointly, for each of them, the mean NLL and the gradients with respect to each parameter are computed on a minibatch. The weighted sum of these gradients are further used by ADADELTA (Zeiler, 2012) to update the weights.

To prevent one of the models' updates from outweighing the other's, gradients of shared parameters are scaled based on their previous $d$ costs, in such a way that the weakest model's gradients get promoted. The intuition behind this is that, if they are generalizing correctly, the main model and the auxiliary model should produce similar costs, as they are defined in a similar fashion. If a model performs better than the other we can decelerate its optimization while accelerating the other to balance them properly. The auxiliary model has less risk of overfitting, as it trains on a bigger dataset. Therefore, if the main model produces costs similar to those produced by the auxiliary model it is generalizing better than when the cost function evaluates lower. The parameters that are not shared by the two models are not scaled.

The scale factor at epoch $t$ for the gradient's of the auxiliary model $s_t^{aux}$ and the target model $s_t^{tgt}$ are computed as follows:

$$s_t^{aux} = \left(\frac{1}{2} - \frac{1}{1 + e^{(\mu_t^{tgt} - \mu_t^{aux})s}}\right) l + \frac{1}{2}, \tag{12}$$

$$s_t^{tgt} = 1 - s_t^{aux}; \tag{13}$$

where $s$ and $l$ are hyperparameters that control the *steepness* and he *range* of the function, respectively. $\mu_t^{tgt}$ and $\mu_t^{aux}$ are the mean of the last $d$ costs for the target model and the auxiliary model, calculated as follows:

$$\mu_t^{\mathrm{m}} = \sum_{j=t-d}^{t} J_j^m, \tag{14}$$

where $m \in \{\mathrm{t}gt, \mathrm{a}ux\}$ and $J_j^m$ is the cost computed by model $m$ at timestep $j$. The pseudo-code for this algorithm can be seen in Algorithm 1 and the shape of Equation 12 in Figure 1.

We group sentences of similar length in minibatches so that each minibatch contains roughly the same number of symbols (in our experiments, no more than 4000 symbols per batch). Therefore, minibatches of longer sentences contain less sentence pairs. Because each minibatch can contain a different number of sentences and different number of words per sentence, averaging over words is necessary. Our cost function averaged the cost of every word in a sentence in every sentence in a minibatch. The cost of a word was measured as Negative Log Likelihood,

$$J_t^m(X_t, Y_t) = \frac{1}{N_t} \sum_{i=1}^{N_t} \left( \frac{1}{\mathrm{length}(Y_t^{(i)})} \sum_{j=1}^{\mathrm{length}(Y_t^{(i)})} -\log P_\theta(Y_t^{(i,j)} | X_t) \right), \tag{15}$$

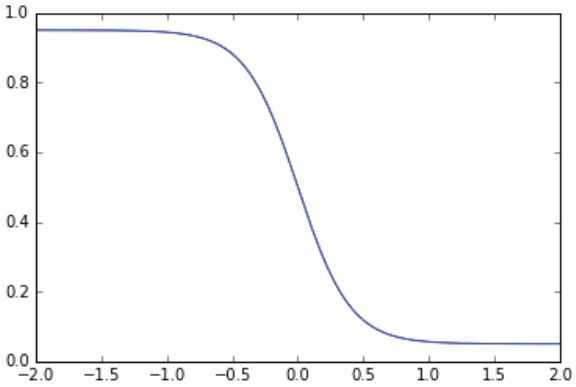



Figure 1: Equation 12 with parameters $s = 5$ and $l = 0.9$

where $X_t$ and $Y_t$ are the source and translation sentences in the minibatch at timestep $t$, $N_t$ is the number of sentence pairs in the minibatch and length$(Y_t^{(i)})$ is the length in symbols of the the translation $i$ at timestep $t$.

In Figure 2 we can see how the costs from the German-English and the French-English models descend at the same pace. The $s$ hyperparameter was set to 50. The weighted sum of these gradients are further used by ADADELTA Zeiler (2012) to update the weights.

### 3.3 Monolingual data

Monolingual data can be leveraged in a similar way to auxiliary data. We input monolingual sentences as both, input and expected output, in the manner of an auto-encoder (e.g. De→De).

In contrast with the data from an auxiliary pair the model for the monolingual data is trained after the target model has converged. The encoder and the decoder are trained separately. First the non-shared part (either encoder or decoder) is trained until convergence and then the rest of the model is trained using Equation 12 to scale the updates.

We used different parameters for Equation 12 for the auxiliary language-pair data and the monolingual data.

This kind of model, similar to an auto-encoder, is expected to quickly memorize a copying mechanism from source to target. The method here described can slow down this memorization.

## 4 Evaluation

### 4.1 Data and Methods

The datasets used in the experiments are similar to some described in Firat et al. (2016). We performed two sets of experiments: one for the case where the auxiliary language is in the source (De+Fr → En) side and the other one for the case where the auxiliary language is in the target side (En → Fi+Fr). In both cases French is the auxiliary language. The parallel data is from the datasets available for WMT'15 for each language pair. We randomly picked 100k and 200k sentences for the En-Fi case and 210k and 420k sentences for the De-En case. The development and test sets are for En-Fi were `newsdev-2015` and `newstest-2015` respectively; and for De-En, `newstest-2013` and `newstest-2015` respectively.

All the sentences were tokenized using the `tokenize.perl` script included in Moses and then cleaned using the scripts `normalize-punctuation.perl`, `remove-non-printing-char.perl` and `deescape-special-chars.perl`. In-

**Algorithm 1** Training

**procedure** TRAIN
  $\texttt{costs\_aux}, \texttt{costs\_tgt} \leftarrow ([], [])$
  $\texttt{b} \leftarrow 0.5$
  **while** not done **do**
    $M_{tgt}, M_{aux} \leftarrow \texttt{getNextMinibatches}()$
    $J_{tgt}, J_{aux} \leftarrow (\texttt{costTgt}(\theta_{tgt}, M_{tgt}), \texttt{costAux}(\theta_{aux}, M_{aux}))$
    **for** $\theta_i \in (\theta_{tgt} \cup \theta_{aux})$ **do**
      **if** $\theta_i \in (\theta_{tgt} \cap \theta_{aux})$ **then**
        $g_{tgt} \leftarrow \frac{\partial}{\partial \theta_i} \texttt{costTgt}(\theta_{tgt}, M_{tgt})$
        $g_{aux} \leftarrow \frac{\partial}{\partial \theta_i} \texttt{costAux}(\theta_{aux}, M_{aux})$
        $\texttt{g} \leftarrow \texttt{b} \cdot g_{aux} + (1 - \texttt{b}) \cdot g_{tgt}$
      **else if** $\theta_i \in \theta_{tgt}$ **then**
        $\texttt{g} \leftarrow \frac{\partial}{\partial \theta_i} \texttt{costTgt}(\theta_{tgt}, M_{tgt})$
      **else**
        $\texttt{g} \leftarrow \frac{\partial}{\partial \theta_i} \texttt{costAux}(\theta_{aux}, M_{aux})$
      $\theta_i \leftarrow \texttt{applyADADELTA}(\theta_i, g)$
    push $J_{tgt}$ onto $\texttt{costs\_tgt}$
    push $J_{aux}$ onto $\texttt{costs\_aux}$
    **if** $|\texttt{costs\_aux}| > d$ **then** shift $\texttt{costs\_aux}$
    **if** $|\texttt{costs\_tgt}| > d$ **then** shift $\texttt{costs\_tgt}$
    $(\mu_{aux}, \mu_{tgt}) \leftarrow$ (mean($\texttt{costs\_aux}$), mean($\texttt{costs\_tgt}$))
    $\texttt{b} \leftarrow \left(\frac{1}{2} - \frac{1}{1+e^{(\mu_{tgt} - \mu_{aux})s}}\right) l + \frac{1}{2}$

stead of space separated words, we used the subword units described in Section 3.1 with a vocabulary size of 30k symbols. Unlike the BPE subword method used in Firat et al. (2016) our method allows for *UNK* symbols. The amount of these symbols and other statistics of the datasets can be seen in Table 1.

Twelve models were trained, three on each of the four described datasets. We first trained four models using only the described subword approach, without any additional data. Then, we also trained four models using the auxiliary data. Finally, we further train the models from the previous step using the monolingual data in addition to the main and auxiliary datasets.

All the models had the same number of parameters in their encoders and decoders.

We evaluate the performance of the trained models based on their BLEU score using the `multi-bleu.perl` script from Moses. We merged the subword tokens into words before evaluation and computed the score on lowercase text.

### 4.2 Implementation

The code[1] for the proposed method was implemented in Python using Theano (2016). It runs on a single GPU computing the cost and gradients for each language pair one at a time. This could be parallelized using an extra GPU to speed the training up. We used three different models of GPU for training: *GeForce GTX 980 Ti*, *Tesla K40m* and *GeForce GTX TITAN X*.

We used a vocabulary size of 30k symbols for each language. All the hidden layers had 1,000 units. The embeddings for the each symbol were 620 dimensions long. The attention vectors from the bidirectional RNNs where projected into 1,000 dimension vector as described

[1] Code available at `https://github.com/basaundi/amta2016`

| | role | sentence pairs | symbols | | UNK | |
|---|---|---|---|---|---|---|
| | | | src | tgt | src | tgt |
| en-fi | train | 100k | 4.7m | 5.2m | 11.6k (0.2%) | 39.6k (0.7%) |
| | train | 200k | 9.3m | 10.4m | 23.3k (0.2%) | 78.6k (0.7%) |
| | development | 1500 | 54.2k | 60.6k | 904 (1.7%) | 986 (1.6%) |
| | test | 1370 | 45.4k | 51k | 940 (2.1%) | 858 (1.7%) |
| en-fr | auxiliary | 4m | 222.3m | 262.6m | 783.6k (0.3%) | 747.7k (0.4%) |
| fi | monolingual | 8m | 399.5m | | 3.6m (0.9%) | |
| de-en | train | 210k | 10m | 9.5m | 350.2k (3.5%) | 90.2k (0.9%) |
| | train | 420k | 20.1m | 19m | 700.3k (3.5%) | 180.3k (0.9%) |
| | development | 3000 | 115.5k | 105.6k | 3641 (3.2%) | 1120 (1.1%) |
| | test | 2169 | 80k | 75.3k | 2801 (3.5%) | 1330 (1.8%) |
| fr-en | auxiliary | 4m | 267.4m | 227.2m | 1m (0.4%) | 862.3k (0.4%) |
| de | monolingual | 8m | 357m | | 5.2m (1.5%) | |

Table 1: Statistics of the corpora used in the experiments. The symbols are subword tokens as described in this article.

| | Size | BPE | Firat | SW | +Aux | +Aux +Mono |
|---|---|---|---|---|---|---|
| En → Fi | 100k | 3.93/3.42 | 3.21/4.2 | 4.17/3.89 | 3.81/3.74 | 4.54/3.99 |
| | 200k | 5.21/4.79 | 4.16/5.71 | 5.28/4.70 | 5.15/5.08 | 5.63/5.32 |

Table 2: BLEU scores for the Finnish development and test datasets (separated by /). SW is our new subword-unit method, `+Aux` is using subwords and English-French auxiliary data and `+Aux +Mono` is using subwords with auxiliary and monolingual data.

in Equation 7.

The models are optimized using ADADELTA Zeiler (2012) with the $\rho$ parameter set to 0.95. We clipped all the gradients to an L2 norm of 1 after weight-summing them as described in Section 3.2.

During training, a minibatch from each used dataset was fed to the corresponding computational graph. Sentences of similar length were grouped together in minibatches of no more than 40k symbols. Therefore, minibatches of longer sentences contained less sentences than those with shorter sentences.

We trained the models for a week and kept the one that performed best on the development set. For the models using monolingual data, we first trained the model on the corresponding training and auxiliary datasets. Then, we further trained the model using also the auxiliary data and monolingual data stopping after the third drop of performance on the development data. This happened quite fast, as the model memorizes the copy mechanism easily.

We used different values for hyperparameters $s$ and $l$ in Equation 12 for those parameters shared with the auxiliary language-pair model ($s_{aux} = 50$ and $l_{aux} = 1$) and those parameters shared with the monolingual model ($s_{mono} = 30$ and $l_{mono} = 0.9$). For both cases the number of previous costs $d$ used for the running average was 30.

### 4.3 Results and analysis

The results for the experiments are summarized in Tables 2 and 3. For German, we can see that the results from Firat et al. (2016) for their original approach using the same dataset were better for the same target dataset in all cases. They didn't try English-Finnish translation with small datasets.

| | Size | Single † | Firat † | SW | +Aux | +Aux +Mono |
|---|---|---|---|---|---|---|
| De → En | 210k | 14.27/13.20 | 16.96/16.26 | 15.85/14.24 | 16.61/15.39 | 16.66/15.30 |
| | 420k | 18.32/17.32 | 19.81/19.63 | 18.72/17.10 | 18.58/17.17 | 18.62/17.26 |

Table 3: BLEU scores for the German development and test datasets (separated by /). SW is our new subword-unit method, `+Aux` is using subwords and auxiliary data and `+Aux +Mono` is using subwords with auxiliary and monolingual data. † Results from Firat et al. (2016).

When compared to the results from the single language-pair model using BPE, we can see that our approach for the subword units worked well with German. The performance gap is bigger with the smaller dataset. Our approach produces more symbols for the same dataset, which means there is more data for training. However, guessing the right word also becomes harder because more subwords need to be decoded correctly in order to get one correct word. This difference should be more noticeable for languages with longer words. In Table 5, we can see that many subwords were guessed correctly by the different models but didn't form the right word.

Using the auxiliary language pair helped for the German-English model with the small dataset. The extra data didn't help with the bigger dataset. Our guess is that one week wasn't enough for the model to benefit from all the data, as our method takes a lot of time to complete one iteration when using a single GPU. The score on the development dataset was still growing when the training was stopped. The performance for the model trained on the bigger dataset improved when trained with the extra monolingual dataset. This improvement in the performance could be an effect of the extra training time. In Table 4 we can see the translations generated by the different models to a sample sentence.

For the English-Finnish models the auxiliary data helped only for the bigger dataset because the smaller dataset overfitted for Finnish before it could use any French data. Our results are inferior to the results from the approach described by Firat et al. (2016) [2]. The auxiliary dataset helped better when the auxiliary language was in the encoder (i.e., the language in the decoder was repeated).

In Figure 2 we can see how the NLL cost descended at the same pace for both German-English and French-English effectively preventing the model from overfitting for the smaller dataset. In our experiments we observe that the BLEU score on the validation set starts to decrease when the cost goes under 1.0 when using our proposed subword units.

Introducing the monolingual data helped prevent the English-Finnish models from overfitting but wasn't very helpful for German. In Figure 3 we observe how, without the additional data, the model over-fits and prepends an unnecessary adjective to the word Ukraine associated to *school* and *Ukraine*. The incorrect translation means "All children return to latter Ukraine".

The proposed subword-units are able to generate new words that do not appear in the training or auxiliary dataset by analogy. As an example, the word *biometrically* was guessed correctly by the German-English model even though this word appears in the test set for the first time and the English-Finnish model could generate the word *liitoksen* (*liitos* + *GEN*, of the annexation).

## 5 Conclusion

We evaluated three different ideas that could help improve NMT for language pairs with limited resources. We trained three models applying from one to the three of the ideas on four different datasets and assessed them by measuring their BLEU scores on the same test-set.

[2] Computed using the code at `https://github.com/nyu-dl/dl4mt-multi`

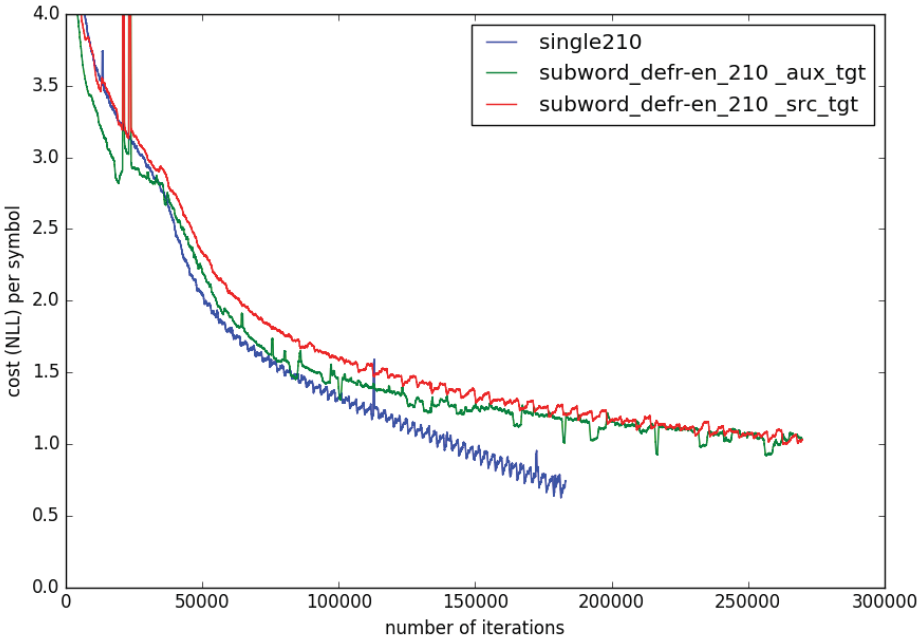



Figure 2: NLL-per-word cost evolution for a model trained on a single language pair (de-en) and one trained using an auxiliary language pair. The cost for the model trained without auxiliary data sinks because of overfitting. For the other model, the costs of the target language-pair (upper) and the auxiliary language-pair (lower) descend at the same pace.

The first idea was to use subword-units instead of words. The subword-units we applied were intended to map better to single morphemes when compared to other methods as BPE. Our experiments showed that these kind of subword-units can help with smaller datasets, getting a +1.24 BLEU score improvement when compared to BPE for German-English translation when trained on a 210k sentence pair dataset.

Using data form an auxiliary language-pair helped improve the performance for small datasets (about 200k parallel sentences) but when the dataset was too small (100k parallel sentences) and the auxiliary language was in the decoder side the model ended memorizing the small dataset despite the extra data. Even though the extra data improved the performance the solution by Firat et al. (2016) got better results for German-English translation.

The monolingual data helped prevent overfitting in the cases when the dataset in the decoder side was too small. The monolingual data didn't help very much with German-English translation. Our proposed solution increased considerably the time needed for each iteration and thus the time for convergence.

In the future, we would like to rethink the subword-unit approach to represent better the consonant-gradation and other small sound changes related to morpheme combination. Also, faster GPUs may make our solution more feasible in the future.

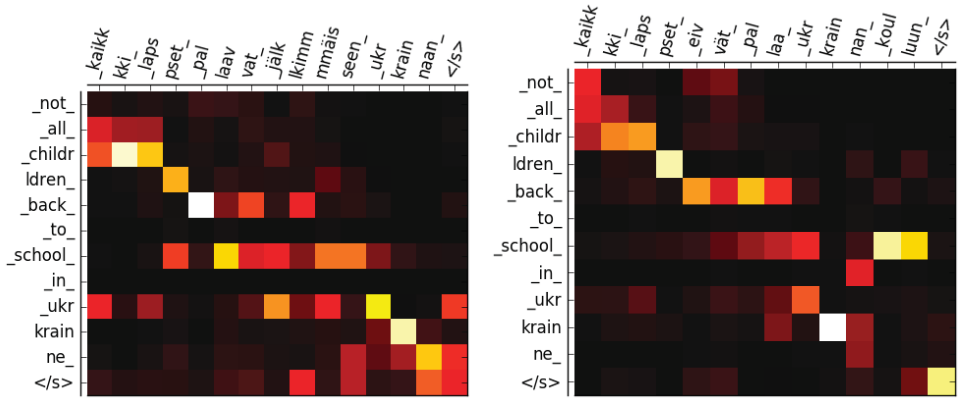



Figure 3: Alignments for English-Finnish translations by the ”SW” (left) and ”+Aux+Mono” models (right). The first model prepends the adjective ”jälkimmäiseen” (latter) to Ukraine and omits the words *eivät* (not) and *koulu* (school).

| | |
|---|---|
| source | die premierminister indiens und japans trafen sich in tokio . |
| reference | india and japan prime ministers meet in tokyo |
| subword 210k | the prime minister of india and japan in tokyo in tokyo . |
| +aux 210k | the prime ministers of india and japan met came in tokyo . |
| +aux+mono 210k | the prime minister of india and japan came to tokyo . |
| subword 420k | prime minister india and japan met in tokio . |
| +aux 420k | the prime minister of india and japan joined in tokyo . |
| +aux+mono 420k | the prime minister , india and japan , met in tokyo . |

Table 4: Sample translations of the first German sentence in the test set produced by the different models.

| | |
|---|---|
| source | the organisations have promised a career solution by the end of autumn , but according to the latest estimation it would be achieved this week . |
| reference | järjestöt ovat luvanneet työuraratkaisun syksyyn mennessä , mutta tuoreimman arvion mukaan se syntyisi tämän viikon aikana . |
| subword 100k | järjestöt ovat luvanneet erinomaisen urauden loppuun saattamisen loppuun saakka , mutta myöhemmin tällä viikolla toteutetulla arviolla voitaisiin saavuttaa tämän viikon kuluessa . |
| +aux 100k | järjestöt ovat luvanneet suorituskeskustelun jälkeen syksyn loppuun mennessä , mutta viimeisimpien arvioiden mukaan tämä ehdotus voitaisiin . |
| +aux+mono 100k | järjestöt ovat luvanneet tehtyä urakentamalla syksyn loppuun mennessä , mutta viimeisimpänä arvUNKstana se olisi saavuttanut tämän viikon kuluesss . |
| subword 200k | järjestöillä on luvattu uralla uratkaisu vuoden loppuun mennessä , mutta viimeisimmän arvion mukaan se olisi saavutettavissa . |
| +aux 200k | järjestöt ovat luvanneet saavutettavan uran loppuun mennessä , mutta viimeisten arvioiden mukaan tämä olisi mahdollista saavuttaa tällä viikolla . |
| +aux+mono 200k | järjestöt ovat luvanneet suoran ratkaisun syksyn loppuun mennessä , mutta viimeisin arvioitu olisi viime viikolla saavutettavissa . |

Table 5: Sample translations into Finnish. Many subwords were guessed correctly but didn’t form the correct word.

## References


Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv:1409.0473 [cs, stat]*. arXiv: 1409.0473.

Blunsom, P. and Kalchbrenner, N. (2013). Recurrent Continuous Translation Models. *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv:1406.1078 [cs, stat]*. arXiv: 1406.1078.

Crego, J. M., Max, A., and Yvon, F. (2010). Local lexical adaptation in Machine Translation through triangulation: SMT helping SMT. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*. Coling 2010 Organizing Committee.

Deoras, A. (2011). RNNLM - Recurrent Neural Network Language Modeling Toolkit. *Microsoft Research*.

Dholakia, R. and Sarkar, A. (2014). Pivot-based triangulation for low-resource languages. In *Proc. AMTA*, pages 315–328.

Dong, D., Wu, H., He, W., Yu, D., and Wang, H. (2015). Multi-Task Learning for Multiple Language Translation. In *ACL*.

Firat, O., Cho, K., and Bengio, Y. (2016). Multi-Way, Multilingual Neural Machine Translation with a Shared Attention Mechanism. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 866–875, San Diego, California. Association for Computational Linguistics.

Gage, P. (1994). A New Algorithm for Data Compression. *C Users Journal*, 12(2):23–38.

Jean, S., Cho, K., Memisevic, R., and Bengio, Y. (2015). On Using Very Large Target Vocabulary for Neural Machine Translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1–10, Beijing, China. Association for Computational Linguistics.

Nakov, P. I. and Ng, H. T. (2012). Improving Statistical Machine Translation for a Resource-Poor Language Using Related Resource-Rich Languages. *Journal of Artificial Intelligence Research*, 44. arXiv: 1401.6876.

Sennrich, R., Haddow, B., and Birch, A. (2015a). Improving Neural Machine Translation Models with Monolingual Data. *arXiv:1511.06709 [cs]*. arXiv: 1511.06709.

Sennrich, R., Haddow, B., and Birch, A. (2015b). Neural Machine Translation of Rare Words with Subword Units. *arXiv:1508.07909 [cs]*. arXiv: 1508.07909.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to Sequence Learning with Neural Networks. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Theano, D. T. (2016). Theano: A Python framework for fast computation of mathematical expressions. *arXiv:1605.02688 [cs]*. arXiv: 1605.02688.

Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014). How transferable are features in deep neural networks? *arXiv:1411.1792 [cs]*. arXiv: 1411.1792.

Zeiler, M. D. (2012). ADADELTA: An Adaptive Learning Rate Method. *arXiv:1212.5701 [cs]*. arXiv: 1212.5701.

Zoph, B., Yuret, D., May, J., and Knight, K. (2016). Transfer Learning for Low-Resource Neural Machine Translation. *arXiv:1604.02201 [cs]*. arXiv: 1604.02201.

# Which Words Matter in Defining Phrase Reorderings in Statistical Machine Translation?

**Hamidreza Ghader** h.ghader@uva.nl
**Christof Monz** c.monz@uva.nl
Informatics Institute, University of Amsterdam, The Netherlands

**Abstract**

Lexicalized and hierarchical reordering models use relative frequencies of fully lexicalized phrase pairs to learn phrase reordering distributions. This results in unreliable estimation for infrequent phrase pairs which also tend to be longer phrases. There are some smoothing techniques used to smooth the distributions in these models. But these techniques are unable to address the similarities between phrase pairs and their reordering distributions. We propose two models to use shorter sub-phrase pairs of an original phrase pair to smooth the phrase reordering distributions. In the first model we follow the classic idea of backing off to shorter histories commonly used in language model smoothing. In the second model, we use syntactic dependencies to identify the most relevant words in a phrase to back off to. We show how these models can be easily applied to existing lexicalized and hierarchical reordering models. Our models achieve improvements of up to 0.40 BLEU points in Chinese-English translation compared to a baseline which uses a regular lexicalized reordering model and a hierarchical reordering model. The results show that not all the words inside a phrase pair are equally important in defining phrase reordering behavior and shortening towards important words will decrease the sparsity problem for long phrase pairs.

## 1 Introduction

The introduction of lexicalized reordering models (LRMs) (Tillmann, 2004; Koehn et al., 2005) was a significant step towards better reordering by modeling the orientation of the current phrase pair with respect to the previously translated phrase. LRMs score the order in which phrases are translated by using a distribution of distinguished orientations conditioned on phrase pairs. Typically, the set of orientations consists of: monotone (M), swap (S) and discontinuous (D). However, LRMs are limited to reorderings of neighboring phrases only. Galley and Manning (2008) proposed a hierarchical phrase reordering model (HRM) for more global reorderings.

LRMs and HRMs both use relative frequencies observed in a parallel corpus to estimate the distribution of orientations conditioned on phrase pairs. As a result, they both suffer from the same problem of estimating reliable distributions for cases that occur rarely during training and therefore have to resort to smoothing methods to alleviate sparsity issues.

Cherry (2013) builds on top of HRMs and proposes a sparse feature approach which uses word clusters instead of fully lexicalized forms for infrequent words to decrease the effect of sparsity on the estimated model.

In this paper, we propose two types of approaches to use the most influential words from inside the original phrase pairs to estimate better orientation distributions for infrequent phrase pairs that takes phrase pair similarity more into account. In the first approach, we define a back-off model to shorten towards important words inside the original phrase pairs following the idea

| | | | | Dirichlet Smoothed | | | |
|---|---|---|---|---|---|---|---|
| | Source | Target | Freq | M | S | DL | DR |
| a | 中国 政府 | chinese government | 2834 | 0.216 | 0.034 | 0.315 | 0.433 |
| b | 日本 政府 | japanese government | 580 | 0.157 | 0.039 | 0.299 | 0.503 |
| c | 尼泊尔 政府 | nepalese government | 11 | 0.525 | 0.001 | 0.101 | 0.370 |

| | | | | Recursive Map Smoothed | | | |
|---|---|---|---|---|---|---|---|
| | Source | Target | Freq | M | S | DL | DR |
| a | 中国 政府 | chinese government | 2834 | 0.216 | 0.034 | 0.315 | 0.432 |
| b | 日本 政府 | japanese government | 580 | 0.158 | 0.039 | 0.300 | 0.501 |
| c | 尼泊尔 政府 | nepalese government | 11 | 0.400 | 0.009 | 0.202 | 0.388 |

Table 1: Examples of similar phrase pairs and their orientation probabilities using Dirichlet (Equation 1) and Recursive MAP (Equation 2) smoothing. M=monotone, S=swap, DL=discontinous left, and DR=discontinous right.

of back-off models in language model smoothing. This is, to some extent, complementary to the HRM in the sense of using smaller phrase pairs to make better prediction. The difference is that within HRMs smaller phrase pairs are merged into longer blocks when possible, while we propose to use shorter forms of phrase pairs when possible. In the second approach, we propose to produce generalized forms of original, fully lexicalized phrase pairs by including important words and marginalizing others allowing for smoothed distributions that better capture the true distributions of orientations. Here, we use syntactic dependencies from the original phrase pair to generalize and shorten in a more linguistically informed way.

The main contribution of this paper includes new methods to use shortened and generalized forms of a phrase pair to smooth the original phrase orientation distributions. We show that our smoothing approaches result in improvements in a phrase-based machine translation system, even when compared against a strong baseline using both LRM and HRM together. These methods do not require any changes to the decoder and do not lead to any additional computations during the decoding.

Our second contribution is a deeper analysis showing that orientation distributions conditioned on long phrase pairs typically depend on a few words within phrase pairs and not the whole lexicalized form. This supports and adds to the sparse reordering features (Cherry, 2013).

## 2 Problem Definition

In order to smooth the original maximum likelihood estimation, LRMs originally back off to the general distribution over orientations:

$$P(o \mid \bar{f}, \bar{e}) = \frac{C(o, \bar{f}, \bar{e}) + \sigma P(o)}{\sum_{o'} C(o', \bar{f}, \bar{e}) + \sigma} \quad (1)$$

which is also known as Dirichlet smoothing, where $\sigma P(o)$ denotes the parameters of the Dirichlet prior that maximizes the likelihood of the observed data, $C(o, \bar{f}, \bar{e})$ refers to the number of times a phrase pair cooccurs with orientation $o$, and $\sigma$ is the *equivalent sample size*, i.e., the number of samples required from $P(o)$ to reflect the observed data (Smucker and Allan, 2005). Cherry (2013) and Chen et al. (2013) introduce recursive MAP smoothing, which makes use of more specific priors by recursively backing off to orientation priors, see Equation 2. While recursive MAP smoothing factorizes phrase pairs into source and target phrases, it still considers the phrases themselves as fixed units.

$$
\begin{aligned}
P(o \mid \bar{f}, \bar{e}) &= \frac{C(o, \bar{e}, \bar{f}) + \alpha_s P_s(o \mid \bar{f}) + \alpha_t P_t(o \mid \bar{e})}{\sum_{o'} C(o', \bar{e}, \bar{f}) + \alpha_s + \alpha_t} \\
P_s(o \mid \bar{f}) &= \frac{\sum_{\bar{e}} C(o, \bar{f}, \bar{e}) + \alpha_g P_g(o)}{\sum_{o', \bar{e}} C(o', \bar{f}, \bar{e}) + \alpha_g} \\
P_t(o \mid \bar{e}) &= \frac{\sum_{\bar{f}} C(o, \bar{f}, \bar{e}) + \alpha_g P_g(o)}{\sum_{o', \bar{f}} C(o', \bar{f}, \bar{e}) + \alpha_g} \\
P_g(o) &= \frac{\sum_{\bar{f}, \bar{e}} C(o, \bar{f}, \bar{e}) + \alpha_u / 3}{\sum_{o', \bar{f}, \bar{e}} C(o', \bar{f}, \bar{e}) + \alpha_u}
\end{aligned}
\tag{2}
$$

To better understand what kind of information is ignored by both of the aforementioned smoothing methods, consider the phrase pairs and their corresponding distributions given in Table 1, for which we would expect similar distributions. The phrase pairs in rows (a) and (b) are frequently observed during training, resulting in reliable estimates. On the other hand, the phrase pair in row (c) is infrequent, leading to a very different distribution, due to the smoothing prior, while being semantically and syntactically close to (a) and (b). In Table 1, we can also observe that recursive MAP smoothing results in slightly more similar distributions compared to plain Dirichlet smoothing but the overall differences remain noticeable.

In this paper, we argue that in order to obtain smoother reordering distributions for phrase-pairs such as the ones in Table 1, one has to take phrase-internal information into account.

## 3 Related Work

The problem of data sparsity of training LRMs has first been addressed by Nagata et al. (2006) who propose to use POS tags and word clustering methods and distinguish the first or last word of a phrase, based on the language, as the head of a phrase.

Somewhat complementary to our work, Galley and Manning (2008) introduced hierarchical reordering models that group phrases occurring next to the current phrase into blocks, ignoring the internal derivation within a block, which biases orientations more towards monotone and swap. At the same time, orientations are still conditioned on entire phrase pairs, which means that their approach suffers from the same sparsity problems as LRMs. This problem has been more directly addressed by Cherry (2013) who uses unsupervised word classes for infrequent words of the phrase pairs in the form of sparse features. Like (Nagata et al., 2006), the first and last words of phrase pair are used as features in his model. Unfortunately, this approach also introduces thousands of additional sparse features, many of which have to be extracted *during* decoding, requiring changes to the decoder as well as a sizable tuning set.

Durrani et al. (2014) investigate the effect of generalized word forms on reordering in an n-gram-based operation sequence model, where they use different generalized representations including POS tags and unsupervised word classes to generalize reordering rules to similar cases with unobserved lexical operations.

While the approaches above use discrete representations, (Li et al., 2014) propose a discriminative model using continuous space representations of phrase pairs to address data sparsity problems. They train a neural network classifier based on recursive auto-encoders to generate vector space representations of phrase pairs and base reordering decision on those representations. They apply their model as an additional hypergraph reranking step since direct integration into the decoder would make hypothesis recombination more complex and substantially increase the size of the search space.

In addition to reordering models, several approaches have used word classes to improve other models within a statistical machine translation system, including translation (Wuebker

et al., 2013) and language models, where the problem of data sparsity is particularly exacerbated for morphologically rich target languages (Chahuneau et al., 2013; Bisazza and Monz, 2014).

## 4 Model Definition

In this section, we propose two different models which use different words as source of information to better estimate reordering distributions of sparse phrase pairs. Each model uses a different generalization scheme to obtain less sparse but still informative representations.

### 4.1 Interpolated Back-off Sub-phrases

In n-gram language modeling shorter n-grams have been used to smooth the probability distributions of higher order sparse n-grams. Lower order n-grams form the basis of Jelinek-Mercer, Katz, Witten-Bell and absolute discount smoothing methods (Chen and Goodman, 1999). For instance, Jelinek-Mercer smoothing linearly interpolates distributions of lower orders to smooth the distributions of higher order n-grams.

We use this as a motivation that shorter phrase pairs in lexicalized reordering models could play the role of lower-order n-grams in language model smoothing. But while backing off is obvious in language modeling, it is not straightforward in the context of lexicalized reordering models as there are several plausible ways to shorten a phrase pair, which is further complicated by the internal word alignments of the phrase pairs.

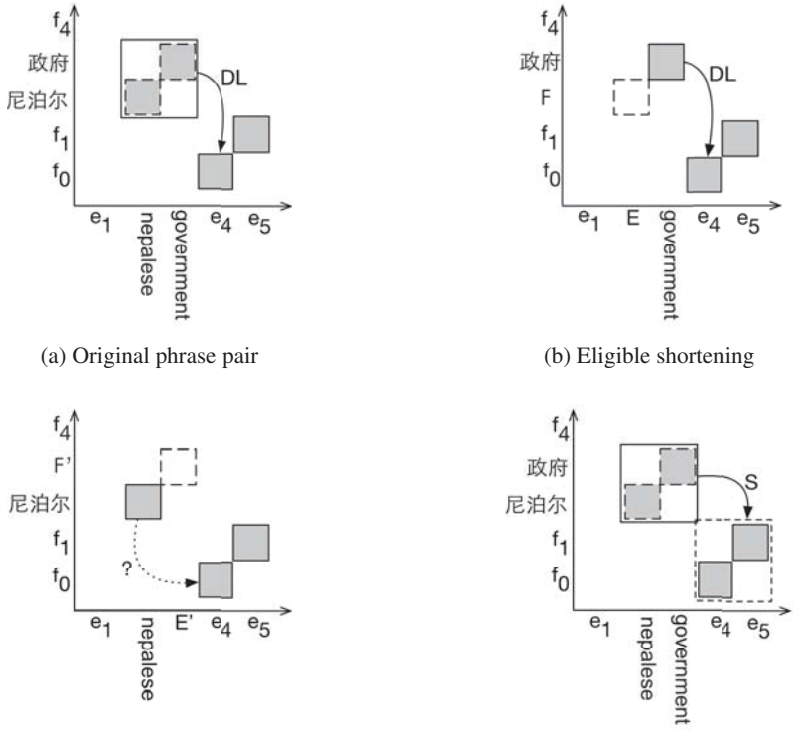



Figure 1: Backing off to shorter phrase pairs using eligible sub-phrase pairs (b). For comparison, we also include an example of a grouping of phrases as done in HRM (d).

The example in Figure 1 illustrates how sub-phrase pairs (Figure 1b and 1c) of the longer

phrase pair (Figure 1a) can be used to estimate the *discontinuous left* orientation for a longer, infrequent phrase pair. Following the strategy within language modeling to back off to shorter n-grams, we back off to the sub-phrase pairs that are consistent with the inside alignment of the longer phrase pair and provide a shorter and less sparse history. In this example, the number of times that sub-phrase pair (政府, government), see Figure 1b, appears with a *discontinuous left* jump of the length of the previous phrase pair for the next translation is considered when estimating the *discontinuous left* orientation for the longer phrase pair. On the other hand, the sub-phrase pair (尼泊尔, nepalese), see Figure 1c, cannot be used to predict a future *discontinuous left* of the long phrase pair, as there is no direct way to connect it to ($f_0$, $e_4$). The difference between this model and HRM can be seen in Figure 1d. HRM groups small phrase pairs from the context into longer blocks and determines the orientation with respect to grouped block, while our model looks into the phrase pair itself and uses possible shortenings to better estimate the orientation distribution conditioned on the original phrase pair. Our model can be applied to HRMs as well as LRMs to estimate a better distribution for long infrequent phrase pairs.

In order to provide a formal definition which sub-phrase pairs to consider when backing off, let us assume that $A$ is the set of alignment connections between the source $\bar{f}$ and target $\bar{e}$ side of a longer phrase pair. The set of eligible sub-phrase pairs, $E_{\bar{f},\bar{e}}$, is defined as follows:

$$E_{\bar{f},\bar{e}} = \{(\bar{f}^{[l,k]}, \bar{e}^{[l',n]}) | \; 1 \leqslant k \leqslant m, \; 0 \leqslant l \leqslant k, 0 \leqslant l' \leqslant n \text{ and } (\bar{f}^{[l,k]}, \bar{e}^{[l',n]}) \text{ consistent with } A \text{ if } l > 0 \wedge l' > 0 \; \} \tag{3}$$

where $\bar{f}^{[l,k]}$ is a sub-phrase of $\bar{f}$ with length $l$ which ends at the $k$th word of $\bar{f}$, $m$ and $n$ are the lengths of $\bar{f}$ and $\bar{e}$ respectively and the consistency with the alignment is ensured by the following three conditions (Koehn et al., 2005):

1. $\exists e_i \in \bar{e}^{[l',n]}, f_j \in \bar{f}^{[l,k]} : (i,j) \in A$
2. $\forall e_i \in \bar{e}^{[l',n]} : (i,j) \in A \Rightarrow f_j \in \bar{f}^{[l,k]}$
3. $\forall f_i \in \bar{f}^{[l,k]} : (i,j) \in A \Rightarrow e_i \in \bar{e}^{[l',n]}$

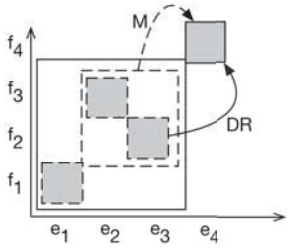



Figure 2: The distributions needed to estimate the conditional probability $\hat{p}(M \mid f_1 f_2 f_3, e_1 e_2 e_3)$ include $p(DR \mid f_2, e_3)$ and $p(M \mid f_2 f_3, e_2 e_3)$

Considering Figure 2, it is clear why ($\bar{f}^{[1,2]}$, $\bar{e}^{[1,3]}$) and ($\bar{f}^{[2,3]}$, $\bar{e}^{[2,3]}$) are considered eligible shortenings. Other possible shortenings such as ($\bar{f}^{[2,2]}$, $\bar{e}^{[3,3]}$) and ($\bar{f}^{[1,3]}$, $\bar{e}^{[1,2]}$) either violate the consistency conditions or do not run up to the end of the target side of the original phrase pair as in the definition above. Note that $n$ is a constant here and $\bar{e}^{[l',n]}$ means that all sub-phrase pairs must finish at the end of the target side of the original phrase pair. Otherwise one cannot directly determine the orientation with respect to the next phrase pair.

In our model, we compute the smoothed orientation distribution conditioned on a phrase pair by linearly interpolating the distribution of all eligible sub-phrases:

$$\hat{P}(o \mid \bar{f}, \bar{e}) = \sum_{(\bar{f}^{[l,k]}, \bar{e}^{[l',n]}) \in E_{\bar{f},\bar{e}}} \lambda_{l,l'} P(\Omega(\bar{f}^{[l,k]}, \bar{f}, o) \mid \bar{f}^{[l,k]}, \bar{e}^{[l',n]}) \quad (4)$$

where $E_{\bar{f},\bar{e}}$ is the set of eligible sub-phrase pairs and $\bar{f}^{[l,k]}$ indicates a sub-phrase of $\bar{f}$ with length $l$ ending at the $k$th word of $\bar{f}$, $\bar{e}^{[l',n]}$ is a sub phrase of $\bar{e}$ with the length of $l'$ which ends at the last word of $\bar{e}$ and the function $\Omega(\bar{f}^{[l,k]}, \bar{f}, o)$ returns the correct orientation considering the position of source sub-phrase $\bar{f}^{[l,k]}$ with respect to either end of the source phrase $\bar{f}$ and orientation $o$.

In order to compute the linear interpolation over the conditional distributions of the sub-phrase pairs, we have to determine the weight of each term in the linear interpolation (Equation 4). Here, we use expectation-maximization (EM) over a held-out data set, which is word-aligned using GIZA++ (Och and Ney, 2003). We extract phrase pairs using a common phrase extraction algorithm (Koehn et al., 2005) and count the number of occurrences of orientations for each phrase pair. These counts are used with unsmoothed reordering probabilities learned over the training data to compute the likelihood over the held-out data. We designed the EM algorithm to learn a set of lambda parameters for each length combination of the original phrase pairs. To reduce the number of parameters, we assume that all sub-phrases with the same length on source and target side share the same weight. This model is referred to as the BackOff model in the remainder of this paper.

### 4.2 Recursive Back-off MAP Smoothing

Above we used linear interpolation to estimate the final distribution from the orientation distributions of shorter sub-phrase pairs. Here we investigate another method aiming to affect the distributions of frequent phrase pairs to a lesser extent than those of non-frequent ones.

To this end, we use recursive MAP smoothing to estimate the distribution of the original phrase pair. In linear interpolation, all phrase pairs with the same length will get the same portion of their estimated distribution from their sub-phrases. On the other hand, for more frequent phrase pairs, the maximum likelihood distribution of the phrase pair itself is more reliable than the distributions of its sub-phrase pairs. Thus, a model relying more on the distribution of the original phrase pairs for frequent phrase pairs would be desirable.

To achieve this, we use a formulation similar to recursive MAP smoothing (Equation 2) with recursively backing off to the distributions of shorter sub-phrase pairs. At each recursion step we use the distribution of the longest sub-phrase pair as the prior distribution. Taking our definition for eligible sub-phrase pairs into account (Equation 3), all other sub-phrase pairs of the original phrase pair are sub-phrase pairs of the longest sub-phrase pair, in the case that the original phrase pair does not include unaligned words. For cases including unaligned words like the example in Figure 3, there could be sub-phrases where none of them is the sub-phrase of the other. In these cases we include the distributions of all those sub-phrases with the same *equivalent sample size* as the prior distributions. The estimated probability distribution of a phrase pair ($\bar{f}$,$\bar{e}$) is defined as follows:

$$\hat{P}(o \mid \bar{f}, \bar{e}) = \frac{C(o, \bar{f}, \bar{e}) + \sum_{(\bar{f}_L, \bar{e}_L) \in L_{\bar{f},\bar{e}}} \alpha \hat{P}(\Omega(\bar{f}_L, \bar{f}, o) \mid \bar{f}_L, \bar{e}_L)}{\sum_{o \in O} C(o, \bar{f}, \bar{e}) + \sum_{(\bar{f}_L, \bar{e}_L) \in L_{\bar{f},\bar{e}}} \alpha} \quad (5)$$

where $L_{\bar{f},\bar{e}}$ refers to the set of eligible sub-phrase pairs of $(\bar{f}, \bar{e})$ that are not sub-phrase pairs of each other and which is defined as follows:

$$L_{\bar{f},\bar{e}} = \{(\bar{f}', \bar{e}') \in E_{\bar{f},\bar{e}} \backslash \{(\bar{f}, \bar{e})\} \mid \neg \exists (\bar{f}'', \bar{e}'') \in E_{\bar{f},\bar{e}} \backslash \{(\bar{f}, \bar{e}), (\bar{f}', \bar{e}')\} : \bar{f}' \sqsubseteq \bar{f}'' \wedge \bar{e}' \sqsubseteq \bar{e}''\} \quad (6)$$

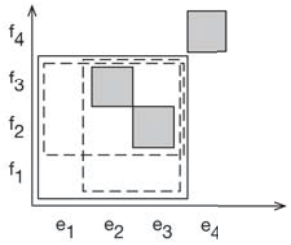



Figure 3: Illustration of sub-phrases where neither of the two pairs $(f_1 f_2 f_3, e_2 e_3)$ and $(f_2 f_3, e_1 e_2 e_3)$ of the original phrase pair $(f_1 f_2 f_3, e_1 e_2 e_3)$ is a sub-phrase of the other.

Here, $\bar{f}' \sqsubseteq \bar{f}''$ means that $\bar{f}'$ is a sub-phrase of or equal to $\bar{f}''$. As a result, $L_{\bar{f},\bar{e}}$ is the set of longest eligible sub-phrase pairs of original phrase pair where none of them is a sub-phrase of the others. For $E_{\bar{f},\bar{e}}$ see Equation 3. $O$ is the set of possible orientations. Note that we refer to this model shortly as RecursiveBackOff model from now on. We also refer to both this model and the BackOff model described in the previous section as back-off models.

### 4.3 Dependency Based Generalization

The methods described so far generalize the original phrase pairs by shortening towards the last aligned words as the most important words to define the reordering behavior of a phrase pair. In the remainder of this section, we use dependency parses to define how to generalize the original phrase pair and shorten towards important words.

Head-driven hierarchical phrase based translation (Li et al., 2012) suggests that using heads of phrases can be beneficial for better reordering in general. In our work, we define the heads of a phrase to be its *exposed heads*. Given a dependency parse, the exposed heads are all words inside a subsequence that are modifying a word outside it. Figure 4 shows an example of exposed heads in a phrase pair. The highlighted words are the exposed heads of the phrase pair. Exposed heads have been used in multiple linguistically motivated approaches as strong predictors of the next word in structured language models (Chelba and Jelinek, 2000; Garmash and Monz, 2015) and the next rule in a hierarchical translation system (Li et al., 2012).

In our model, besides training a regular lexicalized or hierarchical reordering model on surface forms of phrases, we train another reordering model which keeps the exposed heads lexicalized and replaces the remaining words in a phrase pair by a generalized representation. Assume that $RE$ is the set of dependency relations in the dependency parse tree of sentence $S$.

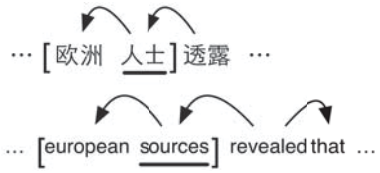



Figure 4: Examples of exposed heads in a Chinese-English phrase pair (between square brackets). The underlined words are the exposed heads since they have an incoming dependency originating outside of the phrase.

We consider each relation as an ordered pair $(w'_l, w_k)$ which means $w$ at index $k$ of sentence $S$ modifies $w'$ at index $l$. In addition, assuming $f_i^j$ is a phrase in $S$, starting from the $i$th and

ending with the $j$th word in sentence $S$, then the generalization $w_G$ of a word is:

$$w_G = \begin{cases} w_k & \text{if } w_k \in f_i^j, \exists (w'_l, w_k) \in RE : \\ & l < i \text{ or } l > j \text{ or } w' = ROOT \\ Gen(w_k) & \text{otherwise} \end{cases}$$

Here, $k$ and $l$ are indices of words $w$ and $w'$ in sentence $S$ and $ROOT$ is the root of the dependency parse of sentence $S$. The function $Gen(w)$ returns a generalization form for word $w$. We define this function in three different ways to create three different models.

1. $Gen(w_k)$ = POS_tag($w_k$)
2. $Gen(w_k)$ = <mod> if $w_{k-1}$ is not equal to <mod> and nothing otherwise
3. $Gen(w_k)$ remove $w_k$.

The question is how to use these generalizations to improve the estimation of the reordering distribution for each phrase pair. Our first model applies a generalization to the bilingual training data and creates a reordering model similar to the regular lexicalized reordering model, but based on relative frequency of generalized phrase pairs. In practice, a phrase pair may have multiple generalizations due to different dependency parses in different contexts. Since it is difficult to produce a dependency parse for the target side during decoding, we assume that a phrase pair will always have one possible generalization. Under this assumption, we can approximate the orientation distribution of a phrase pair to be:

$$\hat{P}(o \mid \bar{f}, \bar{e}) = P(o \mid \bar{f_G}, \bar{e_G}) \tag{7}$$

We can use the orientation distribution of a generalization as our estimate of the distribution of a phrase pair that produces the generalization. Here, $\bar{f_G}$ and $\bar{e_G}$ are word by word generalizations of $\bar{f}$ and $\bar{e}$. In case of multiple generalizations we use the one maximizing $P(\bar{f_G}, \bar{e_G} \mid \bar{f}, \bar{e})$. Depending on which of the three definitions for $Gen(w_k)$ we use, we name our models as PMLH (POS Modifiers Lexicalized Heads), MMLH (Merged Modifiers Lexicalized Heads), and LH (Lexicalized Heads) respectively.

As an alternative model, we propose to use the generalized distributions as a prior distribution in Dirichlet smoothing, where the distribution of each phrase pair is smoothed with the distribution of its generalized form. This should result in more accurate distributions since it affects the distributions of frequent phrase pairs to a lesser extent.

## 5 Experiments

We evaluate our models for Chinese-to-English translation. Our training data consists of the parallel and monolingual data released by NIST's OpenMT campaign, with MT04 used for tuning and news data from MT05 and MT06 for testing, see Table 2. Case-insensitive BLEU (Papineni et al., 2002) and translation error rate (TER) (Snover et al., 2006) are used as evaluation metrics.

### 5.1 Baseline

We use an in-house implementation of a phrase-based statistical machine translation system similar to Moses (Koehn et al., 2007), including the commonly used translation, lexical weighting, language, lexicalized reordering, and hierarchical reordering models. We use both lexicalized and hierarchical reordering models together, since this is the best model reported in (Galley and Manning, 2008) and our smoothing methods can be easily applied to the both models. Word alignments are produced using GIZA++ (Och and Ney, 2003), using grow-diag-final-and (Koehn et al., 2003). A 5-gram language model is trained on the English Gigaword corpus

| Corpus | Lines | Tokens(ch) | Tokens(en) |
|---|---|---|---|
| train | 937K | 22.3M | 25,9M |
| MT04 (dev) | 1,788 | 49.6K | 59.2K |
| MT05 (test) | 1,082 | 30.3K | 35.8K |
| MT06 (test) | 1,181 | 29.7K | 33.5K |

Table 2: Statistics for the Chinese-English bilingual corpora used in all experiments. Token counts for the English side of dev and test sets are averaged across all references.

with 1.6B tokens using interpolated, modified Kneser-Ney smoothing. The lexicalized and the hierarchical reordering models are trained with relative and smoothed frequencies using Dirichlet smoothing (Equation 1), for both left-to-right and right-to-left directions distinguishing four orientations: monotone (M), swap (S), discontinuous left (DL), and discontinuous right (DR). Feature weights are tuned using PRO (Hopkins and May, 2011) and statistical differences are computed using approximate randomization (Riezler and Maxwell, 2005).

In addition to the baseline, we reimplemented the 2POS model by Nagata et al. (2006), which uses the POS tag of the first and last words of a phrase pair to smooth the reordering distributions. The 2POS model is used in combination with the baseline lexicalized and hierarchical reordering models. Comparing our models to the 2POS model allows us to see whether backed-off sub-phrases and exposed heads of phrase pairs yield better performance than simply using the first and last words.

### 5.2 Comparison Systems and Results

We compare the baseline to the models described in Section 4. For all systems other than the baseline, the lexicalized and the hierarchical reordering models are replaced by the corresponding smoothed models. When computing the RecursiveBackOff model (Section 4.2), using Equation 5, we set the value of $\alpha$ to 10, following Cherry (2013) and Chen et al. (2013).

For the dependency-based model, we use the dependency parses of the source and the target side of the training corpus. The Stanford Neural-network dependency parser (Chen and Manning, 2014) is used to generate parses for both sides of the training corpus. From a dependency parse, we extract the smallest subtree that includes all incoming and outgoing relations of the words of a phrase. This is done for both the source and the target side phrases. Considering these subtrees, all words with an incoming connection from outside are exposed heads.

The experimental results for all models are shown in Table 3. As one can see, all our models achieve improvements in terms of BLEU on the test sets. The improvements for our back-off models are only significant for RecursiveBackOff over MT06 and MT05+MT06. The improvements over MT05 by our dependency-based shortenings are statistically significant for all models except PMLH. In the case of MT06, only the improvements resulting from MMLH are not statistically significant. However, both the PMLH and the MMLH model achieve the same improvements over MT05 and MT06 combined, and both are statistically significant. The LH model performs better than these models and also achieves higher improvement on the merged data. This model generalizes much more than the other models and is the only model that changes the distributions of single word phrases which are among the most frequently used phrase pairs. However, for frequent phrase pairs, being mapped to the same generalization form can be potentially harmful. In order to be able to control the effect of the model on the phrase pairs based on their frequency, we use the distributions in LH as a prior distribution with Dirichlet smoothing (Equation 1). This results in the LHSmoothed model shown in Table 3 which achieves the best improvements over both MT05 and MT06.

| Model | MT05 | | MT06 | | MT05 + MT06 | | |
|---|---|---|---|---|---|---|---|
| | BLEU↑ | TER↓ | BLEU↑ | TER↓ | BLEU↑ | TER↓ | RIBES↑ |
| Lex+Hrc | 32.25 | 60.13 | 33.00 | 57.17 | 32.84 | 58.62 | 79.24 |
| Nagata's 2POS | 32.20 | 60.31 | 33.13 | 57.07 | 32.87 | 58.66 | 79.11 |
| BackOff | *32.40* | 60.45 | *33.10* | *57.00* | *33.00* | 58.69 | 79.07 |
| RecursiveBackOff | *32.37* | 60.29 | *33.34*$^{\triangle,-}$ | *57.08* | *33.05*$^{\triangle,-}$ | 58.65 | *79.28* |
| PMLH | *32.41* | *60.08* | *33.26*$^{\triangle,-}$ | *57.05* | *33.04*$^{\triangle,-}$ | *58.53* | *79.26* |
| MMLH | *32.62*$^{\blacktriangle,\triangle}$ | *59.84*$^{-,\blacktriangle}$ | *33.18* | *56.91*$^{\triangle,-}$ | *33.04*$^{\triangle,-}$ | *58.35*$^{\blacktriangle,\triangle}$ | *79.43* |
| LH | *32.64*$^{\blacktriangle,\triangle}$ | *59.85*$^{\triangle,\blacktriangle}$ | *33.26*$^{\triangle,-}$ | *56.85*$^{\blacktriangle,-}$ | *33.11*$^{\blacktriangle,\triangle}$ | *58.32*$^{\blacktriangle,\blacktriangle}$ | *79.34* |
| LHSmoothed | **32.65**$^{\blacktriangle,\blacktriangle}$ | **59.80**$^{\triangle,\blacktriangle}$ | **33.38**$^{\triangle,-}$ | **56.77**$^{\blacktriangle,\triangle}$ | **33.20**$^{\blacktriangle,\triangle}$ | **58.25**$^{\blacktriangle,\blacktriangle}$ | *79.35* |

Table 3: Model comparison using BLEU, TER (lower is better), and RIBES over news data, which is combination of newswire and broadcast_news in case of MT06 and just newswire for MT05. Scores better than the baseline are in italics. ▲ and △ indicate statistically significant improvements at $p < 0.01$ and $p < 0.05$, respectively. The left hand side ▲ or △ is with respect to Lex+Hrc and the right hand side ones with respect to Nagata's 2POS. PMLH refers to the model using POS tags for modifiers and keeps exposed heads lexicalized. MMLH merges modifiers and keeps exposed heads lexicalized. LH removes modifiers. LHSmoothed uses the LH model with Dirichlet smoothing.

In addition to BLEU, we also report results using TER. Results for TER are in line with BLEU. BLEU and TER are general translation quality metrics, which are known to be not very sensitive to reordering changes (Birch et al., 2010). To this end we also include RIBES (Isozaki et al., 2010), a reordering-specific metric that is designed to directly address word-order differences between hypothesis and reference translation in translation tasks with long distant reordering language pairs and is highly sensitive to word-order mistakes.

### 5.3 Analysis

The improvements achieved by our BackOff and RecursiveBackOff methods show that these models capture some useful generalizations by shortening the phrase pairs towards the last aligned words in the target side as the most important words. The difference between the two models indicates that shortening is less beneficial for frequent phrase pairs and shorter phrase pairs which are less affected by lower-order distributions.

The improvements achieved by our generalization models support our hypothesis that not all words inside a phrase pair have the same impact on the reordering properties of the phrase pair as a whole. The experimental results for the PMLH model show that the lexicalized form of modifier words inside a phrase pair may just have the negative effect of increasing data sparsity.

Observing the improvements achieved by MMLH, we can go further and say that even the number of modifiers of an exposed head in a phrase does not influence reordering properties of a phrase pair. The improvements achieved by our LH model show not only that the number of modifiers but also the mere presence or absence of them does not significantly influence the reordering properties of a phrase pair.

One thing to bear in mind is that the PMLH and MMLH models do not change the distribution of single word phrase pairs which are mostly frequent phrase pairs, while the LH model does change these distributions as well. With the LHSmoothed model we have controlled this effect and decreased it to be negligible for frequent single word phrase pairs. However, it still changes the distributions of infrequent single word phrase pairs. Comparing the results of LH and LHSmoothed in Table 3, we suspect that the difference between the models for MT06 is

| Source Length | Target Length | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1 | 8232 | 2719 | 879 | 269 | 89 | 18 | 7 |
| 2 | 2344 | 1777 | 1055 | 410 | 158 | 58 | 18 |
| 3 | 316 | 390 | 376 | 252 | 100 | 61 | 29 |
| 4 | 42 | 46 | 97 | 63 | 29 | 28 | 14 |
| 5 | 2 | 3 | 11 | 11 | 10 | 8 | 12 |
| 6 | 0 | 1 | 1 | 3 | 3 | 5 | 2 |
| 7 | 0 | 0 | 0 | 3 | 1 | 1 | 1 |

Table 4: Number of times that phrase pairs with different lengths and a frequency of less than 10 in the training data have been used during test on MT06 by the baseline. Phrase pairs occurring less than 10 times account for 72% of all phrases used during decoding of MT06.

| | |
|---|---|
| Source | ... 由于 东京 和 汉城 当局 均 寄望 能 于 二〇〇五年 底 前 签署 自由 贸易 协定 ... |
| Baseline | ... tokyo and seoul authorities are to be placed in 2005 before the end of the signing of a free trade agreement ... |
| LHSmoothed | ... tokyo and seoul authorities both in the hope of signing a free trade agreement before the end of 2005 ... |
| Ref | ... tokyo and seoul both hoped to sign a fta agreement by the end of 2005 ... |
| Source | 俄罗斯 多 次 指控 西方 插手 东欧 事务 ... |
| Baseline | russia has repeatedly accused of meddling in the affairs of the western and eastern europe ... |
| LHSmoothed | russia has repeatedly accused western intervention in the eastern european affairs ... |
| Ref | russia has been accusing the west of interfering in the affairs of eastern europe ... |

Table 5: Examples from MT05 illustrating reordering improvements over the baseline.

due to the effect of frequent single word phrase pairs.[1] However, comparing the results of LHSmoothed with other models we can say that even infrequent single word phrase pairs have benefited from the higher generalization level offered by this model. The statistics of infrequent phrase pairs used during testing and their lengths are shown in Table 4, giving an indication of why this model achieves the highest improvements. The table is showing that 41% of the infrequent phrase pairs (frequency $< 10$) used during translating MT06 have length of one in the both sides. So our models other than LH and LHSmoothed can not have neither positive nor negative effect on almost half of the infrequent phrase pairs. This also probably could explain why the back-off models achieve such a little improvements when 75% of infrequent phrase pairs have length $< 3$ in both sides.

In general, our dependency based models change the distributions to a larger extent than our back-off models, where not all long phrase pairs have eligible sub-phrase pairs, while the dependency models more often result in shorter generalizations. Table 5 provides some examples where our LHSmoothed model has improved the translations by better modeling of reorderings. To further the understanding of how the reordering distributions of infrequent phrase pairs used in the examples in Table 5 are affected by our models, we show some examples in Table 6, before and after applying our model. As a result of our model, the phrase pair (寄望, in the hope)

[1] We also used the distributions from PMLH and MMLH as priors in Dirichlet smoothing, but it did not lead to any noticeable changes in the results.

| 寄望 | in the hope | M | S | DL | DR |
|---|---|---|---|---|---|
| Monotone with previous | Baseline | 0.10 | 0.01 | 0.11 | 0.78 |
| | LHSmoothed | 0.28 | 0.01 | 0.01 | 0.70 |
| | Counts in training | 0 | 0 | 0 | 1 |
| 能 于 | of | | | | |
| Discontinuous right with next | Baseline | 0.10 | 0.01 | 0.12 | 0.77 |
| | LHSmoothed | 0.06 | 0.01 | 0.07 | 0.87 |
| | Counts in training | 0 | 0 | 0 | 1 |
| 签署 自由 贸易 协定 | signing a free trade agreement | | | | |
| Discontinuous right with previous | Baseline | 0.10 | 0.02 | 0.68 | 0.20 |
| | LHSmoothed | 0.21 | 0.03 | 0.30 | 0.46 |
| | Counts in training | 0 | 0 | 1 | 0 |

Table 6: Orientation distributions shift of some infrequent phrase pairs that has been used with the correct orientation to produce our translations using LHSmoothed model in Table 5

receives an increase in monotone orientation probability although it has frequency of zero for this orientation. The next phrase pair (能 于, of) receives an increase in discontinuous right probability resulting in the correct usage of this orientation during translation. The most interesting case is the substantial decrease in the probability of discontinuous left and the increase in discontinuous right for the phrase pair (签署 自由 贸易 协定, signing a free trade agreement), even though it has frequency 1 for the former orientation and 0 for the latter. These shifts within the probability distributions lead to the better translation generated by LHSmoothed model for the first example in Table 5.

## 6 Conclusions

We have introduced a novel method that builds on the established idea of backing off to shorter histories, commonly used in language model smoothing, and shown that it can be successfully applied to smoothing of lexicalized and hierarchical reordering models in statistical machine translation. Furthermore, we have shown that not all sub-phrase pairs are influential in that regard. The sub-phrase pair consisting of just exposed heads of a phrase pair tends to be the most important one and most other words inside a phrase pair have negligible influence on reordering behavior. Earlier approaches, such as (Nagata et al., 2006) and (Cherry, 2013), often assume that the last and the first word of a phrase pair are important, but our experiments indicate that exHGposed heads tend to be stronger predictors. We showed that generalized representations of phrase pairs based on exposed heads can help decrease sparsity and result in more reliable reordering distributions.

Considering the analysis of the length of infrequent phrase pairs used during translation, we also conclude that a smoothing model that would be able to further improve the distribution of single word phrase pairs is crucial for achieving higher improvements during translation.

For future work, we plan to investigate the effect surface word forms from outside of a phrase pair can have on reordering. This could further help improve reordering distributions of infrequent single word phrase pairs more effectively as they constitute a large portion of the phrases used during decoding.

**Acknowledgments**

This research was funded in part by the Netherlands Organization for Scientific Research (NWO) under project numbers 639.022.213 and 612.001.218.

## References


Birch, A., Osborne, M., and Blunsom, P. (2010). Metrics for MT evaluation: evaluating reordering. *Machine Translation*, 24(1):15–26.

Bisazza, A. and Monz, C. (2014). Class-based language modeling for translating into morphologically rich languages. In *Proceedings of the 25th Annual Conference on Computational Linguistics (COLING)*, pages 1918–1927.

Chahuneau, V., Schlinger, E., Smith, N. A., and Dyer, C. (2013). Translating into morphologically rich languages with synthetic phrases. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1677–1687.

Chelba, C. and Jelinek, F. (2000). Structured language modeling. *Computer Speech & Language*, 14(4):283–332.

Chen, B., Foster, G., and Kuhn, R. (2013). Adaptation of reordering models for statistical machine translation. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 938–946.

Chen, D. and Manning, C. D. (2014). A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 740–750.

Chen, S. F. and Goodman, J. (1999). An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–393.

Cherry, C. (2013). Improved reordering for phrase-based translation using sparse features. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 22–31.

Durrani, N., Koehn, P., Schmid, H., and Fraser, A. (2014). Investigating the usefulness of generalized word representations in SMT. In *Proceedings of the 25th Annual Conference on Computational Linguistics (COLING)*, pages 421–432.

Galley, M. and Manning, C. D. (2008). A simple and effective hierarchical phrase reordering model. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 848–856.

Garmash, E. and Monz, C. (2015). Bilingual structured language models for statistical machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2398–2408.

Hopkins, M. and May, J. (2011). Tuning as ranking. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1352–1362.

Isozaki, H., Hirao, T., Duh, K., Sudoh, K., and Tsukada, H. (2010). Automatic evaluation of translation quality for distant language pairs. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 944–952.

Koehn, P., Axelrod, A., Mayne, A. B., Callison-Burch, C., Osborne, M., and Talbot, D. (2005). Edinburgh System Description for the 2005 IWSLT Speech Translation Evaluation. In *Proceedings of International Workshop on Spoken Language Translation*.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., et al. (2007). Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the Association for Computational Linguistics*, pages 177–180.

Koehn, P., Och, F. J., and Marcu, D. (2003). Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, pages 48–54.

Li, J., Tu, Z., Zhou, G., and van Genabith, J. (2012). Head-driven hierarchical phrase-based translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 33–37.

Li, P., Liu, Y., Sun, M., Izuha, T., and Zhang, D. (2014). A neural reordering model for phrase-based translation. In *Proceedings of the 25th Annual Conference on Computational Linguistics (COLING)*, pages 1897–1907.

Nagata, M., Saito, K., Yamamoto, K., and Ohashi, K. (2006). A clustered global phrase reordering model for statistical machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 713–720.

Och, F. J. and Ney, H. (2003). A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

Riezler, S. and Maxwell, J. T. (2005). On some pitfalls in automatic evaluation and significance testing for MT. In *Proceedings of the Association for Computational Linguistics Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 57–64.

Smucker, M. D. and Allan, J. (2005). An investigation of dirichlet prior smoothing's performance advantage. Technical Report IR-391, The University of Massachusetts, The Center for Intelligent Information Retrieval.

Snover, M., Dorr, B., Schwartz, R., Micciulla, L., and Makhoul, J. (2006). A study of translation edit rate with targeted human annotation. In *Proceedings the 7th Biennial Conference of the Association for Machine Translation in the Americas (AMTA)*, pages 223–231.

Tillmann, C. (2004). A unigram orientation model for statistical machine translation. In *Proceedings of the 2004 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 101–104.

Wuebker, J., Peitz, S., Rietig, F., and Ney, H. (2013). Improving statistical machine translation with word class models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1377–1381.

# Translation of Unknown Words in Low Resource Languages

**Biman Gujral** bgujral1@jhu.edu
**Huda Khayrallah** huda@jhu.edu
**Philipp Koehn** phi@jhu.edu
Department of Computer Science, Center for Language and Speech Processing
Johns Hopkins University, Baltimore, MD, 21218, USA

**Abstract**

We address the problem of unknown words, also known as out of vocabulary (OOV) words, in machine translation of low resource languages. Our technique comprises a combination of methods, inspired by the common OOV types observed. We also design evaluation techniques for measuring coverage of OOVs achieved and integrate the new translation candidates in a Statistical Machine Translation (SMT) system. Experimental results on Hindi and Uzbek show that our system achieves a good coverage of OOV words. We show that our methods produced correct candidates for 50% of Hindi OOVs and 30% of Uzbek OOVs, in scenarios that have 1 and 3 OOVs per sentence. This offers a potential for improvement of translation quality for languages that have limited parallel data available for training.

## 1 Introduction

Key factors in the performance of Statistical Machine Translation (SMT) systems are the volume and domain of available parallel data. Low resource languages lack sufficient amounts of parallel data as well as advanced linguistic tools for analysis. This results in a high percentage of out of vocabulary (OOV) words (unknown words, not seen in the parallel data). The default solution is to copy the unknown word in the translated output, which may work for named entities but only if the two languages share a script.

We propose a system for generating translation candidates for unknown words. The system uses a combination of methods when translating from a low resource language into English. Each method targets different types of unknown words. These can be named entities, borrowed words, compound words, spelling or morphological variants of seen words or content words unrelated to any seen word. We do not employ any language-specific tools to ensure that our system is applicable across all languages, even resource poor ones that may not have specialized tools. New methods can be added or existing ones pruned, based on the properties of the source language. In our current system, we use (i) transliteration, (ii) Levenshtein distance-based search, and (iii) Canonical Correlation Analysis on word embeddings to generate translation candidates.

We also design an evaluation technique to measure the number of unknown words whose correct translation is found within the set of generated candidates. In addition, we propose strategies to integrate these candidates into a SMT system: i) we create a secondary phrase table comprising unknown words and their candidates, and ii) we pre-specify these translation options as markup in the input, for the language model to choose from.

This paper is organized into five sections. Section 2 discusses prior work that addresses

the problem of unknown words, as well as translation of low resource languages and specific domains. We discuss our techniques to generate translations for unknown words as well as their evaluation and integration in Section 3. In Section 4, we present the specific empirical settings used as well as the results and analysis of our experiments. We conclude with a discussion of future work to further address this problem in Section 5.

## 2 Prior Work

Several methods have been proposed to address the unknown words problem. Many of them are based on generation of new translation pairs from monolingual data in the two languages. Irvine and Callison-Burch (2013) induce new translation pairs from monolingual corpora by modeling it as a supervised classification problem which predicts if a given pair of words are translations of each other based on context, timestamps, frequency, topic and orthography features. They show results in both high and low resource settings. In addition to new lexical translations, there has been work in adding new phrases, induced using lexical reordering, idiom extraction and unknown words as part of seen contexts (Zhang and Zong, 2013).

Habash (2008) handles OOVs in Arabic-English translation by augmenting the phrase table with new entries based on morphological analysis, transliteration, spelling correction and dictionary lookup. Habash and Metsky (2008) present another technique for Urdu-English, wherein they match OOVs to their seen morphological variants to find possible translations. But, these methods are heavily dependent on language-specific resources and linguistic properties, and cannot be directly extended to other languages. In contrast, our system is independent of the language pair. In addition to language-specific tasks, Banerjee et al. (2012) classify OOVs for domain-specific technical support forum parallel data into terminology, spelling errors, content words, URLs, email addresses, and fused words. They use a separate technique to handle each type of OOV including regular expression followed by post-editing, using supplementary parallel data and spell checker. This limits its application to technical domain only and, as with language-targeted techniques, is not broadly applicable.

Vector space models are created and applied in various ways to find semantic similarities. Daumé III and Jagarlamudi (2011) use Canonical Correlation Analysis (CCA) on German-English data to mine translations of OOVs in the new domain. They use contextual and orthographic feature vectors. Faruqui and Dyer (2014) show that bilingually correlated word vectors obtained using CCA perform better than monolingual vectors on word similarity tasks. They use Latent Semantic Analysis on word co-occurrence matrices as well as Skip-gram and RNN based methods from Mikolov et al. (2013a,c) to create these vectors. However, they do not apply this to machine translation.

In this work, we design techniques that can generate translation options for OOV words, irrespective of the source language. We do not use language-specific tools and leverage monolingual data in lieu of additional parallel data, making it suitable for low resource languages. Also, we propose ways to integrate this with an SMT system as well as an evaluation technique to measure the quality of candidate translations generated.

## 3 Methodologies

This section surveys common types of OOVs and suggests strategies that target different types.

### 3.1 Types of Unknown Words

We categorize OOV words based on their properties into the following types:

- **Named Entities:** These refer to names of people, organizations, places, etc., that often remain the same across languages. If the writing system for the two languages differ, they

require transliteration.

- **Acronyms:** These are often transliterations and can be considered a subset of named entities. We create a separate category for these due to their distinctive capitalization and punctuation.

- **Borrowed Words:** These are words of the target language that are borrowed in to the source language. They are distinct from names because they are common words that appear in colloquial use of the language, despite having a designated word in the source language. For instance, the word *shirt*, which appears as a transliteration: शर्ट in Hindi. These may also be borrowed words due to the lack of an equivalent word in the source language, for instance, *technology* or *mobile*. We assign them a separate category because they cannot be identified through standard named entity recognition techniques.

- **Borrowed Words with Source-side Inflection:** It is occasionally the case that transliterated English words are combined with the source-side inflection, for instance, to make them plural. These words require a combination of transliteration and morphological analysis to be translated correctly.

- **Source Content Words:** These are words of the source language that are neither borrowed nor named entities. They are not necessarily rare words and appear as OOVs because they did not happen to be seen in the training data. Therefore, this category constitutes a larger percentage of OOVs in low resource settings, which lack training data. This category also includes OOV words whose morphological variants were seen in training data.

- **Misspellings and Typos:** This category comprises words that have multiple spellings, or were typed incorrectly. These words often differ by a character or two from their correct spelling or variant with a known translation. We can translate them by using techniques such as edit distance between these OOVs and the words in training data.

- **Numbers:** These are copied across languages with a few exceptions where the text may have numbers in source language's script.

Borrowed words, content words and named entities are common across many languages. New categories specific to the source language in use can be added.

### 3.2 Translation Methods

In this section, we describe the methods used to generate translation options for unknown words. These methods are inspried by the different types of unknown words described in Section 3.1.

**Levenshtein Distance:** This method targets translation of unknown words that have seen morphological or spelling variants in the training data. Levenshtein distance (Levenshtein, 1966) measures the similarity between two strings based on the number of deletions, additions and substitutions required to transform the first string, $w_1$ into the other, $w_2$.

We obtain an aligned bilingual lexicon created on the parallel training data using the Moses SMT system (Koehn et al., 2007). For each Hindi word, we choose the English word to which it has been aligned with the maximum probability in the lexicon. Next, we compute Levenshtein distance between the OOV word and every source side word in the parallel corpus. If a close match is found between an OOV and a source side word, its aligned English word is listed as that OOV's candidate translation. We design a few variants of this method:

- **Full words:** This is the straightforward application as explained above. Here, the distance is measured between the full OOV and source side words. The candidates produced by this method are a superset of the other methods.

- **Suffix:** This is implemented to target morphological variants in particular and is suited for languages with suffix-based morphology. It only measures the distance between suffixes of the two words. The length of suffix that is compared can be varied based on properties of the language.
- **Prefix:** This is analogous to the suffix-based method and is added for languages that have a prefix-based morphology system.
- **Vowels only:** This method requires that the consonants in the two words be the same and only the vowels may differ. It is specifically added to target our use case for Hindi, which tends to have spelling variants based on differences in vowels, especially when writing borrowed English words. This method is dependent on obtaining the language-specific vowel set from the user (for English, this would be {*a, e, i, o, u, y*}). In addition to vowel-based differences, this method can also be used to capture language-specific common spelling errors by adding those letters instead of the vowel set.

**Word Embedding:** We create word embeddings from a combination of parallel and monolingual data using the Continuous Bag of Words model in *word2vec* (Mikolov et al., 2013a,b,c). This finds representations for words in a continuous space such that words similar in meaning are closer in this space than others. This helps capture semantic similarities. After obtaining word embeddings for both languages, we use Canonical Correlation Analysis (CCA).

CCA is a technique used to learn common features between two sets of data or multiple views of a dataset (Hotelling, 1936). That is, it aims to find the common subspace that maximizes the correlation between them. This can be mathematically written as: given two data matrices, $x \in \mathbb{R}^{d_x \times N}$ and $y \in \mathbb{R}^{d_y \times N}$, it finds vectors $u \in \mathbb{R}^{d_x \times 1}$ and $v \in \mathbb{R}^{d_y \times 1}$ such that:

$$\max \frac{\text{covar}(u^\intercal x, v^\intercal y)}{\sqrt{\text{var}(u^\intercal x)}\sqrt{\text{var}(v^\intercal y)}}$$

$$\text{which can be written as: } \max \frac{\mathbb{E}_{x,y}[u^\intercal x, v^\intercal y]}{\sqrt{\mathbb{E}_x[u^\intercal x]}\sqrt{\mathbb{E}_y[v^\intercal y]}}$$

Since the above expression is affine invariant, it can be written as the following constrained optimization problem of finding a k-dimensional subspace such that $U \in \mathbb{R}^{d_x \times k}$ and $V \in \mathbb{R}^{d_y \times k}$:

$$\max \mathbb{E}_{x,y}[\text{trace}(U^\intercal x y^\intercal V)]$$
$$\text{subject to: } \mathbb{E}_x[\text{trace}(U^\intercal x x^\intercal U)] = I_k;\ \mathbb{E}_y[\text{trace}(V^\intercal y y^\intercal V)] = I_k$$

Solving the above gives final projection vectors as $U = \text{top } k \text{ eigenvectors of } C_{xx}^{-1} C_{xy} C_{yy}^{-1} C_{yx}$ and $V = C_{yy}^{-1} C_{yx} U$. Thus, we obtain a matrix that can be used to project data from the two views into a common maximally correlated space.

To apply CCA, we first find words aligned to each other from the training data using GIZA++ (Och and Ney, 2003) and use it to obtain projection vectors for the source and target languages (Faruqui and Dyer, 2014). We use these vectors to project OOV words and a large English corpora into the shared space and compute cosine similarity between an OOV and each English word to find the closest translation candidates. In addition, we also find the cosine similarity between the OOV and the source language words from the parallel corpus, using the aligned English word as the candidate. We pick the top 10 words with the highest cosine similarity using both these techniques, generating 20 candidate translations in total.

This method is beneficial, especially for low resource settings, since it leverages large monolingual corpora, which are more readily available, particularly in the target language. Also, it

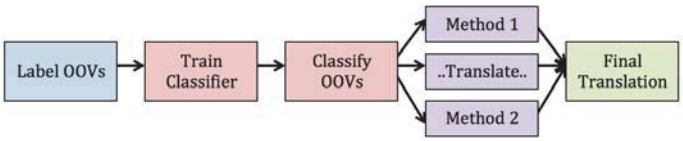


Figure 1: Select method using classifier

has a broad span and helps capture new words without any limitation on the category of OOV (morphological variant, borrowed word etc).

**Transliteration:** This method is used to translate named entities and words that are borrowed from English. We use the unsupervised transliteration module (Durrani et al., 2014b) in the Moses toolkit. First, a non-transliteration model is learned for source-target aligned words that are not transliterations of each other. Second, a transliteration model is learned for word pairs that are transliterations and can be used for learning alignments at the character level. These character alignments are learned using expectation maximization algorithm and is completely unsupervised. Using these learned alignments, unknown source words are transliterated. To improve the unsupervised transliterations learned by the model, we train a larger character-based language model on the target side. This helps the module to produce more accurate spellings.

### 3.3 Oracle Performance

Unknown word resolution is divided in two tasks: (i) the generation of translation candidate and (ii) the selection of the correct one. To assess the performance of the first step, we apply each translation method to generate translation candidates for each OOV. We then search for these candidate translations anywhere in the corresponding English reference sentence. We do not use word-level alignments to check for a particular word in the reference sentence due to noisy alignments. This may lead to some false positives, but a brief manual analysis shows that majority of matches are, indeed, correct translations. To avoid further over-counting, we remove all stop words among the candidate translations. This method helps evaluate the quality of candidates produced by each method by providing an upper bound on the number of OOV words that have a correct translation option and can be correctly translated by the SMT system, when these methods are integrated.

In addition to the candidates generated by each method, we add their synonyms obtained from WordNet (Miller, 1995) using NLTK (Bird, 2006). We include synonyms because the candidates obtained through word embeddings tend to produce semantically similar words which may or may not exactly match the ones in the reference sentence. Similarly, Levenshtein distance produces translations from data seen in training, but it is possible that the correct translation may be its synonym. It also helps cover translations that only differ in grammatical number, truecasing, and other minor variations.

### 3.4 Integration Methods

We consider three ways to integrate our system with an SMT pipeline.

**Classifier:** This method aims to classify OOV words into the categories described in Section 3.1. Using this classification, one can use the translation method appropriate for the category of that OOV word. For instance, if the word is a misspelling, one can use Levenshtein distance to translate it. We hand-tagged OOV words into these categories and used SVM as our supervised classifier. Features included POS of the OOV word and that of its context (window=3). In addition, we added binary features based on if the lemma of OOV is seen in the source side lemmas (suggesting seen morphological variants), if lemma is same as the OOV word (suggesting a named entity) and if the OOV is just comprised of numbers. Length of the word was also used. The entire proposed pipeline for this method is shown in Figure 1.

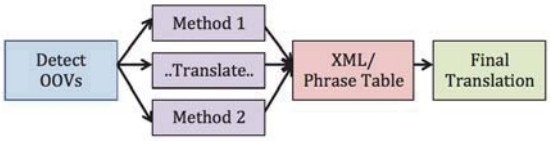


Figure 2: Use all translation methods

**XML Markup:** Here, we use each translation method proposed in Section 3.2 to generate translation options for each OOV. We then add the generated candidates as XML-markup around the OOV in the source test file. Moses has the ability to use such externally-provided translations while decoding (Koehn and Haddow, 2009). Using this method, we use all translation methods proposed in Section 3.2 to generate translation options for each OOV but we do not add any scores to the candidates. The target-side language model chooses the candidate for each OOV.

**Secondary Phrase Table:** To avoid basing the decision of picking the best candidate entirely on the language model, we implement an additional technique. As with XML Markup, we first apply all the translation methods to generate candidates for every OOV. We then create a secondary phrase table comprising OOVs exclusively. For every entry in the phrase table, we use three binary features to indicate the method used to generate the translation candidate. We use one feature to indicate the cosine score for candidates generated using word embeddings. For candidates generated using Levenshtein distance, we use that distance as a feature. In addition to these, we also experimented with using inverse frequency of the candidate word in a large monolingual English corpus as a feature. This is added to balance the preference of language model to always choose the most frequent word as the correct translation. Another variant of secondary phrase table that we implemented discards multi-word candidates. This method is included to prune the candidate list because single word OOVs are more likely to have single word translations.

## 4 Experiments

In this section, we discuss the data and experimental settings we use for implementing and evaluating our translation methods and the obtained results.

### 4.1 Data

We use Hindi as the language for our experiments because of its relatively low resource nature, rich morphology and knowledge of the language. We use Hindi-English news data from the Workshop on Statistical Machine Translation 2014 (Bojar et al., 2014).[1] Details of the source side of this parallel data are listed in Table 1. There are about 2500 sentences in the test set and about as many OOV tokens. Thus, there is a considerable percentage of unknown words.

We also run experiments on Uzbek-English. This data is made available by the Linguistic Data Consortium (LDC2015E89). The training, tuning and test data are sampled from a combination of Uzbek data obtained from news, Wikipedia, social media and discussion forums and translated into English. Additionally, there is Uzbek-English news text that was published in both languages. Details of dataset size are given in Table 1. This setting is lower resource than Hindi. The effects of reduced parallel data can be seen in the analysis. There are about 1000 sentences in the test set and the average number of OOVs per sentence is 4, compared to only 1 for Hindi.

### 4.2 Translation Methodologies

We describe the implementation details of each translation method in this section.

[1] http://statmt.org/wmt14/translation-task.html

| Data | Hindi | | | Uzbek | | |
|---|---|---|---|---|---|---|
| | **Types** | **Tokens** | **Sentences** | **Types** | **Tokens** | **Sentences** |
| Train | 117k | 3.5m | 274k | 49.3k | 161.4k | 55k |
| Tune | 2.5k | 9.2k | 520 | 7.2k | 15.2k | 1k |
| Test | 8.7k | 49k | 2.5k | 7k | 15k | 1k |
| OOVs(Test Set) | 1.9k | 2.9k | - | 3.7k | 4.8k | - |

Table 1: Details of source-side in the parallel data

| Method | OOV Types w/ atleast 1 Candidate Generated | | OOV Types w/ Correct Candidates Detected | |
|---|---|---|---|---|
| Vowel-based | 1025 | 51.9% | 96 | 4.9% |
| Suffix-based | 1020 | 51.6% | 211 | 10.7% |
| Word-based with distance<=1 | 1210 | 61.3% | 289 | 14.6% |
| Word-based with distance<=2 | 1651 | 83.6% | 475 | 24.1% |

Table 2: Comparative performance of variants of Levenshtein Distance (Hindi)

**Levenshtein Distance:** We consider distances of less than or equal to 2 for matching words since Hindi does not have compounding property or unusually long words. For words smaller than length 4, we consider only a distance of 1 to avoid excessive incorrect matches. The same settings are used for Uzbek.

The variants of Levenshtein distance based on suffixes and vowels are beneficial when the goal is to focus on morphological variants or spelling variants respectively. We see good performance by using Levenshtein distance on full words Hindi, this is likely because Hindi is both morphologically rich and prone to spelling variants due to the high percentage of borrowed English words. This is further confirmed by Table 2. We find that using a distance limit of 2 increases the search space but also leads to a considerable improvement in performance, making it a worthy trade-off.

**Word Embedding:** For this technique, we collect Hindi monolingual data from Wikipedia dump (Al-Rfou et al., 2013) and Commoncrawl (Buck et al., 2014),[2] with a total of about 29 million tokens. For Uzbek, the monolingual data is sampled from the same data from which the parallel training data was sampled. For English, we used the Wikipedia data with about 127 million tokens.[3] To the source side monolingual dataset, we add the source side of train, tune and test sets from the parallel data, since embeddings must be generated for OOVs. For English, we only add the target side of the parallel data to the monolingual corpus and not the tune and test reference sentences.

We use *gensim*, Python library's *word2vec* module (Řehůřek and Sojka, 2010) to create word embeddings for each monolingual corpus. We use the Continuous Bag of Words model and vectors of length 100. For both Hindi and Uzbek, we use a low *min_count* setting of 2, such that it only filters words with frequency less than 2. We set this count low in order to obtain embeddings for as many OOVs as possible. Due to the large size of the English corpus, we set this count to 10 to obtain good embeddings and maintain accuracy of the candidate translations. Table 3 shows the number of OOVs for which an embedding is obtained for different *min_count* values in Hindi. The third column shows the oracle performance i.e. the maximum number of OOVs for which the correct candidate was generated using word embeddings.[4] As expected, increasing the minimum frequency filters more OOVs, causing the number of OOVs with embeddings to decrease. But, the number of OOVs with correct candidates do not decrease sharply. This shows that the OOVs seen sufficiently high number of times in monolingual data tend to be more

[2] `http://statmt.org/ngrams/`

[3] `https://code.google.com/archive/p/word2vec/`;`http://mattmahoney.net/dc/textdata.html`

[4] Calculated using the method in Section 3.3

| Min Count | OOV Types w/ atleast 1 Candidate Generated | OOV Types w/ Correct Candidates Detected |
|---|---|---|
| 2 | 1104 | 169 |
| 4 | 945 | 144 |
| 6 | 828 | 151 |
| 8 | 743 | 125 |
| 10 | 681 | 125 |

Table 3: Effect of varying minimum frequency attribute in *word2vec* training for Hindi

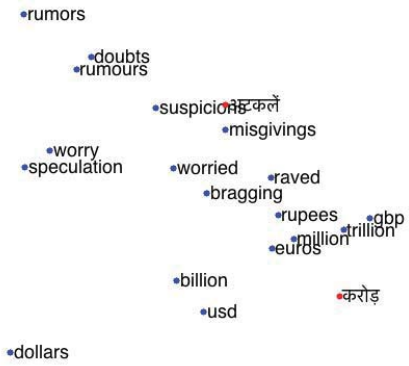



Figure 3: 'अटकलें' (*to speculate*) and 'करोड़' (*crore/ten million*); both have neighboring English words with similar meanings closer to them.

accurately translated due to better vector representations.

Figure 3 shows the top ten words obtained for two Hindi OOVs through cosine similarity between projected OOVs and projected English monolingual corpora.[5] Note that करोड़ which means *crore* (ten million in Indian numbering system) is close to related words like *million, billion, rupees* and so on and farther from *speculation, worried* etc.

**Transliteration:** As described in Section 3.2, we use Moses to transliterate all the OOV words (Durrani et al., 2014b). We use the monolingual English news corpora from past WMT shared tasks (2007-2012) [6] to create the larger character-based language model for more accurate spellings. This data has 1.5 billion words as opposed to 2.9 million words in the target side of the parallel corpus. Since Uzbek uses a Latin script, we copy the OOV for transliteration.

### 4.3 Evaluation

We present results of the oracle evaluation as discussed in Section 3.3.

**Oracle Performance:** The methods differ in how many candidates they produce for each word. We limit transliteration to 1 candidate and the word embeddings method to produce 20 candidates. Levenshtein distance method produces 18 candidates on average. Note that if an OOV word does not have related words (morphological and spelling variants) seen in the source side of training data, the Levenshtein distance method may fail to find the correct candidates or any candidates at all. Similarly, *word2vec* is set up such that every word must appear at least twice in the monolingual data for it to have a vector representation. This may lead to some OOVs being filtered. Transliteration produces a translation candidate for each OOV.[7]

We report the number of OOV types that obtain at least one candidate translation from our

[5] We plot the first vs fifth dimensions here for visual clarity. The candidates shown are the actual top 10 candidates as obtained through a cosine distance across all dimensions

[6] `http://www.statmt.org/wmt12/`

[7] The number of OOVs transliterated are lower in Table 4 because OOVs lost during decoding miss post-decoding

| Method | OOV Types w/ atleast 1 Candidate Generated | | OOV Types w/ Correct Candidates Detected | | OOV Tokens w/ Correct Candidates Detected | |
|---|---|---|---|---|---|---|
| Total OOVs Present | 1975 | - | 1975 | - | 2970 | - |
| All 3 Methods Combined | 1974 | 99.9% | 989 | 50.1% | 1564 | 52.6% |
| Levenshtein Distance | 1651 | 83.6% | 475 | 24.1% | 711 | 23.9% |
| Word Embedding | 1104 | 55.9% | 169 | 8.6% | 236 | 7.9% |
| Transliteration | 1886 | 95.5% | 635 | 32.2% | 1060 | 35.7% |

Table 4: Candidate translations matched in reference sentences (Hindi)

| Method | OOV Types w/ atleast 1 Candidate Generated | | OOV Types w/ Correct Candidates Detected | | OOV Tokens w/ Correct Candidates Detected | |
|---|---|---|---|---|---|---|
| Total OOVs Present | 3719 | - | 3719 | - | 4846 | - |
| All 3 Methods Combined | 3719 | 100% | 1139 | 30.6% | 1453 | 30.0% |
| Levenshtein Distance | 2841 | 76.4% | 696 | 18.7% | 855 | 17.6% |
| Word Embedding | 1596 | 42.9% | 215 | 5.8% | 263 | 5.4% |
| Transliteration | 3719 | 100% | 395 | 10.6% | 551 | 11.4% |

Table 5: Candidate translations matched in reference sentences (Uzbek)

methods in the second column of Table 4. The middle two columns show that on searching in the reference sentence, we match candidates for 50% of OOV types in Hindi. The last two columns present these statistics in terms of OOV tokens. Method-specific OOV coverage is presented in the last three rows while the second row shows the overall coverage computed through a union of these methods.

Let us now turn to Uzbek. Table 5 lists the number of OOV types correctly matched in the reference sentence by running our pipeline on Uzbek data. As noted in Section 4.1, the Uzbek training data is much smaller than that of Hindi and has many more OOVs. Correspondingly, the upper bound achieved in Uzbek is lower.

### 4.4 Results

Here, we present the results obtained by integrating the translation candidates using the methods discussed in Section 3.4.

**Classifier:** To perform the classification experiments, we hand-tagged $\approx 1200$ Hindi OOV words into the categories discussed in Section 3.1. The distribution is shown in Table 6. We used the Hindi POS Tagger by Reddy and Sharoff (2011) for generating features.

The best accuracy obtained with these features was $35.9\%$.[8] This was too low to be used to make decisions about which translation method to use. In addition, several other drawbacks were identified in this approach. Firstly, gold-standard labels are obtained through manual annotation, which requires time, money, and knowledge of the language. Secondly, the distribution of OOV words across these categories is uneven, making it difficult to achieve a high accuracy for smaller classes with limited data. Lastly, for training the classifier, the part of speech tags (POS) of the OOV and its context words are important features. However, low resource languages do not always have such tools readily available.

**XML Markup and Secondary Phrase Table:** We use Moses as the Statistical Machine Translation (SMT) system to run our translation experiments. Settings for the Hindi baseline system have been derived from Edinburgh's submission for WMT-2014 (Durrani et al., 2014a) because the same dataset is being used here. We use basic Moses settings for Uzbek.

transliteration

[8]Note that these are preliminary results and were not explored further due to the drawbacks identified

| Category | Percentage |
|---|---|
| Source Content Words | 22.2% |
| Named Entities | 35.5% |
| Borrowed Words | 28.7% |
| Misspellings & Typos | 7.4% |
| Acronyms | 3.2% |
| Numbers & Punctuation | 1.0% |
| Transliterated English Words with Hindi Inflection | 1.9% |

Table 6: Distribution of OOV categories

| Method | BLEU Score | OOV Types Detected as Correct |
|---|---|---|
| Baseline | 12.15 | 57[10] |
| Transliteration | 12.49 | 593 |
| Transliteration + Bigger LM | 12.67 | 616[11] |
| XML Markup | 12.61 | 412 |
| Secondary Phrase Table | 12.16 | 372 |
| Secondary Phrase Table without multi-word candidates | 12.30 | 376 |
| Secondary Phrase Table with frequency as feature | 12.02 | 281 |
| Oracle | 13.48 | 989 |

Table 7: Results of integration of OOV candidates with SMT pipeline (Hindi)

Table 7 shows results of running Moses with the various integration options. We use BLEU as the evaluation metric (Papineni et al., 2002). We run experiments with the baseline system with no OOV translation, i.e. all OOVs in Hindi script are copied as is in the translated English output. We also run only-transliteration experiments with both the original and bigger language model. Since transliteration is integrated in Moses and we generate only one-best transliteration, these experiments do not require any system to prune or pick a translation. Using this method alone to translate all OOVs is good for languages that are related or have many named entities. We also present results of using XML Markup and secondary phrase table for integration of new translation pairs. In addition, results for the two proposed variations of secondary phrase table are included. First, we add inverse of frequency of the English candidate as an additional feature to the phrase table. Frequency for English words is computed using a large English corpus of past WMT news data.[9] Second, we discard all multi-word synonyms obtained from WordNet. This is because we address single word OOVs here and they are more likely to translate into single word candidates.

In addition to BLEU score, we also record the number of correct OOV translations in the final Moses output. Once we know the correct translation for an OOV by searching for the candidates in the reference sentence, we check if this correct translation appears anywhere in the translated output sentence. In other words, out of the OOVs which obtain a correct translation among their translation options, how many of them had the correct one picked through the SMT system. We use our technique of searching for the translation *anywhere* in the sentence. The possibility of false positives noted in the method of measuring oracle performance also exists in this analysis. But, as mentioned before, there are only a few of those.

Using only transliteration for Hindi-English gives the best performance in terms of both BLEU scores and number of OOVs translated correctly. One of the reasons for this is the high

[9] `http://www.statmt.org/wmt12/`

[10] These are either numerical or other OOVs, like words in English script, that are correctly translated when copied

[11] This number differs from the higher oracle performance of transliteration (635) in Table 4 because synonyms are also included in our system's transliteration candidates

| Method | BLEU Score | OOV Types Detected as Correct |
|---|---|---|
| Baseline | 9.93 | 394[10] |
| XML Markup | 8.91 | 455 |
| Secondary Phrase Table | 9.77 | 507 |
| Oracle | 10.18 | 1139 |

Table 8: Results of integration of OOV candidates with SMT pipeline (Uzbek)

percentage of English words that tend to be used in Hindi language, as shows in Table 6. Furthermore, being news data, there are many named entities. According to our small hand-tagged set in Table 6, these two classes together constitute about 65% of the OOVs. The current integration methods lack sophisticated feature sets that can pick the correct candidate for an OOV from among its 40 (on an average) generated candidates and their synonyms. Discarding multi-word candidates as a basic pruning technique only shows slight improvements. The decrease in performance on adding inverse frequency of candidates as a feature is likely to be due to strong bias for low frequency words. Not all OOVs are rare words, especially in low resource settings, and strongly favoring very unfamiliar translation candidates can lead to a worse performance, unless combined with other useful features.

We also present results of integration of OOV candidates in the SMT pipeline on Uzbek-English data in Table 8. Since Uzbek uses Roman script, simply copying the OOVs in to the output helps translate some OOVs, which is why the baseline and transliteration method are the same.

These results show that we obtain comparable BLEU scores across all the techniques. On re-running identical experiment setups, it was observed that the BLEU scores varies by $\pm 0.35$. Furthermore, an increase in number of OOVs correctly translated did not *always* result in a corresponding higher BLEU score. This suggests that in addition to the augmented OOV translations, other factors also have a considerable influence on the BLEU score, making it less reliable for this task. Also, while we obtain good quality candidates for OOV as seen in the oracle performance, the pipeline for integration requires additional features and pruning techniques to be able to pick the one correct candidate.

## 5 Conclusion and Future Work

We present a system that applies a combination of language-independent techniques to translate OOV words in the absence of large amounts of parallel data, a significant problem for low resource languages. In addition, we also present ways to evaluate these techniques and integrate the generated candidates into an SMT pipeline.

Here, we discuss the current limitations and possible future improvements for each translation method proposed:

- **Word Embeddings:** As we make more progress in word embeddings, we can understand the dimensions and limit it from producing antonyms and distant words. In addition, there are known methods like Deep CCA, which find *non-linear* projections for vectors in the two views (Andrew et al., 2013; Lu et al., 2015). We do not include that in the scope of this work because they require extensive tuning of parameters. For languages with related rich-resource languages, Generalized CCA has the ability to leverage more than two views of data to find a shared subspace with richer and more informed embeddings as has been shown in previous work (Rastogi et al., 2015). Furthermore, to make the cosine distance-based search faster, one can use approximation techniques (Zhao et al., 2015) like Locality Sensitive Hashing (Gionis et al., 1999) and Redundant Bit Vectors (Goldstein et al., 2005), which could not be included in the scope of current work.

- **Levenshtein Distance:** Although Levenshtein distance-based candidates show exact match for a good percentage of OOV words, they generate a lot of candidates. With no score other than the distance from OOV word, there is no way to rank these candidates. In a more resource-rich setting, features such as part of speech tag can be used to prune the list by removing words that do not have a matching tag. When working with a specific language which has, for instance, seen morphological variants for a majority of its OOVs, one can limit to a suffix or prefix based Levenshtein distance to keep the candidate list short.

- **Transliteration:** In this work, we only produce the 1-best transliteration using Moses. But, with better ways of ranking translation options, one can generate an n-best list of transliterations using the same module to widen the scope of coverage for named entities and rectify spelling errors.

To improve the end-to-end performance of this system, better ways of scoring and picking from among the candidate translations are required in addition to the above translation methods. For instance, for domains like news data with a high percentage of named entities, one can use named entity recognition to classify OOVs. The limitations imposed by working in a language-independent and low resource setting make implementing such methods less straightforward. Rich features cannot be created due to low amount of data as well as limited language-specific tools. A seperate phrase table could be used for each method, allowing for method-specific weights and manual tuning.

In conclusion, we present a completely unsupervised pipeline that applies a combination of techniques to translate OOVs and integrates them with an SMT system. We also propose ways to evaluate and measure upper bounds on the performance of these techniques. While there is a scope of improvement in how these candidates are integrated and picked by the SMT system, we obtain a good potential improvement in finding translations for unknown words in low-resource settings.

## Acknowledgments

This material is based upon work supported in part by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR0011-15-C-0113. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the Defense Advanced Research Projects Agency (DARPA). We thank Rebecca Knowles for useful discussions and suggestions. We thank the reviewers for their comments and suggestions.

## References

Al-Rfou, R., Perozzi, B., and Skiena, S. (2013). Polyglot: Distributed word representations for multilingual NLP. *CoNLL-2013*, page 183.

Andrew, G., Arora, R., Bilmes, J., and Livescu, K. (2013). Deep canonical correlation analysis. In *Proceedings of the 30th International Conference on Machine Learning*, pages 1247–1255.

Banerjee, P., Naskar, S., Roturier, J., Way, A., and van Genabith, J. (2012). Domain adaptation in smt of user-generated forum content guided by oov word reduction: Normalization and/or supplementary data. In *Proceedings of the 16th Annual Meeting of the European Association for Machine Translation, Trento, Italy*, pages 169–176.

Bird, S. (2006). NLTK: the natural language toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, pages 69–72. Association for Computational Linguistics.

Bojar, O., Diatka, V., Rychlý, P., Straňák, P., Suchomel, V., Tamchyna, A., and Zeman, D. (2014). HindEnCorp - Hindi-English and Hindi-only Corpus for Machine Translation. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland. European Language Resources Association (ELRA).

Buck, C., Heafield, K., and van Ooyen, B. (2014). N-gram counts and language models from the common crawl. In *Proceedings of the Language Resources and Evaluation Conference*, Reykjavk, Icelandik, Iceland.

Daumé III, H. and Jagarlamudi, J. (2011). Domain adaptation for machine translation by mining unseen words. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 407–412. Association for Computational Linguistics.

Durrani, N., Haddow, B., Koehn, P., and Heafield, K. (2014a). Edinburgh's phrase-based machine translation systems for WMT-14. In *Proceedings of the ACL 2014 Ninth Workshop on Statistical Machine Translation, Baltimore, MD, USA*, pages 97–104.

Durrani, N., Sajjad, H., Hoang, H., and Koehn, P. (2014b). Integrating an unsupervised transliteration model into statistical machine translation. In *EACL*, pages 148–153.

Faruqui, M. and Dyer, C. (2014). Improving vector space word representations using multilingual correlation. In *Proceedings of EACL*.

Gionis, A., Indyk, P., Motwani, R., et al. (1999). Similarity search in high dimensions via hashing. In *VLDB*, volume 99, pages 518–529.

Goldstein, J., Plat, J. C., and Burges, C. J. (2005). Redundant bit vectors for quickly searching high-dimensional regions. In *Deterministic and Statistical Methods in Machine Learning*, pages 137–158. Springer.

Habash, N. (2008). Four techniques for online handling of out-of-vocabulary words in Arabic-English statistical machine translation. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, pages 57–60. Association for Computational Linguistics.

Habash, N. and Metsky, H. (2008). Automatic learning of morphological variations for handling out-of-vocabulary terms in Urdu-English machine translation. *Proceedings of the Association for Machine Translation in the Americas (AMTA-08), Waikiki, HI*.

Hotelling, H. (1936). Relations between two sets of variates. *Biometrika*, 28(3/4):321–377.

Irvine, A. and Callison-Burch, C. (2013). Supervised bilingual lexicon induction with multiple monolingual signals. In *HLT-NAACL*, pages 518–523.

Koehn, P. and Haddow, B. (2009). Edinburgh's submission to all tracks of the WMT2009 shared task with reordering and speed improvements to moses. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 160–164. Association for Computational Linguistics.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., and Herbst, E. (2007). Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, pages 177–180. Association for Computational Linguistics.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710.

Lu, A., Wang, W., Bansal, M., Gimpel, K., and Livescu, K. (2015). Deep multilingual correlation for improved word embeddings. In *Proceedings of NAACL.*

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Mikolov, T., Yih, W., and Zweig, G. (2013c). Linguistic regularities in continuous space word representations. In *HLT-NAACL*, pages 746–751.

Miller, G. A. (1995). WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41.

Och, F. J. and Ney, H. (2003). A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Rastogi, P., Van Durme, B., and Arora, R. (2015). Multiview LSA: Representation learning via generalized CCA. In *Proceedings of NAACL.*

Reddy, S. and Sharoff, S. (2011). Cross language POS taggers (and other tools) for indian languages: An experiment with Kannada using Telugu resources. In *Proceedings of the Fifth International Workshop On Cross Lingual Information Access*, pages 11–19, Chiang Mai, Thailand. Asian Federation of Natural Language Processing.

Řehůřek, R. and Sojka, P. (2010). Software framework for topic modelling with large corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. `http://is.muni.cz/publication/884893/en`.

Zhang, J. and Zong, C. (2013). Learning a phrase-based translation model from monolingual data with application to domain adaptation. In *ACL (1)*, pages 1425–1434.

Zhao, K., Hassan, H., and Auli, M. (2015). Learning translation models from monolingual continuous representations. In *Proc. NAACL.*

# Automatic Construction of Morphologically Motivated Translation Models for Highly Inflected, Low-Resource Languages

**John Hewitt** johnhew@seas.upenn.edu
Department of Computer and Information Science, University of Pennsylvania
Philadelphia, PA 19104

**Matt Post** post@cs.jhu.edu
**David Yarowsky** yarowsky@jhu.edu
Center for Language and Speech Processing, Johns Hopkins University
Baltimore, MD 21211

**Abstract**

Statistical Machine Translation (SMT) of highly inflected, low-resource languages suffers from the problem of low bitext availability, which is exacerbated by large inflectional paradigms. When translating into English, rich source inflections have a high chance of being poorly estimated or out-of-vocabulary (OOV). We present a source language-agnostic system for automatically constructing phrase pairs from foreign-language inflections and their morphological analyses using manually constructed datasets, including Wiktionary. We then demonstrate the utility of these phrase tables in improving translation into English from Finnish, Czech, and Turkish in simulated low-resource settings, finding substantial gains in translation quality. We report up to +2.58 BLEU in a simulated low-resource setting and +1.65 BLEU in a moderate-resource setting. We release our morphologically-motivated translation models, with tens of thousands of inflections in each of 8 languages.

## 1 Introduction

Statistical machine translation systems are typically trained on large bilingual parallel corpora (bitext). Low-resource machine translation focuses on translation of languages for which there exists little bitext, and where translation quality is subsequently often poor. Highly inflected languages—those that exhibit large inflectional paradigms of words with a common dictionary entry—excacerbate the problems of a low-resource setting. Many inflections of words in an inflectional paradigm are complex and rare, and their translations are unlikely to be well-estimated even in a moderately large parallel corpus. For example, Koehn (2005) point to the highly inflected nature of Finnish as a reason for poor translation performance into English even in high-resource settings.

However, even where bitext may be lacking or scarce, there are often many other resources available. One source of rich morphological information is Wiktionary.[1] This paper describes a method for using resources extracted from Wiktionary to automatically map inflections in paradigms of morphologically rich languages to ranked sets of English phrasal translations. This is done by the following procedure:

[1] https://www.wiktionary.org/

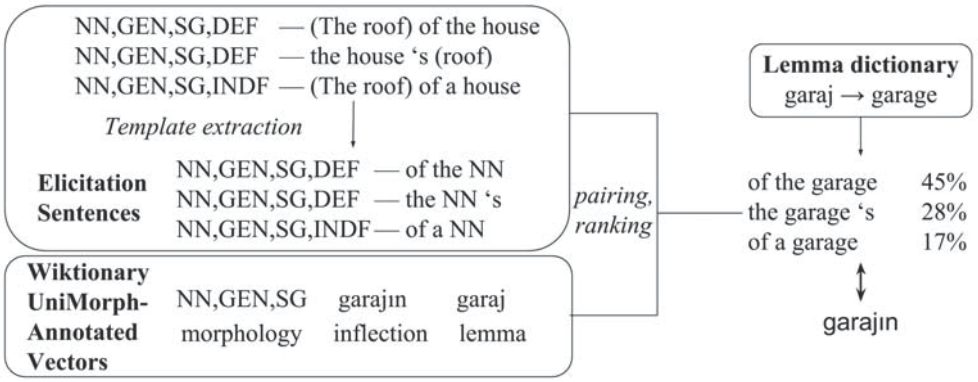



Figure 1: The translation model (TM) construction pipeline. This depicts the process by which we map each morphologically annotated inflection to a ranked set of English phrasal translations.

1. We begin with resources from the UniMorph project (Sylak-Glassman et al., 2015), which produced millions of tuples pairing inflected words forms with their lemmas and a rich morphological tag (which we refer to as a UniMorph tag or vector) that was designed to be a universal representation of morphological features (§3).

2. We next take a small set of pairs of UniMorph vectors and short English sentences that were produced in an *Elicitation Corpus*, designed to collect inflections that in English are expressed phrasally instead of morphologically (§4).

3. We then produce phrasal translation pairs by extracting English phrases from these sentences and pairing them with the foreign language through the UniMorph tag (§5). We investigate different methods for extracting and scoring phrase pairs.

4. Finally, we evaluate the utility of these phrase pairs to improve machine translation in simulated low-resource settings (§6).

A depiction of the full pipeline is in Figure 1.

## 2 Prior Work

Maximizing the utility of a baseline phrase table has been the focus of a large body of prior work in translating from morphologically rich languages. Habash (2008) work on the OOV problem in Arabic, mapping OOV types to in-vocabulary (INV) types by orthographic and morphological smoothing methods. Mirkin et al. (2009) take inspiration from the Textual Entailment (TE) problem, using WordNet to determine a set of entailed alternatives for English OOV tokens. However, since this OOV-resolution scheme is dependent on the existence of a semantic resource like WordNet in the source language, it is unsuitable in general low-resource settings. Yang and Kirchhoff (2006) implement a backoff model for Finnish and German, stemming and splitting OOV tokens at test time and searching a baseline phrase table for the resulting simplified forms.

Many systems attempt to address the incorrect independence assumptions traditional phrase-based MT systems impose on inflections in the same paradigm. Koehn and Haddow (2012) train a baseline phrase-based translation model, and back off to a factored model that

| Inflection | Lemma | Mood | POS | Tense | Gender | Number | Animacy | Person |
|---|---|---|---|---|---|---|---|---|
| bude absolvovat | absolvovat | IND | VB | FUT | | 2 | | SG |
| absolvuj | absolvovat | IMP | VB | | | 2 | | SG |
| absolvujete | absolvovat | IND | VB | PST | MASC | 2 | ANIM | SG |
| absolvoval jste | absolvovat | IND | VB | PST | MASC | 2 | INAN | SG |

Table 1: Czech verb inflections and partial annotations from Wiktionary. Empty cells indicate that the inflection is not marked in that dimension.

decomposes OOV tokens into `lemma+morphology`. Dyer (2007) address the intuition that even INV tokens may have poorly estimated translations. They decode on a confusion network of tranformations of source sentences, imposing penalties for backing off from surface tokens.

Each of these approaches attempts to use pre- or post-processing to make up for poorly estimated, low-coverage phrase tables. We take advantage of an entirely new resource, a very large, massively multilingual, morphologically-annotated dictionary, to directly improve phrase table coverage and provide improved estimations for INV types. Thus, the practical translation gains we see through our methods should be orthogonal to those of prior work. This paper presents a method of constructing English phrases that express inflectional features, and a novel system of mapping these phrases to foreign inflections, with the following qualities:

- We use no bitext, and no language-dependent morphological information.
- We apply our system to the Wiktionary dataset, constructing substantial phrase tables for 8 languages, with the capacity to build a model for each of the 73 languages with more than 10,000 inflections in Wiktionary.
- We demonstrate the utility of these phrase tables, finding substantial gains when augmenting low-resource MT of Czech, Finnish, and Turkish.
- We present insights on the utility of morphological information in translation by conducting an ablation study in two dimensions, analyzing the effects of varying available bitext and available morphological information for each language.

## 3 UniMorph Inflectional Paradigms

The Universal Morphological Feature Schema (UniMorph) represents fine-grained distinctions in meaning expressed through inflectional morphology cross-lingually (Sylak-Glassman et al., 2015). The schema defines 23 distinct dimensions across which inflections can vary independently. Though English does not express many of these features through morphology, their purposes can still be intuitive:

> of the houses : NN, GEN, PL, DEF (Noun, Genitive, Plural, Definite)
> with a hammer : NN, COM, SG, INDF (Noun, Comitative, Singular, Indefinite)

The dimensions are as follows:

> Aktionsart, Animacy, Aspect, Case, Comparison, Definiteness, Deixis, Evidentiality, Finiteness, Gender+, Information Structure, Interrogativity, Mood, Number, Part of Speech, Person, Polarity, Politeness, Switch-Reference, Tense, Valency, Voice

A total of 212 possible values are distributed across the dimensions. The morphological information of each inflection is encoded in its "UniMorph vector."

| Inflection | Lemma | Case | Number |
|---|---|---|---|
| absurdity | absurdita | GEN | SG |
| absurdit | absurdita | GEN | PL |
| absurdit | absurdita | DAT | SG |
| absurditm | absurdita | DAT | PL |

Table 2: Czech noun inflections and partial annotations from Wiktionary.

Sylak-Glassman et al. (2015) scraped Wiktionary, extracted inflectional paradigms in hundreds of languages, and transformed the inflectional annotations to UniMorph vectors. The editors of Wiktionary often include all inflections for each lemma, so full inflectional paradigms are scraped. The result, as shown for nouns in Table 2 and verbs in Table 1, is a list of inflections with corresponding lemmata, and inflectional values as a UniMorph vector. Each language has a corresponding table of inflection entries with their corresponding UniMorph vector. The size of the Wiktionary dataset varies from language to language, including 2,990,482 Finnish inflections and 15,132 Swahili inflections.

To a large extent, combining a lemma with the information in the UniMorph vector reconstructs the meaning of the inflection. We do not claim perfect reconstruction, as no inflectional morphological schema can perfectly encode language-independent meaning. However, practical gains in translation quality do not require a perfect schema. We instead focus on the substantial signal provided by UniMorph annotations, and show that they are highly effective at providing cross-linguistic information.

## 4 English Inflectional Elicitation

Wiktionary provides us with UniMorph vectors for inflections, but provides no information about how to express these vectors in English. These expressions are difficult to generate, as English expresses inflectional information phrasally. For this, we use a corpus of 8,000 UniMorph-annotated English sentences, which we call the *Elicitation Corpus*. The corpus was developed in an attempt to document the full inflectional variation in 49 languages of morphological interest.

The process worked as follows. For each language, we first collected all its important morphological tags. We then hand-built the Cartesian product of all inflectional variation expressed by the language. Thus, if a language inflected nouns exactly for 4 values of number and 3 values of case, we constructed 12 vectors of (Number $\times$ Case). Then, for each UniMorph tag, keeping the specific language and part of speech in mind, we manually wrote an English sentence with a head word on which the same inflectional features are expressed. For example, for the tag

VB, 3, PRS, PRF, PASS (Verb, 3rd person, Present, Perfect, Passive)

we might generate the sentence

[The apple] has been eaten.

These sentences were given to a bilingual native speaker. The goal of each sentence was to elicit, in the foreign language, the inflection encoded by the lemma we used and the UniMorph vector expressed in English. We wanted to avoid sentential translations, aiming instead for individual inflections. To this end, portions of each sentence necessary for syntactic coherence but deemed unnecessary to express inflectional values were enclosed in brackets. Each line in the corpus is a tuple of (English sentence, UniMorph vector).

By construction, the corpus spans a large amount of the variance of inflectional expression of most languages. Further, equivalent UniMorph vectors in multiple languages were not

| English Sentence | UniMorph Vector |
|---|---|
| [They] were eating [when the door broke.] | IND, PST, PROG, 3, PL |
| [The view was blocked] by the houses | INS, PL |
| Was [he] not speaking? | PST, PROG, NEG, INT, 3, SG |
| [He is] sleeping | PRS, PROG, POS, 3, SG |

Table 3: Entries from the Elicitation Corpus. Each sentence on the left was constructed to express the UniMorph vector on the right. Note that not all are fully defined, e.g., missing definiteness, potentially due to the original source language not inflecting for that dimension.

de-duplicated, so many common vectors are paired with multiple English sentences. This permits the corpus to store information about the frequency with which varying ways are used to express the same features. This eventually aids us in ranking phrase templates. For example, the genitive in English is expressed equivalently as *the house's roof* and *the roof of the house*. More examples of sentences in the Elicitation Corpus are given in Table 3.

## 5 Constructing Morphological Phrase Tables

In this section, we detail the pipeline by which we automatically construct a translation model or phrase table for each language represented in the Wiktionary dataset.

1. We extract and rank English *phrase templates* from the 8000-sentence Elicitation Corpus.
2. We use UniMorph vectors to pair our phrase templates with inflections from Wiktionary, and estimate direct and inverse translation probabilities.
3. We complete phrase templates with lemmata to finish the translation hypotheses.

### 5.1 Phrase Template Extraction

Each sentence in the Elicitation Corpus expresses a UniMorph vector in English. However, the context and specific head word of the sentence constrain the usefulness of the sentence. Taking our earlier example, we wish to generalize

[The apple] has been eaten.

such that the inflectional values are kept, but the resulting "template" is maximally reusable in different contexts, and for different lemmata. Recalling the UniMorph vector associated with this sentence, we wish to extract

has been VBN : {VB, 3, PRS, PRF, PASS}

The context has been removed, the morphological head word replaced with a part of speech "variable". In effect, the goal of this extraction is to retain exactly the information described by the UniMorph vector. Information like the lemma will be provided by each source language inflection. We use two simple methods to extract phrase templates from the Elicitation Corpus.

#### 5.1.1 Naive Template Extraction

From each sentence in our Elicitation Corpus, we extract a "naive template". After part-of-speech tagging the sentence, all parenthesis-denoted context is removed. Then, we replace the rightmost word whose POS matches the UniMorph vector POS with a variable. For this variable, we choose the most descriptive POS, e.g., VBD instead of VB, to preserve the information necessary to conjugate an English lemma as a replacement for the variable.

Table 4 gives examples in which this naive method produces incorrect or incomplete results. To augment template extraction, we also use a slightly more principled heuristic method.

| English Sentence | Naive Template | Generated Templates |
|---|---|---|
| (They) were eating (when the door broke.) | were VBG | they were VBG, were VBG |
| (The dog went) from the boy. | from the NN | from the NN |
| (The view was blocked) by the leaves. | by the NNS | by the NNS |
| Was (he) not speaking? | was not VBG* | was he not VBG |
| (He is) sleeping. | VBG* | he is VBG, is VBG |

Table 4: Template extractions. * denotes a clearly incorrect result given the UniMorph vector in Table 3.

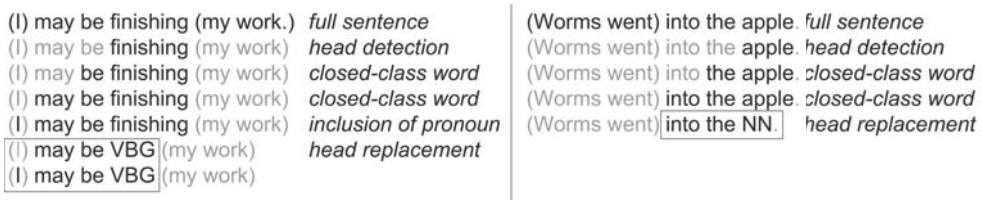



Figure 2: The extraction process, from sentences to templates. Each line shows the next word added to the template. Boxed phrases are final templates. Greyed words have not yet been considered by the algorithm, or have been excluded from the template.

#### 5.1.2 Heuristic Template Extraction

To automatically generate phrase templates, we make the assumption that we're working only with simple sentences, and we assume the presence of context-marking parenthesis. Given these assumptions, we construct an algorithm to extract only the inflectional value-carrying neighbors of the head as part of the phrase table.

1. Determine the head of the sentence by searching for the last word tagged with a POS corresponding to the correct word class. (e.g., VBN and VBP correspond to VB)

2. Walk backwards from the head, prepending every closed-class word to the output template.

3. When an open-class word is seen, stop.

4. Replace the head of the sentence with its part-of-speech tag.

Open-class words such as nouns or verbs are unlikely to encode inflectional values, and are likely to include undesirable specifics for the sentence (such as a verb's subject.) However, there are a few verbs that are necessary for expressing, for example, tense and aspect. As such, we manually compiled a list of these words, and modified our POS tagger to let them pass. Words such as *had, have, going, am, are, did...*, for example, are used to express aspect and mood in English. We used a Brown corpus-trained tagger from the `nltk` python package (Bird et al., 2009).

A few examples of this process are given in Figure 2. The first example demonstrates the multiple potential phrase templates for a single sentence. Because our system is language-independent, we have no information about whether a pronoun in an MT setting will be omitted (as in languages with pro-drop). By extracting phrase templates with and without a pronoun, we expect that the language model will bias towards the with-pronoun phrase in sentences with pro-drop, and towards the without-pronoun phrase in sentences with a marked pronoun.

| Inflection | Phrase Template | Lemma Translation | Phrasal Translation |
|---|---|---|---|
| blafovali | they were *VBG* | blafovat, bluff | they were bluffing |
| filmujme | let 's *VB* | filmovat, film | let 's film |
| kaupunginvaltuuston | of the *NN* | kaup. . . , city council | of the city council |

Table 5: Phrase completion example. Once an inflection has been paired with (a) phrase template(s), we look up its lemma translation, conjugate it, and insert it into the template to complete the phrasal translation.

### 5.2 Matching Phrase Templates to inflections' UniMorph Vectors

A phrase template $t$ with UniMorph vector $t.v$ is proposed as a candidate translation for inflection $i$ with UniMorph vector $i.v$ if $t.v$ is a superset of the features in $i.v$, where UniMorph vectors are considered unordered sets of inflectional values. We use this superset-match method instead of only constructing phrase pairs with exact morphological matches to account for a large amount of underdefined Wiktionary inflections, each with very few values in its UniMorph vector. The superset matching scheme provides these underdefined inflections with a large number of low-probability phrase templates, reflecting the noise due to the lack of morphological information.

The set of all phrase templates $\mathrm{T}$ for vector $i.v$ is

$$\mathrm{T}(i.v) = \{\, t \mid t.v \supseteq i.v \}$$

Each $t \in \mathrm{T}(i.v)$ has a value $\mathrm{freq}(t|\,\mathrm{T}(i.v))$, the count of sentences in the Elicitation Corpus with the phrase template $t$ whose UniMorph vector is a superset of $i.v$. Thus, each vector $i.v$ for which there exists at least 1 template has a total count

$$\mathrm{total}(i.v) = \sum_{t \in \mathrm{T}(i.v)} \mathrm{freq}(t|\,\mathrm{T}(i.v))$$

The direct translation probability, that $t$ is the correct way to express UniMorph vector $i.v$, is thus

$$\mathrm{P}(t|i.v) = \frac{\mathrm{freq}(t|\,\mathrm{T}(i.v))}{\mathrm{total}(i.v)}$$

We also calculate the probability that $v$ is the best morphological analysis of $t$ by calculating the total probability mass of $t$ in all $T$. Thus, $\mathrm{total}_{\mathrm{P}}(t) = \sum_{i \in I} \mathrm{P}(t|\,\mathrm{T}(i.v))$, where $I$ is the set of all inflections. The inverse translation probability is thus

$$\mathrm{P}(i.v|t) = \frac{\mathrm{P}(t|i.v)}{\mathrm{total}_{\mathrm{P}}(t)}$$

Intuitively, the inverse translation probability discounts highly specified templates (with a rich UniMorph vector) in underspecified settings.

### 5.3 Phrase Template Completion

So far, we have described a system for mapping morphological feature vectors to sets of English phrase templates. The Wiktionary dataset provides a map from foreign inflections to their corresponding feature vectors. Composing the two, we map foreign inflections to phrase templates. The final step in the process is to complete, or compile out the phrase templates, replacing the part-of-speech variable with an inflected English word whose corresponding lemma translates to the foreign inflection's lemma. The Wiktionary dataset provides a mapping from foreign

inflection to corresponding foreign lemma. We use a lemma dictionary built from Wiktionary and Google Translate to map between foreign lemma and English lemma. (Note that for all languages in Wiktionary, including those missing from Google Translate, a lemma dictionary is extractable with the inflections used.) Finally, we inflect the English lemma using an English pattern library (De Smedt and Daelemans, 2012). The input and output of phrase template completion are shown in Table 5.

### 5.4 Phrase Table Construction

Running our system on the Wiktionary inflections for Finnish, Czech, Russian, Korean, Georgian, Swahili, Turkish, and Urdu, we construct a phrase table for each language, containing the top-5 phrasal translations for each inflection, as well as their computed direct and inverse translation probabilities. We release all constructed models for use in morphological analysis as well as end-to-end SMT.[2]

## 6 Experimental Design

We evaluate our system by examining its effectiveness in improving the quality of end-to-end MT from Czech, Finnish, Russian and Turkish into English. We use the Joshua decoder (Post et al., 2015) with Hiero grammars (Chiang, 2007). For many of our target languages, the only bitext available is the Bible. We thus simulate a low-resource setting by training on the Bible. We simulate moderately higher-resource settings by appending differing numbers of lines of modern bitext (described below) to the bible. We test on newswire from the Workshop on Statistical Machine Translation (Bojar et al., 2015a).

For all translation models, we use a gigaword-trained 5-gram language model (LM). We anticipated that the English phrases in our tables might be discounted by the language model due to higher-order (3- and 4-gram) misses in the gigaword corpus. In preliminary experiments, we trained language models with `lmplz` (Heafield et al., 2013) on the training data with our phrases appended. However, we saw best performance when using the gigaword LM, and by using it across all TMs, the BLEU scores are kept comparable.

Table 6 presents statistics on how many tokens and types in the test data are found in our Wiktionary inflections, anchoring the potential benefit of the system. We note that for Finnish, Czech, and Turkish, our system covers a large number of both OOV and in-vocabulary (INV) tokens in each of the resource settings. This points to potential translation quality gains through coverage of previously OOV wordforms as well as improved translation of poorly estimated wordforms.

### 6.1 Morphological Information Ablation Study

Along with testing the validity of our particular generation of morphologically-motivated phrasal translations, we present an ablation study, highlighting the effects of using varying portions of the morphological information provided to us, in 4 cases.

1. We consider the use of no morphological information. This completely unmodified Joshua system is our baseline.

2. We test the inclusion of a lemma dictionary with the bitext, including no morphological information. As a small dictionary largely comes for free for even low-resource languages, we see this as a stronger baseline.

3. We test the inclusion of an *inflection dictionary*. This is made possible through the Wiktionary dataset. This augmentation method pairs each inflection with a corresponding bare

[2] `https://github.com/john-hewitt/morph16`

| | Bible | | | Bible +20k | | | Bible+20k+Wiktionary | | | Abs % |
|---|---|---|---|---|---|---|---|---|---|---|
| | Covered | Total | % | Covered | Total | % | Covered | Total | % | Added |
| Finnish Types | 1,999 | 8,497 | 23.5 | 4,558 | 8,497 | 53.6 | 5,896 | 8,497 | 69.3 | **15.75** |
| Finnish Tokens | 7,456 | 16,447 | 45.3 | 11,613 | 16,447 | 70.6 | 13,188 | 16,447 | 80.1 | **9.58** |
| Czech Types | 3,445 | 16,581 | 20.7 | 9,616 | 16,581 | 57.9 | 10,110 | 16,581 | 60.9 | **2.98** |
| Czech Tokens | 26,377 | 51,373 | 51.3 | 41,972 | 51,373 | 81.7 | 42,618 | 51,373 | 82.9 | **1.26** |
| Turkish Types | 1,658 | 3,677 | 45.0 | 2,698 | 3,677 | 73.3 | 2,824 | 3,677 | 76.8 | **3.43** |
| Turkish Tokens | 4,268 | 7,337 | 58.1 | 6,036 | 7,337 | 82.2 | 6,205 | 7,337 | 84.5 | **2.30** |
| Russian Types | 4,112 | 15,346 | 26.8 | 9,949 | 15,346 | 64.8 | 9,950 | 15,346 | 64.8 | 0.01 |
| Russian Tokens | 25,882 | 45,390 | 57.0 | 38,407 | 45,390 | 84.6 | 38,408 | 45,390 | 84.6 | 0.00 |

Table 6: The first column reports the number of tokens and types in the newswire test set that are in-vocabulary in a model trained on the Bible. It also reports the total number of tokens and types in the test set, and the percent of coverage. The second column reports the same statistics for a model trained on the Bible and 20,000 sentences of modern text. The third column reports the same statistics for a model that is trained on the Bible plus 20,000 sentences of modern text, and includes our inflections. The column *Abs % Added* reports how many percentage points of type and token coverage are gained when adding the inflections.

English lemma. This encodes *some* morphological information, as it groups all inflections of a paradigm together by their common lemma.

4. We test our full system, mapping inflections to English phrasal translations motivated by the morphological features of the inflection.

### 6.2 How much does morphology help when I have X much data?

Our methods are particularly exciting as they come largely for free from the Wiktionary corpus, and do not depend on the amount of bitext available for a language. However, it is interesting to examine the utility of our work as we vary the definition of low-resource. We evaluate our work by training models on Bible bitext with varying amounts of bitext and analyzing the benefit of augmenting each model with our system:

1. We train a model only on translated portions of the Bible.

2. We consider a slightly higher-resource setting, appending 1000 sentences of Europarl (Koehn, 2005), SETIMES[3] (Tyers and Alperen, 2010), extracted from OPUS (Tiedemann, 2009), or Common Crawl (Bojar et al., 2015b) (depending on the language, described below) to the Biblical training set.

3. We train a model with 20,000 lines of modern data and the biblical training set.

4. For Finnish and Czech only, we consider the highest of our low-resource settings, appending 50,000 lines of modern data to the biblical training set.

### 6.3 Augmentation Method

The inclusion of outside data as augmentation for an existing translation model is non-trivial, as the probabilities in the outside data are unrelated to those of the table. The well-estimated translations of each must be given substantial probability mass in the resulting combined table without adversely promoting poorly estimated phrase pairs. We test a *dual grammar* method,

[3] http://nlp.ffzg.hr/resources/corpora/setimes/

| Language | # of Inflections |
|---|---|
| Finnish | 2,990,482 |
| Russian | 326,361 |
| Turkish | 275,342 |
| Czech | 145,230 |
| Georgian | 121,625 |
| Korean | 76,739 |
| Swahili | 15,132 |
| Urdu | 13,682 |

Table 7: The number of annotated inflections provided by the Wiktionary dataset for each of the eight languages for which we build phrase tables. Note that these numbers come from the latest release of UniMorph, and may differ modestly from the number of inflections used in our experiments.

wherein our artificial phrases and their translation probabilities are constructed as a Joshua phrase table, and assigned weights tuned by Joshua in tandem with its bitext-derived phrase table. We also test a *bitext augmentation* method, wherein the artificial phrases are appended to the bitext. In particular, we concatenate the bitext to itself 10X, and allocate a quota of 10 lines total for all translation candidates of each inflection in our artificial data. Finally, we test the two methods in combination.

We found no consistent best method out of the three. Thus, we present either the best of the three, or just the dual grammar method, as it simplified manually identifying when artificially constructed phrases were used by the decoder.

### 6.4 Training, Tuning, Testing Sets

For the Finnish-English, Czech-English, Russian-English, and Turkish-English language pairs, we train a Hiero model on 29,000 sentences of Biblical data. Separately, for Finnish and Czech, we train models on 29,000 sentences of Biblical data with 1,000, 20,000, and 50,000 sentences of Europarl. For Russian and Turkish, we train models on 29,000 sentences of biblical data with 1,000 and 20,000 sentences of CommonCrawl and SETIMES data, respectively.

Our tuning and test sets are consistent, per language, through all experiments. We tuned Finnish-English on the Workshop on Machine Translation (WMT) 2015 dev set (1810 sentences), and tested on the WMT 2015 test set (1724 sentences). We tuned Czech-English on the WMT 2013 test set (3000 sentences) and tested on the WMT 2014 test set (3287 sentences). We tuned Turkish-English on half of the WMT 2016 test set (506 sentences) and tested on the other half (505 sentences). We tuned Russian-English on the WMT 2013 test set (3000 sentences) and tested on the WMT 2014 test set (3308 sentences).

## 7 Results and Analysis

Figure 3 illustrates that our system is able to correctly generate the English expression of complex inflectional information, both intelligently filling in OOVs and providing better translations for poorly-estimated inflections. Table 8 shows the results of our end-to-end tests. Our study of end-to-end MT proceeds in two dimensions, varying both the amount of training data and the amount of morphological information. We focus our analysis on Finnish, Czech, and Turkish, for which we see substantial gains in BLEU. We present a negative result for Russian, documented in Table 8. Post-hoc examination of the tokens and types of test corpora in Table 6 explains this: only a few, low-frequency types in the Russian test corpus are covered in the

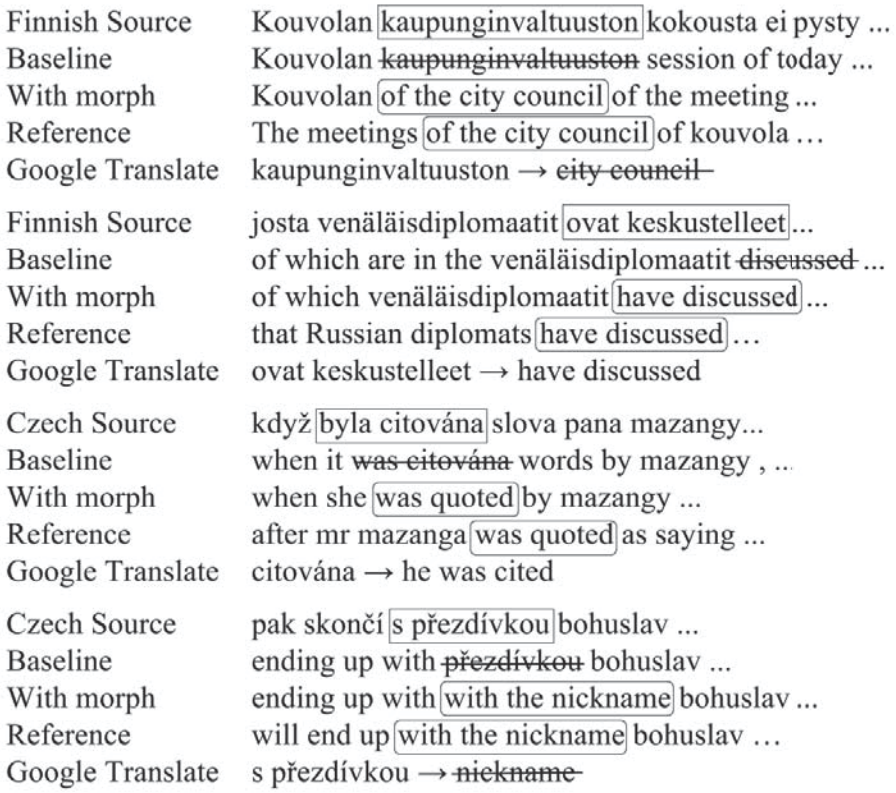



Figure 3: Newswire sentences from the test set where a source inflection is either OOV or poorly estimated in a low-resource setting, and a precise translation is generated by our system. The baseline system is trained on the Bible + 20,000 sentences of Europarl. Our system's translation is denoted by "With morph". Targeted inflections are boxed, and their translations from our system are in rounded boxes along with the reference translations. Also provided, for reference, is Google Translate's translation of the inflection. We note that Google's Finnish and Czech models were not constrained in the amount of training data, but still fail to capture the genitive case of *city council* in Finnish and the instrumental case of *nickname* in Czech. Manual analysis showed that for the Czech *citovna*, the baseline *was* was not generated in the system translation; instead, the Czech inflection was expressed fully as *was quoted*.

| | Unmodified | Lemmata | Inflections | Full Morph |
|---|---|---|---|---|
| Finnish Bible | 4.70 | 5.29† | **5.85 | **7.28 |
| Bible + 1k Europarl | 5.39 | 6.07† | **6.59 | **7.73 |
| Bible + 20k Europarl | 8.58 | 9.07† | **9.51 | **10.37 |
| Bible + 50k Europarl | 9.76 | 10.43 | 10.61† | ****11.41** |
| Czech Bible | 5.29 | 5.69† | **5.91 | **6.17 |
| Bible + 1k Europarl | 6.98 | 7.33† | **7.64 | ** 8.01 |
| Bible + 20k Europarl | 13.98 | 14.04 | 14.22 | *14.33 |
| Bible + 50k Europarl | 16.23 | 16.20 | *16.02 | ****16.69** |
| Turkish Bible | 3.58 | 4.09† | 4.02 | 4.23 |
| Bible +1k SETIMES | 4.78 | 5.00 | 4.96 | *5.28 |
| Bible +20k SETIMES | 7.85 | 8.05 | 8.19 | **8.26** |
| Russian Bible | 1.18 | 1.32† | 1.26 | 1.20 |
| Bible + 1k Common | 6.24 | 6.26 | 6.25 | 6.26 |
| Bible + 20k Common | 11.20 | **11.22** | 11.17 | 11.21 |

Table 8: All experiment results. * and ** denote significantly better results than the lemma-lemma model of on the same bitext, at $p < 0.05$ and $0.01$, respectively. To motivate our use of the lemma-lemma model as a sronger baseline for significance testing than the unaugmented MT system, we denote with a † where the lemma-lemma model is significantly better than the unaugmented system. We use the bootstrap resampling method of Koehn (2004) to estimate significance tests.

Wiktionary dataset. We also note that the potential benefit of our system is constrained by the quality and size of the Wiktionary dataset for a given language. Table 7 gives the size the dataset for each of the eight languages for which we release a phrase table.[4]

### 7.1 Full Morphological Translation Model

Augmenting an SMT system with a translation model built by our full morphological analysis and generation improves translation quality significantly across Finnish, Czech, and Turkish, even at higher levels of resources. We expected that the potential benefit of adding morphological information would decrease as the training set size of the baseline model increased. At increasing sizes of training corpora, the gains from morphology decrease, but remain significant for Finnish and Czech. This may suggest that morphological information aids in the translation of poorly estimated inflections even in settings of moderate resources. It is worth noting that adding even 20k sentences of Europarl to the training data improved over a baseline Bible system more than adding the entirety of our morphological information. However, when adding the morphology *on top of* the added bitext, there are substantial gains.

### 7.2 Inflection-Lemma Model

Augmenting an SMT system with a translation model that naively pairs all Wiktionary inflections with their English lemma equivalents also improves translation quality, often to a substantial percentage of the full model's gains. These gains may be due to OOV coverage, even with poor translations. It is encouraging to see that this model does not perform as well as the full morphological model. This points to the utility of the UniMorph vectors and our phrases in providing the capacity for *good* translation estimates, not just OOV coverage. For Finnish and Czech, the percentage of a full morphological system's gains recovered by an inflection-lemma

[4] http://unimorph.org

model seems to be independent of the resource setting. This model recovered an average of 50% of the gains of the full Finnish system, and 50.2% for Czech.

### 7.3 Lemma-Lemma Model

Augmenting an SMT system with a translation model that pairs entries in a dictionary also improves translation at most resource levels. However, it is an impoverished system that lacks the OOV coverage of the inflection-lemma model. Except for Turkish at the 0k and 1k levels, the lemma-lemma model underperforms the inflection-lemma model, as expected. Regardless, it is a stronger baseline than an unmodified MT system, significantly outperforming the unmodified system in half of the experimental settings.

## 8 Summary and Future Work

Translation of highly inflected languages presents compounding data scarcity problems for SMT systems with little bitext. The information encoded in the inflections of highly inflected languages is formalized in UniMorph, and a large, multilingual, freely available repository of UniMorph-annotated inflections exists in the Wiktionary dataset. Using a small UniMorph-annotated English corpus, we generalize English inflectional phrase templates to express a wide range of UniMorph vectors. We then use the inflectional information to assign English phrase templates as translation candidates to inflections from Wiktionary. Building translation models from these pairs, we substantially improve the quality of MT in a range of low-resource settings.

Analyzing a range of resource settings and levels of morphological information, we find that a full morphological system outperforms inflection-lemma mappings and lemma-lemma mappings. We also find that morphological information is less useful in higher-resource settings, but can still provide substantial gains. We believe this approach holds promise for constructing translation systems for language pairs that do not have much in the way of bitext. In service of this goal, we release translation models constructed for Finnish, Czech, Russian, Korean, Georgian, Swahili, Turkish, and Urdu.

For future work, we believe it would be useful to investigate how well these phrase-table augmentation techniques work when combined with other approaches to low-resource machine translation into English, such as lemmatized backoff models.

## References


Bird, S., Klein, E., and Loper, E. (2009). *Natural language processing with Python*. " O'Reilly Media, Inc.".

Bojar, O., Chatterjee, R., Federmann, C., Haddow, B., Huck, M., Hokamp, C., Koehn, P., Logacheva, V., Monz, C., Negri, M., Post, M., Scarton, C., Specia, L., and Turchi, M. (2015a). Findings of the 2015 workshop on statistical machine translation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 1–46, Lisbon, Portugal. Association for Computational Linguistics.

Bojar, O., Chatterjee, R., Federmann, C., Haddow, B., Huck, M., Hokamp, C., Koehn, P., Logacheva, V., Monz, C., Negri, M., Post, M., Scarton, C., Specia, L., and Turchi, M. (2015b). Findings of the 2015 workshop on statistical machine translation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 1–46, Lisbon, Portugal. Association for Computational Linguistics.

Chiang, D. (2007). Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

De Smedt, T. and Daelemans, W. (2012). Pattern for python. *J. Mach. Learn. Res.*, 13:2063–2067.

Dyer, C. J. (2007). The'noisier channel': translation from morphologically complex languages. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 207–211. Association for Computational Linguistics.

Habash, N. (2008). Four techniques for online handling of out-of-vocabulary words in arabic english statistical machine translation. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, pages 57–60. Association for Computational Linguistics.

Hajič, J., Hajičová, E., Panevová, J., Sgall, P., Bojar, O., Cinková, S., Fučíková, E., Mikulová, M., Pajas, P., Popelka, J., Semecký, J., Šindlerová, J., Štěpánek, J., Toman, J., Urešová, Z., and Žabokrtský, Z. (2012). Announcing prague czech-english dependency treebank 2.0. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC 2012)*, pages 3153–3160, İstanbul, Turkey. ELRA, European Language Resources Association.

Heafield, K., Pouzyrevsky, I., Clark, J. H., and Koehn, P. (2013). Scalable modified kneser-ney language model estimation. In *ACL (2)*, pages 690–696.

Koehn, P. (2004). Statistical significance tests for machine translation evaluation. Barcelona, Spain.

Koehn, P. (2005). Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pages 79–86.

Koehn, P. and Haddow, B. (2012). Interpolated backoff for factored translation models. In *Proceedings of the Tenth Conference of the Association for Machine Translation in the Americas (AMTA)*.

Mirkin, S., Specia, L., Cancedda, N., Dagan, I., Dymetman, M., and Szpektor, I. (2009). Source-language entailment modeling for translating unknown terms. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 791–799. Association for Computational Linguistics.

Post, M., Cao, Y., and Kumar, G. (2015). Joshua 6: A phrase-based and hierarchical statistical machine translation system. *The Prague Bulletin of Mathematical Linguistics*.

Sylak-Glassman, J., Kirov, C., Yarowsky, D., and Que, R. (2015). A language-independent feature schema for inflectional morphology. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 674–680, Beijing, China. Association for Computational Linguistics.

Tiedemann, J. (2009). News from OPUS - A collection of multilingual parallel corpora with tools and interfaces. In Nicolov, N., Bontcheva, K., Angelova, G., and Mitkov, R., editors, *Recent Advances in Natural Language Processing*, volume V, pages 237–248. John Benjamins, Amsterdam/Philadelphia, Borovets, Bulgaria.

Tyers, F. M. and Alperen, M. S. (2010). South-east european times: A parallel corpus of balkan languages. In *Proceedings of the LREC Workshop on Exploitation of Multilingual Resources and Tools for Central and (South-) Eastern European Languages*, pages 49–53.

Yang, M. and Kirchhoff, K. (2006). Phrase-based backoff models for machine translation of highly inflected languages. In *EACL*, pages 3–7.

# Investigating the Impact of Various Partial Diacritization Schemes on Arabic-English Statistical Machine Translation

**Sawsan Alqahtani, Mahmoud Ghoneim, Mona Diab**

{sawsanq, mghoneim, mtdiab}@gwu.edu

Department of Computer Science

The George Washington University, USA

Washington, DC 20052

**Abstract**

Most diacritics in Arabic represent short vowels. In Arabic orthography, such diacritics are considered optional. The absence of these diacritics naturally leads to significant word ambiguity to top the inherent ambiguity present in fully diacritized words. Word ambiguity is a significant impediment for machine translation. Despite the ambiguity presented by lack of diacritization, context helps ameliorate the situation. Identifying the appropriate amount of diacritic restoration to reduce word sense ambiguity in the context of machine translation is the object of this paper. Diacritic marks help reduce the number of possible lexical word choices assigned to a source word which leads to better quality translated sentences. We investigate a variety of (linguistically motivated) partial diacritization schemes that preserve some of the semantics that in essence complement the implicit contextual information present in the sentences. We also study the effect of training data size and report results on three standard test sets that represent a combination of different genres. The results show statistically significant improvements for some schemes compared to two baselines: text with no diacritics (the typical writing system adopted for Arabic) and text that is fully diacritized.

## 1 Introduction

Resolving natural language ambiguity is at the crux of the NLP enterprise. Ambiguity refers to the problem of possibly having different interpretations for different segments (words, phrases, etc.) of a sentence. Languages such as Arabic, Hebrew and Persian are typically written in a manner that exacerbates this ambiguity problem and increases the homograph rate by underspecifying some of the letters such as short vowels and consonantal gemination, which in turn increases the effect of having multiple interpretations for the same word. This renders text even more ambiguous than typically expected.

While context helps native speakers of the language resolve some of the ambiguity, context alone does not always produce adequate clarity for interpretation. The problem is further complicated in Arabic by the fact that there are no native speakers of Modern Standard Arabic (MSA), which is the language used in education and formal settings. Instead, speakers of Arabic converse in various dialects of Arabic which are at times starkly different from MSA.

One solution for this problem is diacritic restoration, or diacritization, which refers to ren-

dering the underspecified diacritics explicit in the text. We investigate the problem of diacritization within the context of Arabic-to-English Statistical Machine Translation (SMT) system. We address the problem in MSA texts, the majority of which are underspecified for these diacritic marks. We focus here on the most prominent Arabic diacritics which are short vowels (i, u, a), the syllable boundary marker, known as sukoon (o), indefiniteness marker, known as nunation (F, K, N), and the consonantal doubling marker (gemination) known as shadda (∼)[1]. In this study, we aim to investigate what is the appropriate level and type of diacritic restoration that would have the biggest impact on natural language understanding as tested and evaluated via machine translation. Hence we experiment with various diacritization schemes based on lexical and/or syntactic information.

This current work is a follow on to the pilot work presented in (Diab et al., 2007). However it is different in the following respects: 1- we explore automatically diacritized data; 2- we define more schemes that target both lexical and/or syntactic properties of the Arabic language. 3- we test the robustness of our observations taking into consideration varying training size and cross genre evaluation.

## 2 Related Work

Automatic Arabic diacritization has been addressed thoroughly in (Zitouni et al., 2006; Elshafei et al., 2006; Nelken and Shieber, 2005; Habash and Rambow, 2007; Pasha et al., 2014; Maamouri et al., 2008). Full diacritization indicates rendering the text with all the most prominent diacritics, namely (a, i, u, o, ∼).[2] Initial efforts in automatic diacritization include rule-based approaches to add all diacritics in the texts (Debili and Achour, 1998; El-Imam, 2004); however, it is expensive to maintain these rules to be generalized for unseen instances.

Most studies focused on full diacritic restoration. For Automatic Speech Recognition (ASR), (Vergyri and Kirchhoff, 2004; Ananthakrishnan et al., 2005) perform full diacritization on MSA speech transcripts for language modeling. They show that developing ASR models on fully diacritized datasets improves performance significantly. Supervised classifiers such as Hidden Markov Model (HMM) and Maximum Entropy (MaxEnt) have been employed for diacritization (Gal, 2002; Bebah et al., 2014; Zitouni and Sarikaya, 2009; Zitouni et al., 2006). In a study conducted by Ananthakrishnan et al. (2005), the researchers use MaxEnt trained on MSA with lexical and n-gram features to improve ASR. Another study uses decision trees and stochastic language models to fully diacritize texts in order to render graphemes to synthesized speech (Cherif et al., 2015). The Buckwalter Arabic Morphological Analysis (BAMA) (Buckwalter, 2002) system has been used along with a single tagger or a language model to select amongst the diacritized analyses in context to render text fully diacritized (Vergyri and Kirchhoff, 2004; Ananthakrishnan et al., 2005).

In Marton et al. (2010), the authors show that some inflectional and lexical related morphological features improve the performance of syntactic parsing in Arabic. Although Marton et al. (2010) have not used diacritics directly in their work, they use the same essential information that is used to diacritize Arabic texts. Diab et al. (2007) not only investigate the impact of full diacritization on Statistical Machine Translation (SMT) but also introduce the notion of partial diacritization. They also show that several schemes have a small positive effect albeit not significant on SMT performance over none and full diacritization despite the significant increase in the number of types. Although the results in Diab et al. (2007) are not statistically significant, they provide directions of research that we can exploit to increase the performance of Arabic related NLP applications. In a study conducted by AlHanai and Glass (2014), three partial diacritic schemes have been defined and compared to both fully and non-diacritized versions of the

[1] We use Buckwalter Transliteration encoding: http://www.qamus.org/transliteration.htm

[2] In some studies such as (Habash and Rambow, 2007; Pasha et al., 2014), they also address hamza restoration.

words. In their study, it is found that fully-diacritized text without gemination have statistically better performance than fully diacritized texts including gemination in ASR application. Our work follows the same general procedure as (Diab et al., 2007; AlHanai and Glass, 2014) where we study the impact of some aspects of diacritization information in NLP applications, SMT in particular.

For Arabic reading comprehension, Hermena et al. (2015) studies the impact of partial diacritics in improving Arabic speakers' reading comprehension. Their study shows the effectiveness of having some level of diacritization between none and fully diacritized forms that help the readers disambiguate homographs that cannot be understood by the surrounding contexts. This shows the importance of having accurate automatic partial diacritization not only in improving different NLP applications but also to diacritize texts to help readers understand Arabic texts better. Having the goal of helping other researchers develop partial diacritization, Bouamor et al. (2015) has conducted a pilot study that minimally diacritize the dataset to reduce lexical ambiguity and help generate models to find an optimal level of diacritization in some NLP applications. Although the result of this minimally-diacritized annotation has been highly affected by the annotators' subjectivity and background, it has shown some promising results for future studies.

The idea of integrating Word Sense Disambiguation (WSD) technologies into the SMT framework has been studied previously, tackling different aspects of the phenomenon and showing statistically significant improvement integrating explicit WSD into the SMT system (Chan et al., 2007; Carpuat and Wu, 2007; Yang and Kirchhoff, 2012; Aminian et al., 2015). Mainly, WSD integration improves the ability of the system to choose the target translation if it has been incorporated efficiently. Carpuat and Wu (2007) show an improvement in Chinese-to-English SMT system in eight different automatic evaluation metrics when they integrate WSD in their translation system at decode time. They use the same parallel corpus used for training and the phrase translation table generated by the SMT tool to disambiguate senses of the words by using the aligned phrases in the target language. All of the previous work incorporates features that help disambiguate senses in a supervised or unsupervised manner to generate better quality translation. Some of these studies change the SMT pipeline to integrate WSD but others implement it as a pre-processing step at decode time. In this study, we have the same goal as theirs which is to appropriately select the correct sense of a target word at decode time. We implement this by adding a certain amount of diacritics in Arabic as preprocessing in the data preparation step. Thus, the translation quality is not only enhanced by the appropriate choice of target word but also by the fact that the word alignment procedure is improved.

## 3 Scheme Extraction

We investigate the impact of various partial diacritization schemes on SMT application. We compare their performance against two baselines, specifically FULL diacritization where all the diacritics are present and NONE where no diacritics are present. Similar to the extraction strategy of (Diab et al., 2007; AlHanai and Glass, 2014), each of these schemes is identified from fully diacritized Arabic datasets. Additionally, the extraction process of some schemes involves the full morphological analysis of the words' part of speech and their lemmas. To identify these morphological features, we use MADAMIRA, a morphological analyzer and disambiguator for the Arabic language (Pasha et al., 2014). The quality of diacritization schemes rely on the performance of the automatic diacritization to predict diacritics. It is important to note that we rely on the underlying diacritized lemma form for ensuring extraction accuracy.

(Diab et al., 2007) define six different diacritization schemes based on their usage prominence in the Arabic Treebank (ATB) (Maamouri et al., 2008). Namely, they are fully diacritized (FULL), passive voice diacritic marks (PASS), consonant doubling or gemination (GEM), pres-

ence of the syllable boundary marker sukoon (SUK), syntactic case and mood diacritics (CM), and the case of no diacritization (NONE). In this study, we adopt the same previously mentioned schemes in addition to introducing several new ones: FULL-CM, PASS+CM, PASS+GEM, SUK+GEM, PASS+SUK, PASS+SUK+GEM, FULL-CM-PASS, TANWEEN.[3] The following is a detailed explanation of these diacritic schemes.

The schemes are linguistically-motivated reflecting lexical, syntactic, or both types of information. The Arabic sentences are written in Buckwalter Transliteration[4] and are tokenized according to the ATB style (Arabic TreeBank Tokenization). It is crucial to note that if the word is not affected by the defined diacritic pattern, we remove all of its diacritics (i.e. NONE scheme).

**Baselines:** NONE: indicates that no diacritics are kept at all in the sentence, including the removal of the naturally occurring diacritics.

*e.g. w+ ADAft An Tbyb AEln wfAp AlmEtql , bEd An HAwl AtxA* kl AlAjrA'At AllAzmp l+ AnqA* +h .*

FULL: indicates that all diacritics are kept in the sentence.

*e.g. wa+ AaDAfat Aan∼a TabiybAF AaEolana wafApu AlmuEotaqalu , baEoda Aano HAwala Ait∼ixA*a kul∼i AlAijorA'Ati All∼Azimapi li+ AinoqA*i +hu .*

**Singleton Schemes (Lexical):** SUK: is an explicit marking of the absence of a short vowel typically between syllables, known as sukoon. We keep sukoon in the word whenever it is present in the underlying lemma.

*e.g. w+ ADAft An Tbyb AEoln wfAp AlmEotql , bEod Ano HAwl AtxA* kl AlAjorA'At AllAzmp l+ AnoqA* +h .*

GEM: renders explicit the doubling of consonants (shaddah or gemination) whenever the shaddah is present in the underlying lemma of the word.

*e.g. w+ ADAft An∼ Tbyb AEln wfAp AlmEtql , bEd An HAwl At∼xA* kl∼ AlAjrA'At AllAzmp l+ AnqA* +h .*

**Singleton Schemes (Inflectional):** CM (Case and Mood): reflects syntactic case and mood on nominals and verbs, respectively. We keep the last diacritic marker whenever the part of speech explicitly indicates the presence of case or mood.

*e.g. w+ ADAft An TbybAF AEln wfApu AlmEtqlu , bEda An HAwl AtxA*a kl∼i AlAjrA'Ati AllAzmpi l+ AnqA*i +h .*

PASS (passivization): indicates that the diacritic(s) reflecting passive voice on verbs are the only markers kept.

*e.g. w+ qAlt AlqyAdp Aljnwbyp b+ myAmy fy byAn , An jvmAn Almtwfy s+ yurAEay wfq AltqAlyd w+ AlAErAf Aldynyp , Alty yntmy l +hA .*

**Singleton Schemes (Both):** TANWEEN: reflects syntactic case and indefiniteness on nominals. We keep all tanween marks (K, F, N).

*e.g. w+ ADAft An TbybAF AEln wfAp AlmEtql , bEd An HAwl AtxA* kl AlAjrA'At AllAzmp l+ AnqA* +h .*

**Combined Schemes (Lexical):** SUK+GEM: Combines SUK and GEM diacritic schemes.

*e.g. w+ ADAft An∼ Tbyb AEoln wfAp AlmEotql , bEod Ano HAwl At∼xA* kl∼ AlAjorA'At AllAzmp l+ AnoqA* +h .*

[3]Naming Convention: Similar to the mathematical operations, the symbol [-] indicates that we remove the diacritics of the scheme presented after the symbol from the scheme presented before the symbol. The symbol [+] indicates that we add the diacritic scheme for each of these schemes to define a new one.

[4]More description can be found in http://www.qamus.org/transliteration.htm

FULL-CM-PASS: indicates that all diacritics are kept in the word except the syntactic level diacritics.

*e.g. wa+ qAlat AlqiyAdap Aljanuwbiy∼ap bi+ mayAmiy fiy bayAn , Aino juvomAn Almutawaf∼ay sa+ yrAEy wifoq Alt∼aqAliyd wa+ AlAaEorAf Ald∼iyniy∼ap , Al∼atiy yanotamiy li +hA .*

**Combined Schemes (Inflectional):** PASS+CM: combines the properties of PASS and CM schemes.

*e.g. w+ qAlt AlqyAdpu Aljnwbypu b+ myAmy fy byAnK , An jvmAnu Almtwfy s+ yurAEay wfqa AltqAlydi w+ AlAErAfi Aldynypi , Alty yntmy l +hA .*

**Combined Schemes (Both):** FULL-CM: the same as FULL but we remove the CM related diacritics from the word.

*e.g. wa+ AaDAfat Aan∼a Tabiyb AaEolana wafAp AlmuEotaqal , baEod Aano HAwala Ait∼ixAV kul AlAijorACAt All∼Azimap li+ AinoqAV +hu .*

PASS+GEM: Combines the features of PASS and GEM schemes.

*e.g. w+ qAlt AlqyAdp Aljnwby∼p b+ myAmy fy byAn , An jvmAn Almtwf∼y s+ yurAEy wfq AltqAlyd w+ AlAErAf Aldyny∼p , Al∼ty yntmy l +hA .*

PASS+SUK: Combines the features of PASS and SUK schemes.

*e.g. w+ qAlt AlqyAdp Aljnwbyp b+ myAmy fy byAn , Ano jvomAn Almtwfy s+ yurAEay wfoq AltqAlyd w+ AlAErAf Aldynyp , Alty ynotmy l +hA .*

PASS+SUK+GEM: Combines PASS, SUK, and GEM schemes.

*e.g. w+ qAlt AlqyAdp Aljnwby∼p b+ myAmy fy byAn , Ano jvomAn Almtwf∼y s+ yurAEay wfoq AltqAlyd w+ AlAErAf Aldyny∼p , Al∼ty ynotmy l +hA .*

As indicated previously, the goal of the various schemes is to reduce the number of possible choices for translating a sentence by distinguishing meanings at the word level which in turn affect the phrase level. For example, the word 'bEd' can be understood as baEod (Adv), buEod (Noun), baEuda (Verb), baEida (Verb), buEida (Verb-Passive), baE∼ada (Verb), buE∼ida (Verb-Passive) and biEad∼i (Prep+Noun) which means "after;post, yet", "dimension; distance; remoteness", "became distant (Aspect:State)", "became distant (Aspect:Action)", "kept distant to something", "displace; exclude; alienate", "became displaced; excluded; alienated", "by counting", respectively. The diacritic schemes defined here segment the space in different ways to reduce some aspects of this ambiguity. For example, SUK segments the space into two subgroups where both 'baEod' and 'buEod' share the form 'bEod' while others remain 'bEd'. Similarly, GEM creates three segments. It uniquely identifies 'biEad∼i' as 'bEd∼' solving its ambiguity and groups 'baE∼ada' and 'buE∼ida' as 'bE∼d' (still ambiguous, although to a lesser extent) and the remaining five words as 'bEd'. Adding PASS to GEM will solve the ambiguity regarding 'baE∼ada' and 'buE∼ida' (become 'bE∼d' and 'buE∼id', respectively).

While FULL solves all ambiguity, it actually over specifies every word by including Case and Mood which in turn increases data sparsity. For example the word 'buEod' can take the following forms 'buEoda', 'buEodi', 'buEodu', 'buEodK', 'buEodN', 'buEod' according to the FULL scheme. FULL-CM would decrease this sparseness but again there is still some redundancy (e.g. 'baEod' can take the forms 'baEodu' 'baEodi' 'baEoda' as the last diacritic is neither a Case nor a Mood marker). The ability of these schemes to disambiguate is sensitive to the genre of the text where some variations might not appear or become rarely used. Also sparsity would be a limiting factor for small training data sizes. The objective of this study is to explore the appropriate level of diacritization information that reduces ambiguity to practical levels within the context of SMT. As it is hard to define these practical levels, we report results on different training data sizes and using test sets that exhibit different combinations of genres. It is important to note that this study does not aim at distinguishing the different possible senses

of the word with diacritics specified (fully diacritized). To illustrate, the word 'buEod' may mean dimension, distance or remoteness; this level of disambiguation cannot be addressed using diacritics alone but depends on context which is not addressed directly by this work, but is assumed to be taken care of through the SMT pipeline as a whole.

## 4 Experimental Setup

### 4.1 Dataset

To train the SMT model, we use an Arabic-English parallel dataset which includes 60M tokens and is created from 53 LDC (Linguistic Data Consortium) catalogs. This dataset includes multiple genres such as newswire, broadcast news, broadcast conversations, newsgroups, and weblogs. We use three different test datasets from multiple genres: NIST 2009, 2006, and 2005 Open Machine Translation Evaluation,[5] which correspond to MT09, MT06, and MT05, respectively. MT09 is 41,640 tokens from weblogs and newswires; MT06 consists of 49,154 tokens from newswire, broadcast news, and weblogs; MT05 consists of 33,407 tokens from newswire. We use NIST 2008 Open Machine Translation Evaluation (MT08),[6] which is 45,555 tokens from newswire and web collection genres, for tuning. It is important to note that the number of types varies across diacritic schemes as opposed to the number of tokens which is consistent for all schemes. Table 1 shows the number of types for each of the train and test datasets associated with each diacritic scheme.

For both Arabic and English datasets, we separate punctuations and numbers from the text and convert them to standard forms (PUNC and NUM) in order to reduce the number of types and errors to some extent. We use the morphological analyzer toolkit, MADAMIRA (Pasha et al., 2014) to tokenize the Arabic side of the parallel dataset according to Arabic Treebank tokenization (ATB) style (Maamouri et al., 2004). All diacritic patterns have the same exact preprocessing; the only difference is the number and the type of diacritics added to the dataset. For the English side of the parallel dataset, we tokenize the dataset using Tree Tagger (Schmid, 1995) and lowercase all letters.

| Diacritic Pattern | No. of Types (Train) | Type Increase % (Train) | No. of Types (MT09/MT06/MT05) | Type Increase % (MT09/MT06/MT05) |
|---|---|---|---|---|
| NONE | 303,049 | - | 8,562 / 9,205 / 6,128 | - |
| FULL | 432,832 | 42.83 | 11,072 / 12,027 / 7,966 | 29.32 / 30.66 / 29.99 |
| SUK | 306,648 | 1.19 | 8,644 / 9,324 / 6,186 | 0.96 / 1.29 / 0.95 |
| GEM | 308,424 | 1.77 | 8,638 / 9,312 / 6,175 | 0.89 / 1.16 / 0.77 |
| CM | 414,615 | 36.81 | 10,936 / 11,845 / 7,868 | 27.73 / 28.68 / 28.39 |
| PASS | 306,003 | 0.97 | 8,603 / 9,237 / 6,155 | 0.48 / 0.35 / 0.44 |
| TANWEEN | 342,025 | 12.86 | 9,363 / 10,134 / 6,720 | 9.36 / 10.09 / 9.66 |
| SUK+GEM | 311,024 | 2.63 | 8,702 / 9,400 / 6,220 | 1.64 / 2.12 / 1.50 |
| FULL-CM-PASS | 329,123 | 8.60 | 8,912 / 9,611 / 6,337 | 4.09 / 4.41 / 3.41 |
| PASS+CM | 417,876 | 37.89 | 10,969 / 11,869 / 7,892 | 28.11 / 28.94 / 28.79 |
| FULL-CM | 329,632 | 8.77 | 8,939 / 9,652 / 6,359 | 4.40 / 4.86 / 3.77 |
| PASS+GEM | 311,202 | 2.69 | 8,676 / 9,344 / 6,201 | 1.33 / 1.51 / 1.19 |
| PASS+SUK | 309,499 | 2.13 | 8,683 / 9,353 / 6,211 | 1.41 /1.61 / 1.35 |
| PASS+SUK+GEM | 313,788 | 3.54 | 8,739 / 9,429 / 6,245 | 2.07 / 2.43 / 1.91 |

Table 1: This table shows the number of types for each diacritic scheme for test and train datasets. Type Increase columns indicate the percentage of increase in the number of types compared to NONE.

[5]Catalog Numbers: LDC2010T23 (MT09), LDC2010T17 (MT06), LDC2010T14 (MT05).
[6]Catalog Number: LDC2010T21.

### 4.2 SMT System

We train standard phrase-based SMT system using Moses toolkit version 2.1.1 (Koehn et al., 2007). The parallel corpus is word aligned using GIZA++ (Och and Ney, 2003) with a maximum sentence length of 250 words. The phrase tables contains up to 8-words phrases. We use SRILM (Stolcke et al., 2002) to build 5-gram language model with modified Kneser-Ney smoothing (James, 2000). Our language modeling data consists of the English Gigaword 5th edition LDC2011T07 and the English side of the training datasets. The best weight parameters are tuned using the Minimum Error Rate Training (MERT) algorithm (Och, 2003) to maximize BLEU score (Papineni et al., 2002). To account for optimization algorithm instability, we replicate optimization three times per experiment. We use bootstrap resampling and approximate randomization (Clark et al., 2011) to statistically test for significant differences using two evaluation metrics: BLEU (Papineni et al., 2002) and TER (Snover et al., 2006). As BLEU reflects a bias toward fluency in the target language and TER identifies the least post editing, they capture complementary aspects of the translation. We consider NONE and FULL as the baselines which show the the impact of under- and over-specification of the diacritics. As discussed before, NONE accounts for the dataset without any diacritics added (consonants only) which is the default setting for most current SMT systems whereas FULL shows the impact of all automatically generated lexical and syntactic diacritic marks.

## 5 Results & Discussion

| **Diacritic Pattern** | **BLEU ↑** | **TER ↓** | **BLEU ↑** | **TER ↓** | **BLEU ↑** | **TER ↓** |
|---|---|---|---|---|---|---|
| **Dataset** | **MT09** | | **MT06** | | **MT05** | |
| **Baselines** | | | | | | |
| NONE | 47.0 ◇ | 45.5 | 25.4 | 56.5 | 27.9 | 48.0 |
| FULL | 46.7 | 45.4 | 25.3 | 56.3 | 27.9 | 47.7 ● |
| **Singleton Schemes (*Lexical*)** | | | | | | |
| SUK | 47.0 ◇ | 45.5 | 25.4 | 56.5 | 27.8 | 48.1 |
| GEM | 47.2 ● ◇ | 45.3 ● | 25.5 ◇ | 56.0 ● ◇ | 27.9 | 47.6 ● |
| **Singleton Schemes (*Inflectional*)** | | | | | | |
| CM | 46.9 | 45.5 | 25.2 | 56.7 | 27.8 | 48.1 |
| PASS | 47.1 ◇ | 45.4 | 25.4 | 56.0 ● ◇ | 27.9 | 47.8 ● |
| **Singleton Schemes (*Both*)** | | | | | | |
| TANWEEN | 46.7 | 45.9 | 25.3 | 56.7 | 27.9 | 48.1 |
| **Combined Schemes (*Lexical*)** | | | | | | |
| SUK+GEM | 47.2 ◇ | 45.4 | 25.4 | 56.2 ● | 27.9 | 48.3 |
| FULL-CM-PASS | 47.4 ● ◇ | 45.2 ● ◇ | 25.5 ● ◇ | 55.9 ● ◇ | 28.0 | 47.7 ● |
| **Combined Schemes (*Inflectional*)** | | | | | | |
| PASS+CM | 47.0 ◇ | 45.5 | 25.3 | 56.5 | 27.7 | 48.0 |
| **Combined Schemes (*Both*)** | | | | | | |
| FULL-CM | 47.2 ● ◇ | 45.3 | 25.5 ● ◇ | 56.2 ● | 28.0 | 47.9 |
| PASS+GEM | 47.5 ● ◇ | 45.1 ● ◇ | 25.7 ● ◇ | 56.0 ● ◇ | 28.1 ● ◇ | 47.6 ● |
| PASS+SUK | 47.3 ● ◇ | 45.3 ● | 25.3 | 56.4 | 27.9 | 48.0 |
| PASS+SUK+GEM | 47.1 ◇ | 45.3 | 25.3 | 56.3 ● | 27.9 | 47.8 ● |

Table 2: This table shows the SMT performance using BLEU and TER evaluation metrics. The symbol ● indicates statistically significant improvement compared to NONE; the symbol ◇ indicates statistically significant improvement compared to FULL.

The main goal of this study is to investigate the relative performance between the different diacritic schemes on the Arabic-English SMT. We use $p$-value $<0.05$ as the level of significance. Generally speaking, PASS+GEM, which involves both lexical and inflectional information, has significantly higher results than both baselines across both metrics; exception can be found in MT05 using TER metric where the PASS+GEM has comparable performance to FULL. FULL-CM-PASS, which includes all semantic distinguishing diacritics, has also significantly higher

results in MT09 and MT06, compared to the baselines across all metrics. In MT05, FULL-CM-PASS significantly outperforms NONE using the TER metric only; however, it has comparable performance to both baselines using the BLEU metric. Additionally, GEM has a considerably significant performance in a fair number of experiments. TANWEEN and CM are the worst performing models because it could not outperform the baselines in any of the datasets using any of the metrics. This is expected since both CM and TANWEEN reflect a relatively low lexical semantic variation compared to other schemes.

Furthermore, although the explicit marker for the absence of diacritic, SUK, in the Arabic vocabulary plays a major role in distinguishing meaning, it does not yield competitive results against either baseline except in one experiment. The same finding can be observed in PASS+CM which covers inflectional properties only. Comparing the baselines with each other, NONE seems to have higher results using the BLEU metric although the increase of the performance is significant only in MT09. On the other hand, FULL yields higher results than NONE using TER metric; the increase in performance is also not significant except in MT05.

| | **Type OOV Rate** | | | **Number of Types** | **Token OOV Rate** | | |
|---|---|---|---|---|---|---|---|
| **Diacritic Pattern** | **MT09** | **MT06** | **MT05** | **(MT09/MT06/MT05)** | **MT09** | **MT06** | **MT05** |
| NONE | 2.51% | 4.36% | 1.29% | 8,562 / 9,205 / 6,128 | 1.06% | 1.30% | 0.31% |
| FULL | 2.83% | 4.37% | 1.39% | 11,072 / 12,027 / 7,966 | 1.29% | 1.58% | 0.40% |
| **Singleton Schemes (*Lexical*)** | | | | | | | |
| SUK | 2.50% | 4.31% | 1.28 % | 8,644 / 9,324 / 6,186 | 1.06% | 1.31% | 0.31% |
| GEM | 2.54% | 4.37% | 1.31% | 8,638 / 9,312 / 6,175 | 1.07% | 1.32% | 0.32% |
| **Singleton Schemes (*Inflectional*)** | | | | | | | |
| CM | 2.68% | 4.20% | 1.28% | 10,936 / 11,845 / 7,868 | 1.25% | 1.52% | 0.37% |
| PASS | 2.52% | 4.43% | 1.28 % | 8,603 / 9,237 / 6,155 | 1.07% | 1.32% | 0.31% |
| **Singleton Schemes (*Both*)** | | | | | | | |
| TANWEEN | 2.51% | 4.24% | 1.29% | 9,363 / 10,134 / 6,720 | 1.10% | 1.36% | 0.34% |
| **Combined Schemes (*Lexical*)** | | | | | | | |
| SUK+GEM | 2.54% | 4.33% | 1.30% | 8,702 / 9,400 / 6,220 | 1.08% | 1.32% | 0.32% |
| FULL-CM-PASS | 2.72% | 4.46% | 1.36% | 8,912 / 9,611 / 6,337 | 1.13% | 1.37% | 0.33% |
| **Combined Schemes (*Inflectional*)** | | | | | | | |
| PASS+CM | 2.69% | 4.28% | 1.28% | 10,969 / 11,869 / 7,892 | 1.25% | 1.54% | 0.37% |
| **Combined Schemes (*Both*)** | | | | | | | |
| FULL-CM | 2.73% | 4.46% | 1.35% | 8,939 / 9,652 / 6,359 | 1.13% | 1.37% | 0.33% |
| PASS+GEM | 2.55% | 4.44% | 1.31% | 8,676 / 9,344 / 6,201 | 1.08% | 1.33% | 0.32% |
| PASS+SUK | 2.51% | 4.38% | 1.27% | 8,683 / 9,353 / 6,211 | 1.07% | 1.32% | 0.31% |
| PASS+SUK+GEM | 2.55% | 4.40% | 1.30% | 8,739 / 9,429 / 6,245 | 1.08% | 1.33% | 0.32% |

Table 3: This table shows the rate of type and token OOV in the test sets for each diacritic scheme. For convenience, we also add the number of types (the same information as Table 1). The number of tokens for each dataset is: 41,640 tokens for MT09, 49,154 tokens for MT06, and 33,407 tokens for MT05.

Obviously, the number of type increase for each diacritic scheme in each dataset follows the same trend. NONE has the lowest number of types while FULL has the highest; the remaining diacritic schemes lie in between. When we compare the number of types and the performance of the diacritic schemes, we can see that there is a level of number of types between NONE and FULL that achieves good performance in distinguishing lexical meaning. This suggests that the increase of the number of types to some extent between NONE and FULL is acceptable with adequate amount of tokens/types in the training phase. This increase must provide appropriate lexical signals to enhance the overall performance because providing redundant lexical/inflectional signals may also degrade the performance. The number of types in the best performing diacritic schemes is closer to NONE as we can see from Table 1. On the other hand, we can observe that schemes that did not contribute to distinguishing the meaning have relatively high number of types compared to NONE (i.e. the number of types in such

schemes is close to that in FULL). The increase in out of vocabulary (OOV) tokens follows the same trend as the number of types as shown in Table 3. They have comparative rate that ranges between 1.06% to 1.29% in MT09, 1.30% to 1.58% in MT06, and 0.31% to 0.40% in MT05. Because all diacritic schemes have the same number of tokens, the slight increase in OOV rate shows the impact of the diacritic scheme coverage on the test dataset.

Looking at the performance of the diacritic schemes in each dataset, it is observed that MT05 has not been affected by any of them. The only diacritic scheme that outperforms both baselines in this dataset is PASS+GEM with a 0.2 BLEU point improvement. However, although the results in MT05 are not significant, almost all diacritic schemes have comparable performance to NONE and FULL. Using the BLEU metric, MT09 benefits the most from these diacritics schemes: GEM, FULL-CM-PASS, FULL-CM, PASS+GEM, and PASS+SUK. It is unclear whether the genres in each dataset plays a major role here; MT05 is extracted from newswire collection only while MT09 and MT06 includes both newswire and web collection. Moreover, We can observe from Table 1 that the number of types in MT09 and MT06 considerably higher than MT05 which may also be a factor.

The overall performance for the defined diacritic schemes shows potential improvement which would enhance the SMT system performance to some degree. Because the phrase-based SMT system implicitly takes context of the word into its consideration, we believe that developing more sophisticated schemes that recognize context would have even a more significant impact on SMT performance especially in distinguishing words with much less sparsity.

| **Label** | **NONE** | **GEM** |
|---|---|---|
| **Source Sentence** | E$ kl lHZp kAn +hA Axr lHZp fy HyAp +k | E$ kl∼ lHZp kAn∼ +hA Axr lHZp fy HyAp +k |
| **Target Sentence** | live every moment was the last moment in your life | live every moment as if it were the last moment in your life |
| **Gold References** | 1: Live each moment as though it is the last moment in your life .<br>2: Live every moment as if it was the last moment of your life .<br>3: Live each moment as though it were the last moment of your life .<br>4: Live every moment as if it's the last moment of your life . | |
| **Label** | **NONE** | **FULL-CM** |
| **Source Sentence** | w+ hl wDEt qmp brwksyl AlAxyrp | wa+ halo waDaEat qim∼ap bruwkosiyl AlAaxiyrap Had∼ |
| | Hd l+ Hlm Aldstwr AlAwrwby ? | li+ Hulom Ald∼usotuwr AlAuwruwb∼iy ? |
| **Target Sentence** | do you put the recent summit in brussels | and whether the recent brussels summit put an end to |
| | according to the dream of the european constitution | the dream of the european constitution |
| **Gold References** | 1: and has the latest brussels summit put an end to the dream of a european constitution<br>2: did the latest summit in brussels put an end to the dream of a european constitution<br>3: did the recent brussels summit put an end to the dream of a european constitution<br>4: has the last brussels summit put an end to the dream of a european constitution | |

Table 4: This table shows the outputs for NONE and GEM systems for the first sentence; the second sentence shows the output for NONE and FULL-CM, along with their gold references.

Table 4 shows an output example from GEM and PASS+GEM systems compared to the baseline NONE to illustrate our intuition behind specifying partial diacritic schemes. The word 'kAn' [7] may have the meanings 'was' or 'as if it was/were' as produced by NONE and GEM respectively. Adding ∼ to this word makes the system consider 'kAn' and 'kAn∼' as distinct words with different lexical meanings. Although FULL produces the same translation output for the first sentence as GEM, FULL's overall performance is less than most of the diacritic patterns and it causes the dataset to be extremely sparse. For the second sentence, we compare the target translation for NONE and FULL-CM. NONE considers different sense for the word "wDEt" which is "waDaEota" (means 'you put') whereas the correct sense for this word is

[7] We normalize all forms of letter Alef (A,|,<,>) to 'A' during our preprocessing. Also, words in the datasets may exhibit tokenization and preprocessing errors generated by MADAMIRA such as the word 'kAn' which should have been tokenized as 'k+ An".

”waDaEat” (means ’it/she puts’) as appears in FULL-CM. Additionally, NONE translation of ”Hd” into ’according’ is not the accurate choice whereas the lexical target choice has been accurately predicated in the FULL-CM (’an end’).

### 5.1 Training Size Effect

To tease apart the training size effect on performance, we conduct the same experiments using 30% and 50% of tokens in the training dataset chosen randomly, and then compare their performance with the full training size. Table 5 shows the training number of types for 30% and 50% of the dataset for each diacritic scheme. Results in Table 6 illustrates the drop in the overall performance using smaller amount of training data for almost all the experimental conditions. As we decrease the training size, the influential triggers in the diacritic patterns become less impactful compared to NONE and do not contribute to the performance except in few cases. Most of the diacritic schemes in MT05 outperform FULL, as we decrease the training size, which is expected since FULL introduces the most sparsity in the experimental set up.

| Diacritic Pattern | 100% | 50% | 30% |
|---|---|---|---|
| NONE | 303,049 | 232,282 | 186,791 |
| FULL | 432,832 | 336,951 | 266,819 |
| SUK | 306,648 | 235,322 | 187,858 |
| GEM | 308,424 | 236,682 | 188,899 |
| CM | 414,615 | 322,722 | 258,234 |
| PASS | 306,003 | 234,613 | 187,256 |
| TANWEEN | 342,025 | 264,595 | 211,293 |
| SUK+GEM | 311,024 | 238,878 | 190,627 |
| FULL-CM-PASS | 329,123 | 253,209 | 201,258 |
| PASS+CM | 417,876 | 325,260 | 260,187 |
| FULL-CM | 329,632 | 253,751 | 201,804 |
| PASS+GEM | 311,202 | 238,882 | 190,570 |
| PASS+SUK | 309,499 | 237,575 | 189,568 |
| PASS+SUK+GEM | 313,788 | 241,058 | 192,282 |

Table 5: This table shows the number of types for each diacritic scheme in the training data for each proportional experiment.

Reducing to 50% of the training data size, we still maintain some experimental conditions that achieve significantly higher results than the baselines. GEM and PASS+SUK+GEM provides significant higher results than both baselines across all datasets. FULL-CM-PASS also outperforms both baselines in MT09 and MT06 only. The number of diacritic schemes that outperform NONE in MT05 increases compared to using the full training data. For the 30% training condition, there are four diacritic schemes that significantly outperform NONE in MT09; namely, SUK, GEM, PASS, and their combination PASS+SUK+GEM. None of the diacritic schemes achieve significantly higher results than NONE in both MT06 and MT05, except FULL-CM in MT05 with 0.2 improvement. MT05 and MT06 actually show significantly worse results for several of the diacritic schemes that involve inflectional diacritics: FULL, CM, PASS+CM in case of both datasets in addition to PASS and PASS+GEM in MT06. We can observe that each proportion of the dataset exhibits different diacritic schemes that outperform the baselines with FULL-CM-PASS always outperforming both baselines although not consistently statistically significant.

### 5.2 Automatic Diacritization Performance

The impact of the diacritic schemes on the Arabic-to-English SMT performance is highly dependent on the accuracy of the automatic Arabic diacritization toolkit in addition to its performance in part of speech tagging. Evaluating their performance in the newswire genre, MADAMIRA reports an overall error rate of 13.7% for FULL diacritization and tokenization

| Dataset | MT09 | | | MT06 | | | MT05 | | |
|---|---|---|---|---|---|---|---|---|---|
| **Diacritic Pattern** | **100%** | **50%** | **30%** | **100%** | **50%** | **30%** | **100%** | **50%** | **30%** |
| NONE | 47.0 ◇ | 46.9 ◇ | 45.2 ◇ | 25.4 | 25.3 ◇ | 24.3 ◇ | 27.9 | 28.0 | 27.3 ◇ |
| FULL | 46.7 | 46.3 | 44.4 | 25.3 | 24.9 | 23.2 | 27.9 | 27.8 | 26.6 |
| SUK | 47.0 ◇ | 47.1 ◇ | 46.0 ●◇ | 25.4 | 25.4 ◇ | 24.4 ◇ | 27.8 | 28.2 ●◇ | 27.3 ◇ |
| GEM | 47.2 ●◇ | 47.2 ●◇ | 45.5 ●◇ | 25.5 ◇ | 25.6 ●◇ | 24.2 ◇ | 27.9 | 28.3 ●◇ | 27.3 ◇ |
| CM | 46.9 | 46.4 | 44.4 | 25.2 | 24.8 | 23.9 ◇ | 27.8 | 27.7 | 27.0 ◇ |
| PASS | 47.1 ◇ | 46.9 ◇ | 45.6 ●◇ | 25.4 | 25.2 ◇ | 24.0 ◇ | 27.9 | 28.1 ◇ | 27.2 ◇ |
| TANWEEN | 46.7 | 46.8 ◇ | 45.1 ◇ | 25.3 | 25.3 ◇ | 24.1 ◇ | 27.9 | 27.9 | 27.3 ◇ |
| SUK+GEM | 47.2 ◇ | 47.0 ◇ | 45.3 ◇ | 25.4 | 25.1 ◇ | 24.3 ◇ | 27.9 | 28.2 ◇ | 27.3 ◇ |
| FULL-CM-PASS | 47.4 ●◇ | 47.2 ●◇ | 45.4 ◇ | 25.5 ●◇ | 25.5 ●◇ | 24.3 ◇ | 28.0 | 28.1 ◇ | 27.4 ◇ |
| PASS+CM | 47.0 ◇ | 46.3 | 44.4 | 25.3 | 24.5 | 23.7 ◇ | 27.7 | 27.8 | 26.9 ◇ |
| FULL-CM | 47.2 ●◇ | 46.9 ◇ | 45.2 ◇ | 25.5 ●◇ | 25.2 ◇ | 24.3 ◇ | 28.0 | 28.1 ◇ | 27.5 ●◇ |
| PASS+GEM | 47.5 ●◇ | 47.3 ●◇ | 45.3 ◇ | 25.7 ●◇ | 25.2 ◇ | 23.9 ◇ | 28.1 ●◇ | 28.0 | 27.1 ◇ |
| PASS+SUK | 47.3 ●◇ | 46.8 ◇ | 45.5 ◇ | 25.3 | 25.0 | 24.3 ◇ | 27.9 | 28.1 | 27.4 ◇ |
| PASS+SUK+GEM | 47.1 ◇ | 47.2 ●◇ | 45.53 ●◇ | 25.3 | 25.5 ●◇ | 24.2 ◇ | 27.9 | 28.3 ●◇ | 27.4 ◇ |

Table 6: This table shows the SMT performance using BLEU for each test dataset according to the different training sizes. For convenience, we repeat the numbers for the whole dataset (100%). The symbol ● indicates statistically significant improvement compared to NONE; the symbol ◇ indicates statistically significant improvement compared to FULL.

and 3.9% for part of speech tagging error rate. To assess the performance of MADAMIRA in diacritization for each of the schemes, we use LDC Arabic Treebank (ATB) corpora from some genres that are used in our SMT model, which is approximately 500,510 tokens: broadcast news which includes ATB 5, 7, 10, and 12 in addition to web collection data which includes ATB 6 and 11. This corpora provides many gold morphological features including diacritization and tokenization. We exclude newswire collection from our evaluation because MADAMIRA is trained on this genre; thus, we try to avoid the performance bias towards newswire.

| **Diacritic Pattern** | NONE | FULL | SUK | GEM | CM | PASS | TANWEEN |
|---|---|---|---|---|---|---|---|
| **Diacritization Error** | - | 9.35% | 0.73% | 0.67% | 6.77% | 0.17% | 2.82% |
| **Diacritic Pattern** | SUK+GEM | FULL-CM-PASS | PASS+CM | FULL-CM | PASS+GEM | PASS+SUK | PASS+SUK+GEM |
| **Diacritization Error** | 1.17% | 3.42% | 6.9% | 3.62% | 0.82% | 0.87% | 1.3% |

Table 7: This table shows MADAMIRA diacritization error for each of the linguistically motivated schemes. The diacritization error excludes the tokenization error rate and considers the word to be a match if they have the same tokenization and diacritization.

In this evaluation, we consider the words in both the automatically tagged corpus and the gold data to be a match if they have the exact same tokenization and diacritization. The tokenization F1 score for this dataset is 96.40% which is the same tokenization performance score for all schemes. Table 7 shows the diacritization error for each of the diacritization schemes which takes the frequency of words that are affected by the associated diacritic pattern into its consideration. We can observe from Table 7 that FULL has the highest error rate followed by CM and PASS+CM. Removing CM ( or CM+PASS) related diacritics from FULL substantially decreases the error rate compared to FULL. SUK, GEM, and PASS related scheme have the highest performance. Hence, we can say that case and mood related diacritics are the hardest diacritics to predict using MADAMIRA; this suggests a strong correlation with the underperformance of such schemes that involve CM on SMT. As we discussed previously, diacritic schemes are built from automated sources for tokenization, diacritization, part of speech tagging, and lemmatization. This reliance on external morphological analyzers and disambiguators easily propagates errors to higher levels especially in pipelined set ups. However, it is also not realistic to rely on gold sources as they are expensive and limited in their sizes.

## 6 Conclusion

We investigated the impact of various linguistically motivated partial diacritization schemes on Arabic-to-English SMT performance. We find that PASS+GEM and FULL-CM-PASS have statistically higher results than two robust baselines using several SMT metrics despite significant increase in the number of types in the data sets which indicates that such diacritization schemes are capturing and modeling important information at a more appropriate level of granularity. There are other diacritic schemes that perform well in some of the evaluation metrics and datasets. Additionally, having a relatively large dataset has a significant impact on performance. Moreover improving the underlying diacritization technology will probably have a significant impact on performance.

## 7 Acknowledgements

We thank Abdelati Hawwari and the three anonymous reviewers for their valuable comments and suggestions. This publication was made possible by grant NPRP 6-1020-1-199 from the Qatar National Research Fund (a member of Qatar Foundation). The statements made herein are solely the responsibility of the author[s].

## References

AlHanai, T. and Glass, J. (2014). Lexical modeling for Arabic ASR: A systematic approach. *Proceedings of INTERSPEECH*.

Aminian, M., Ghoneim, M., and Diab, M. (2015). Unsupervised false friend disambiguation using contextual word clusters and parallel word alignments. *Syntax, Semantics and Structure in Statistical Translation*, page 39.

Ananthakrishnan, S., Narayanan, S., and Bangalore, S. (2005). Automatic diacritization of Arabic transcripts for automatic speech recognition. In *Proceedings of the 4th International Conference on Natural Language Processing*, pages 47–54.

Bebah, M., Amine, C., Azzeddine, M., and Abdelhak, L. (2014). Hybrid approaches for automatic vowelization of Arabic texts. *International Journal on Natural Language Computing (IJNLC)*.

Bouamor, H., Zaghouani, W., Diab, M., Obeid, O., Oflazer, K., Ghoneim, M., and Hawwari, A. (2015). A pilot study on Arabic multi-genre corpus diacritization annotation. In *ANLP Workshop 2015*, page 80.

Buckwalter, T. (2002). Buckwalter Arabic morphological analyzer version 1.0. *Linguistic Data Consortium*.

Carpuat, M. and Wu, D. (2007). Improving statistical machine translation using word sense disambiguation. In *EMNLP-CoNLL*, volume 7, pages 61–72.

Chan, Y. S., Ng, H. T., and Chiang, D. (2007). Word sense disambiguation improves statistical machine translation. In *Annual Meeting-Association for Computational Linguistics*, volume 45, page 33.

Cherif, W., Madani, A., and Kissi, M. (2015). Towards an efficient opinion measurement in Arabic comments. *Procedia Computer Science*, 73:122–129.

Clark, J. H., Dyer, C., Lavie, A., and Smith, N. A. (2011). Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 176–181. Association for Computational Linguistics.

Debili, F. and Achour, H. (1998). Voyellation automatique de l'arabe. In *Proceedings of the Workshop on Computational Approaches to Semitic Languages*, pages 42–49. Association for Computational Linguistics.

Diab, M., Ghoneim, M., and Habash, N. (2007). Arabic diacritization in the context of statistical machine translation. In *Proceedings of MT-Summit*.

El-Imam, Y. A. (2004). Phonetization of Arabic: Rules and algorithms. *Computer Speech & Language*, 18(4):339–373.

Elshafei, M., Al-Muhtaseb, H., and Alghamdi, M. (2006). Statistical methods for automatic diacritization of Arabic text. In *The Saudi 18th National Computer Conference. Riyadh*, volume 18, pages 301–306.

Gal, Y. (2002). An HMM approach to vowel restoration in Arabic and Hebrew. In *Proceedings of the ACL-02 Workshop on Computational Approaches to Semitic Languages*, pages 1–7. Association for Computational Linguistics.

Habash, N. and Rambow, O. (2007). Arabic diacritization through full morphological tagging. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, pages 53–56. Association for Computational Linguistics.

Hermena, E. W., Drieghe, D., Hellmuth, S., and Liversedge, S. P. (2015). Processing of Arabic diacritical marks: Phonological–syntactic disambiguation of homographic verbs and visual crowding effects. *American Psychological Association*.

James, F. (2000). Modified Kneser-Ney smoothing of n-gram models. *Research Institute for Advanced Computer Science, Tech. Rep. 00.07*.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., et al. (2007). Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, pages 177–180. Association for Computational Linguistics.

Maamouri, M., Bies, A., Buckwalter, T., and Mekki, W. (2004). The penn Arabic treebank: Building a large-scale annotated Arabic corpus. In *NEMLAR conference on Arabic language resources and tools*, volume 27, pages 466–467.

Maamouri, M., Bies, A., and Kulick, S. (2008). Enhancing the Arabic treebank: A collaborative effort toward new annotation guidelines. In *LREC*.

Marton, Y., Habash, N., and Rambow, O. (2010). Improving Arabic dependency parsing with lexical and inflectional morphological features. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 13–21. Association for Computational Linguistics.

Nelken, R. and Shieber, S. M. (2005). Arabic diacritization using weighted finite-state transducers. In *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages*, pages 79–86. Association for Computational Linguistics.

Och, F. J. (2003). Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 160–167. Association for Computational Linguistics.

Och, F. J. and Ney, H. (2003). A systematic comparison of various statistical alignment models. *Computational linguistics*, 29(1):19–51.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318. Association for Computational Linguistics.

Pasha, A., Al-Badrashiny, M., Diab, M., Kholy, A. E., Eskander, R., Habash, N., Pooleery, M., Rambow, O., and Roth, R. (2014). MADAMIRA: A fast, comprehensive tool for morphological analysis and disambiguation of Arabic. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*. European Language Resources Association (ELRA).

Schmid, H. (1995). Improvements in part-of-speech tagging with an application to German. In *Proceedings of the ACL SIGDAT-Workshop*.

Snover, M., Dorr, B., Schwartz, R., Micciulla, L., and Makhoul, J. (2006). A study of translation edit rate with targeted human annotation. In *Proceedings of Association for Machine Translation in the Americas*, volume 200.

Stolcke, A. et al. (2002). SRILM-an extensible language modeling toolkit. In *Interspeech*, volume 2002, page 2002.

Vergyri, D. and Kirchhoff, K. (2004). Automatic diacritization of Arabic for acoustic modeling in speech recognition. In *Proceedings of the Workshop on Computational Approaches to Arabic Script Based Languages*, pages 66–73. Association for Computational Linguistics.

Yang, M. and Kirchhoff, K. (2012). Unsupervised translation disambiguation for cross-domain statistical machine translation. In *Proceedings of Association for Machine Translation in the Americas*.

Zitouni, I. and Sarikaya, R. (2009). Arabic diacritic restoration approach based on maximum entropy models. *Computer Speech & Language*, 23(3):257–276.

Zitouni, I., Sorensen, J. S., and Sarikaya, R. (2006). Maximum entropy based restoration of Arabic diacritics. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, pages 577–584. Association for Computational Linguistics.