Evaluation of Terminology Translation in Instance-Based Neural MT Adaptation

M. Amin Farajian[1,2], **Nicola Bertoldi**[1], **Matteo Negri**[1], **Marco Turchi**[1], **Marcello Federico**[1]
[1] Fondazione Bruno Kessler, Trento, Italy
[2] University of Trento, Trento, Italy
{farajian,bertoldi,negri,turchi,federico}@fbk.eu

Abstract

We address the issues arising when a neural machine translation engine trained on generic data receives requests from a new domain that contains many specific technical terms. Given training data of the new domain, we consider two alternative methods to adapt the generic system: *corpus-based* and *instance-based* adaptation. While the first approach is computationally more intensive in generating a domain-customized network, the latter operates more efficiently at translation time and can handle on-the-fly adaptation to multiple domains. Besides evaluating the generic and the adapted networks with conventional translation quality metrics, in this paper we focus on their ability to properly handle domain-specific terms. We show that instance-based adaptation, by fine-tuning the model on-the-fly, is capable to significantly boost the accuracy of translated terms, producing translations of quality comparable to the expensive corpus-based method.

1 Introduction

When deployed in production lines, machine translation (MT) systems need to serve requests from various domains (*e.g.* legal, medical, finance, sports, etc.) with a variety of structural and lexical differences. Considering that technical translation (*e.g.* user guides, medical reports, etc.) represents the largest share in the translation industry (Kingscott, 2002) and that a significant part of it deals with domain-specific terms, it is important that machine translation delivers not only generic quality but also accurate translations of terms. The possibility of bearing different meanings in different contexts increases the difficulty of translating terms, making it an interesting and challenging topic in MT. Table 1 shows two examples in which Google Translate[1] (GT) and Bing translator[2] (BT) wrongly translate domain terminology. In the first example, the English word *apple* is wrongly recognized and translated as a term of the computer domain (*apple*) while it actually refers to the fruit type (*mele*). In the second example, on the contrary, Bing fails to recognize the multi-word term *broken Windows* by producing instead a literal translation that departs from the original sense. These examples show that existing MT systems still have difficulties in handling domain-specific terms, which calls for solutions to improve this aspect of MT.

Ideal solutions for this real-world multi-domain translation scenario should be scalable enough to enable the industrial deployment at a reasonable cost, while guaranteeing a high level of flexibility in delivering good-quality translations for all (or most of) the domains. This is of higher importance for the neural approach, where building the systems usually requires expensive GPU machines trained for several days to weeks on large amounts of parallel data.

In this paper we analyze the ability of instance-based adaptation strategy in handling domain terminology (technical terms) and compare its performance with a non-adaptive generic neural MT (NMT) system trained on a large pool of parallel data, and a corpus-based adaptive NMT system as

[1] https://translate.google.com
[2] https://www.bing.com/translator

Pérez-Ortiz, Sánchez-Martínez, Esplà-Gomis, Popović, Rico, Martins, Van den Bogaert, Forcada (eds.)
Proceedings of the 21st Annual Conference of the European Association for Machine Translation, p. 149–158
Alacant, Spain, May 2018.

Src.	Composition and nutritive value of apple products.
GT	Composizione e valore nutritivo dei prodotti apple.
Ref.	Composizione e valore nutritivo dei prodotti a base di mele.
Src.	It also contains system recovery tools you can use to repair broken Windows.
BT	Esso contiene anche gli strumenti di ripristino del sistema possibile utilizzare per riparare le finestre rotte.
Ref.	Esso contiene anche gli strumenti di ripristino del sistema che possibile utilizzare per riparare Windows non funzionante.

Table 1: Examples of incorrectly translating technical terms from English into Italian by online translation engines. Translation queries submitted on 29/03/2018. GT and BT refer to Google Translate and Bing translator, respectively.

a strong (and expensive) term of comparison.

Our results show that, in contrast to the generic and corpus-based adaptive solutions which compromise either the translation quality or the architectural cost, recently proposed instance-based adaptation methods (Farajian et al., 2017b) provide a flexible solution at reasonable costs. This adaptive system is based on a retrieval mechanism that, given a test sentence to be translated, extracts from the pool of parallel data the top (*source*, *target*) pairs in terms of similarity between the *source* and the test sentence. Using this small set of retrieved pairs, it then fine-tunes the model, and applies it to translate the input sentence. As shown in (Farajian et al., 2017b), by applying local adaptation to few training instances, not only the system is able to improve the performance of the generic NMT but, in some domains, it can also outperform strong specialized corpus-based NMT systems trained for several epochs on the corresponding domain-specific data.

In this paper, we further explore the effectiveness of the instance-based adaptation method reporting, in addition to global corpus-level BLEU scores, empirical results on how they perform in translating domain terminology. To this aim, we divide the terms into two categories of single- and multi-word phrases, and study the systems' behaviour in each class separately. Unsurprisingly, in both cases corpus-based adaptation improves the performance of the generic model by a large margin. Such improvements, however, come at the cost of computationally intensive adaptation on all the in-domain data. In contrast, instance-based adaptation achieves comparable results with a less demanding strategy based on adapting the model to few training instances retrieved from the pool of data on the fly. This empirical proof, focused on the proper treatment of domain terms in NMT adaptation, is the main contribution of this paper.

2 Related works

When exposed to new domains (Koehn and Knowles, 2017) or applied in multi-domain scenarios (Farajian et al., 2017a), machine translation systems in general and neural MT in particular, experience performance degradations due to the distance between the target domain and the domain(s) on which they were trained. Previous studies on multi-domain MT provide solutions for this issue, making it possible to cover more than one domain while reducing performance degradations in the target domains (Luong and Manning, 2015; Chen et al., 2016; Zhang et al., 2016; Freitag and Al-Onaizan, 2016; Chu et al., 2017; Farajian et al., 2017b; Kobus et al., 2017). These solutions can be categorized into *static* and *adaptive* approaches. To train one single model using heterogeneous data from many domains, static approaches assume to have simultaneous access to all the training data and their corresponding domain/topic information. This information, which is either manually assigned or automatically inferred from the data, is passed as additional signal to the MT system, helping it to produce higher quality translations for the desired target domain. Existing solutions in the field of NMT propose to incorporate this topic/domain information only on the source side (*i.e.* to support the encoder) (Kobus et al., 2017), only on the target side (*i.e.* to support the decoder) (Chen et al., 2016), or on both sides (Zhang et al., 2016). Although consistent and significant improvements have been reported by this approach, its application to new domains is not trivial, mostly due to the fact that it requires performing expensive NMT and topic model adaptations using the original multi-domain data and the training set for the new domain.

Adaptive approaches, on the other hand, propose to fine tune an existing MT system, trained either on another domain or pool of parallel data, to the new domain or task. While Luong and Manning (2015) report significant improvements

by this approach on the new target domain, Freitag and Al-Onaizan (2016) observe a significant drop in system's performance on the original domain, which is due to the severe overfitting of the model to the new domain. To solve this issue, they propose a slightly different approach, which performs ensemble decoding using both the adapted and the generic model. The *mixed fine tuning* method proposed by Chu et al. (2017) is another approach for keeping under control the performance degradation on the original out-domain data while adapting the model to the new domain. Given the out-domain and in-domain training sets and a model pre-trained only on the out-domain data, this approach continues the training on a parallel corpus that is a mix of the two training corpora, in which the smaller in-domain corpus is oversampled to have the same size as the larger out-domain corpus. The specialized models obtained by these adaptation techniques are empirically shown to be effective, improving the translation quality of a generic NMT system on the target domains. However, the practical adoption of this approach results in developing and maintaining multiple specialized NMT engines (one model per domain), which increases the infrastructure's costs and limits its scalability in real-world application scenarios.

To combine the advantages of the two worlds, (*i.e.* to get close to the high quality of corpus-based adaptation still keeping the scalability of one single model), Farajian et al. (2017b) introduce an instance-based adaptation method for NMT inspired by Hildebrand et al. (2005). Instead of adapting the original generic model to the whole in-domain training corpus, the instance-based method retrieves from the pool of parallel data a small set of sentence pairs in which the source side is similar to the test sentence. Then, it fine-tunes the generic model on-the-fly by using the set of retrieved samples. This makes the instance-based adaptive approach a reasonable solution for real-world production lines, in which the MT system needs to cover a wide range of application domains while keeping under control the architecture's cost.

In addition to the architectural costs of NMT deployment in multi-domain application scenarios, there is another important factor that has to be considered, that is their ability in translating domain-specific words and phrases (*i.e.* terms). Based on their nature, these expressions can be frequently observed in their corresponding domains, being at the same time infrequent or even out of vocabulary (OOV) in the other domains. Nevertheless, data-driven MT systems need to be trained on large amounts of training data, which is generally collected from different sources. This further reduces the relative frequency of these words, making them less probable to be translated correctly by the system. This makes it even more challenging for NMT approach where rare and OOV words are either segmented into their corresponding sub-word units (Sennrich et al., 2016) or mapped to a special "`unk`" token (Luong and Manning, 2015). However, in the relatively recent history of NMT, there are few works that analyze its behavior focusing on domain terminology. Chatterjee et al. (2017) achieve significant improvements with a guide mechanism that helps the NMT decoder to prioritize and adequately handle translation options obtained from terminology lists. Arčan and Buitelaar (2017) empirically show that offline adaptation of a generic NMT system to the new domain improves its performance in translating domain-specific terms. However, they discuss only corpus-based adaptation techniques that, compared to instance-based methods, are less suitable for real-world application. Moreover, they mostly work in a setting in which domain terminology has to be translated in isolation without any context, while in our working scenario we work with full sentences.

3 Neural machine translation adaptation

In this section we first briefly review the state-of-the-art sequence-to-sequence neural machine translation and then describe the two corpus-based and instance-based adaptation approaches.

3.1 Neural machine translation

We build our adaptive systems on top of the state-of-the-art attention-based encoder-decoder neural MT (Bahdanau et al., 2015) in which given the source sentence $x = x_1, ..., x_M$, the translation is modeled as a two-step process. The source sentence x is first encoded into a sequence of hidden states by means of a recurrent neural network (RNN). Then, another RNN decodes the source hidden sequence into the target string. In particular, at each time step the decoder predicts the next target word from the previously generated target word, the last hidden state of the decoder,

and a weighted combination of the encoder hidden states, where the weights are dynamically computed through a feed-forward network, called attention model.

Training of the presented NMT architecture is generally carried out via maximum-likelihood estimation, in which the model parameters such as word embedding matrices, hidden layer units in both the encoder and decoder, and the attention model weights are optimized over a large collection of parallel sentences. In particular, starting from a random initialisation of the parameters, optimization is performed via stochastic gradient descent (Goodfellow et al., 2016), in which at each iteration a randomly selected batch β is extracted from the data and each parameter θ is moved one step in the opposite direction of the mean gradient of the log-likelihood (L), evaluated on the entries of β:

$$\Delta\theta = -\eta \frac{1}{\mid \beta \mid} \sum_{(x,y)\in\beta} \frac{\partial L(x,y)}{\partial \theta} \quad (1)$$

where η is a hyperparameter moderating the size of the step $\Delta\theta$ and is usually referred to as the learning rate. It can either be fixed for all parameters and all iterations, or vary along one or both dimensions (Goodfellow et al., 2016). During training, the optimization is performed by going through several so-called epochs, *i.e.* the number of times the whole training data is processed.

3.2 Corpus-based adaptation in neural MT

Given an existing NMT model, trained either on another domain or on a generic pool of parallel data, corpus-based adaptation methods fine-tune the model parameters by applying the same training procedure described in Section 3.1. Depending on the application scenario, the optimization is performed by iterating over a combination of both the current and new data (Chu et al., 2017) or only the training data of the new domain (Luong and Manning, 2015). The former is usually used when the goal is to adapt the model to the new domain while avoiding performance degradation in the domain on which the model was initially trained. Otherwise, only the training data of the new domain is used. In this paper, we opt for the latter solution because we are interested only in the performance of the system in the new domain.

These solutions, however, require a few hours to fine-tune the system to the target domains, which is scarcely compatible with application scenarios in which users need to instantly start interacting with the MT system. In spite of this (and the inherent cost and scalability-related issues), the competitiveness of this solution motivates its adoption as a strong term of comparison in this paper.

3.3 Instance-based adaptation in neural MT

Instance-based adaptation (Farajian et al., 2017b) is an extension of the aforementioned adaptation method, in which, instead of adapting the model to all the available in-domain training data, only few instances (*i.e.* sentence pairs) are used to tune the model. In particular, given an already existing NMT model, the pool of in-domain parallel data, and a sentence to be translated, it performs the following three steps: 1) retrieve from the pool a set of (*source, target*) pairs in which the *source* is similar to the test sentence; 2) locally adapt the parameters of the model using the retrieved sentence pairs; 3) translate the given test sentence by applying the resulting locally-tuned model. The diagram of the approach is shown in Figure 1. In order to leverage more effectively the information of the retrieved training samples, Farajian et al. (2017b) propose to set the hyperparameters of the training process (*i.e.* learning rate and number of epochs) proportional to the similarity of the retrieved set to the test. This results in fine tuning the model with larger learning rates and for more epochs when the retrieved sentence pairs are highly similar to the test and vice versa.

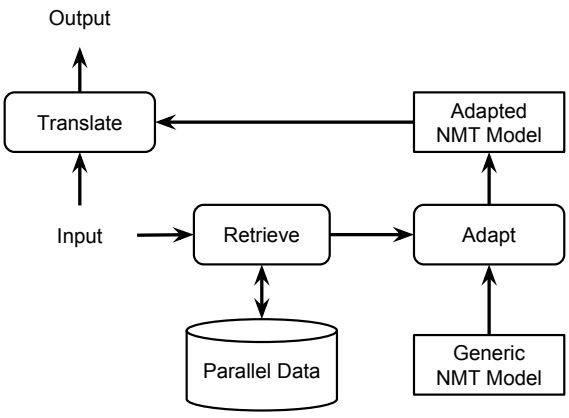

Figure 1: Instance-based NMT adaptation.

4 Experimental setup

4.1 Data

The experiments of this paper are carried out in the English-Italian translation direction. The train-

	Segments	Tokens	Types
Generic	20.8M	373.5M	1.7M
Gnome	76.5K	685.2K	36.0K
KDE4	179.5K	2.1M	75.3K

Table 2: Statistics of the Italian side of the training corpora. Generic data is used for training the generic NMT system, while the domain-specific data (*i.e.* Gnome and KDE4) are used only in the adaptation step.

ing set consists of a large collection of proprietary data collected from several industrial translation projects in different domains (*i.e.* medical, software documentations, user guides, etc.). The statistics of the training corpus are presented in Table 2 (first row).

To evaluate the performance of the systems in translating domain-specific terms we need test sets in which the terms are annotated. Moreover, both adaptive systems need in-domain training data in order to fine tune the generic model to the given domain. This further increases the difficulty of finding the evaluation data. The *BitterCorpus*[3] (Arčan et al., 2014) is a collection of parallel English-Italian documents in the information technology (IT) domain (extracted from Gnome and KDE4 projects) in which technical terms are manually marked and aligned. However, this corpus is not ready to be used in our task *as-is*, since: *i)* it contains only the annotated test data without any in-domain training set, and *ii)* test data are aligned at document level, while in our experiments we need sentence-level aligned corpora.

In order to compile an evaluation package that addresses our needs, we used the publicly available Gnome and KDE4 corpora[4] which are sentence-level aligned, divided them into training and test sets, and then automatically annotated the terminologies in the test by means of the term list extracted from the BitterCorpus[5]. The statistics of the Italian side of the training and test corpora are reported in Table 2 and 3. Some examples of the English terms and their corresponding Italian translations are presented in Table 4. As we see, there are several cases where, in addition to the specific translations used in IT domain (marked with *), the English term can have other translations that are valid in other domains. For example, depending on the domain, the English word *button* can refer to the object used to fasten something (*i.e.* in Italian referred to as *bottone*), or the electrical/electronic equipment that is pressed to turn on or off a device (*i.e.* translated as *pulsante* in Italian). This ambiguity is usually observed in the case of single-word terms, while multi-words often disambiguate each other.

[3] https://hlt-mt.fbk.eu/technologies/bittercorpus

[4] http://opus.lingfil.uu.se/

[5] https://gitlab.com/farajian/TermTraGS

	Seg.	Avg. Len.	Terms		
			single word	multi word	all
Gnome	2000	20.5	2,660	183	2,843
KDE4	2000	25.7	3,767	256	4,023

Table 3: Statistics of the Italian side of the test corpora.

English	Italian
list	lista*, elenco*
path	path*, percorso*, indirizzo*
button	pulsante*, bottone
toolbar	barra degli strumenti
wrapping	a capo automatico*, avvolgere
title bar	barra del titolo
wildcards	caratteri jolly
tree view	vista ad albero
konversation	konversation
mouse pointer	puntatore del mouse
destination folder	cartella di destinazione
regular expression	espressione regolare
right mouse button	tasto destro del mouse

Table 4: Examples of term pairs in our test corpora. Words marked with * represent in-domain translations of the term.

4.2 NMT systems

We conducted the experiments with our in-house developed and maintained branch of the OpenNMT-py toolkit (Klein et al., 2017), which is an implementation of the attention-based encoder-decoder architecture (Bahdanau et al., 2015). Our code is integrated with the open source `ModernMT` project[6], and is highly optimized and already deployed for production systems. In our experiments, we segmented the infrequent words into their corresponding sub-word units by applying the byte pair encoding (BPE) approach (Sennrich et al., 2016). In order to increase the consistency between the source and target segmentations, we learned the BPE merge rules from the concatenation of the source and target side of the training data. We set the number of merge rules to 32K, resulting in vocabularies of size 34K and 35K respectively for English and Italian. We used 2-layered LSTMs in both the encoder and decoder, each of which containing 500 hidden units. We

[6] www.modernmt.eu

set the word embedding size to 500 for both the source and target languages. The parameters are optimized with SGD using the initial learning rate of 1.0 with a decaying factor of 0.9. The batch size is set to 64, and the model is evaluated after each epoch. We trained the system for 11 epochs and selected the model with the highest BLEU score on our development set.

Our reimplementation of the instance-based adaptive system uses the open source Lucene library (McCandless et al., 2010) to store the training samples (*i.e.* pool of the generic and domain-specific data). Similar to (Farajian et al., 2017b), given the test sentence it retrieves the most similar instance from the pool (*i.e.* top-1) and adapts the aforementioned generic model accordingly. It sets the hyperparameters of the fine-tuning process proportional to the similarity of the retrieved instance and the test sentence. For example, it fine-tunes the model with the learning rate of 0.2 for 10 iterations if the similarity of the retrieved instance to the test is 1.0. In this work we used sentence-level BLEU (Chen and Cherry, 2014) as the similarity measure. In our experiments, the average time for updating the model was about 0.5 seconds per sentence.

The corpus-based adapted NMT systems are multiple instances of the generic system each of which trained on the corresponding domain-specific training data for several epochs, until no improvement is observed in the model perplexity on our development set for four consecutive epochs. We then used, for each domain, the model with minimum perplexity on the development set (*i.e.* model obtained after 26 and 4 epochs respectively for Gnome and KDE4). We used the same settings as the generic system for training these systems. However, for fine tuning we started with a learning rate of 0.5. In our experiments, the corpus-based adaptation of the model took about 3:00 and 1:15 hours for Gnome and KDE4 domains, respectively.[7]

4.3 Evaluation metrics

We evaluate the systems' performance both in terms of BLEU (Papineni et al., 2002) and *term hit rate* (THR). While the former measures the overall quality of the translations with respect to the manually-translated reference, the latter analyzes the ability of the system in learning the vocabulary of each specific domain. To this aim it computes the proportion of terms in the reference that are correctly translated by the MT system. However, in order to avoid assigning higher scores to the systems which over-generate the same term, it clips the counts of the matched terms by their frequency in the reference (2).

$$THR = \frac{\sum_{term \in ref} count_{clip}(term)}{\sum_{term \in ref} count_{ref}(term)} \quad (2)$$

Since there are two types of single-word and multi-word terms in our test sets, in order to have a more detailed analysis of systems' behaviour we report the scores for each class separately in addition to the overall THR score.

5 Analysis and discussion

In this section we present a detailed analysis of the results obtained by different systems and compare the systems in translating the technical terms in Gnome and KDE4.

5.1 Translation quality

Table 5 reports the performance of the generic, instance-based, and corpus-based adaptive systems on Gnome and KDE4 test sets in terms of BLEU. As the results show (first two rows), the instance-based system significantly improves the performance of the generic system by +7.80 and +6.55 BLEU points. However, it obtains a lower BLEU score compared to its corpus-based counterparts. In our investigations, we noticed that in almost all the cases where the application domain is new (*i.e.* the training data of the target domain was not included in the pool of data used for training the generic system), the corpus-based adapted system produces translations of higher quality compared to the instance-based system. Nevertheless, this comes at the cost of training the system for several hours, instead of instantly starting the translation process.

Another interesting phenomenon that we observed in these experiments is the correlation of the performance gain and the average similarity of the retrieved samples to the test sentences. We noticed a larger performance gain in the case of Gnome compared to KDE4 (+7.80 vs. +6.55) while the average similarity of the retrieved sentence pairs in this domain is lower (0.36 compared to 0.56). Comparing the ratio of the successful retrievals in

[7]We carried out the experiments on Azure instances with NVIDIA Tesla k80 GPUs.

	Avg. Sim.	Generic	Instance based	Corpus based
Gnome	0.36	35.97	43.77	49.79
KDE4	0.56	35.09	41.64	46.26
Gnome †	0.43	38.06	51.36	56.00
KDE4 †	0.61	36.99	51.84	48.95

Table 5: BLEU score of the generic and adaptive NMTs on the test sets. The corpora marked with †are subsets of the original corpora for which a similar instance is retrieved.

	Term Type	Generic	Instance based	Corpus based
Gnome	Single-word	79.58	82.16	86.55
	Multi-word	62.79	70.54	80.62
	Overall	78.59	81.48	86.20
KDE4	Single-word	73.70	79.48	81.94
	Multi-word	48.15	58.52	61.48
	Overall	72.24	78.28	80.78

Table 6: Performance of the generic and adaptive NMTs on the test sets, in terms of THR.

	Term Type	English	Italian
Gnome	Single-word	70.1	62.0
	Multi-word	60.0	51.6
	Overall	69.7	61.4
KDE4	Single-word	71.7	59.5
	Multi-word	68.2	45.5
	Overall	71.5	58.7

Table 7: Percentage of the retrieved samples that contain the desired terms.

the two systems partially explains this behaviour: in the case of Gnome, in 83.9% of the cases the system is able to find training samples similar to the test while in KDE4 this figure decreases to 75.8%. Moreover, by limiting our analysis to these cases, *i.e.* sentences for which the system has successfully retrieved a similar instance (last two rows in Table 5), we see a correlation higher than 0.9 between the performance gain and the similarity. Even more surprisingly, we observe that on this subset of KDE4 corpus the instance-based system outperforms its corpus-based counterpart. This is mostly due to the fact that retrieved instances in this case are highly similar to the test sentences.

5.2 Term translation

Table 6 presents the performance of the systems on both Gnome and KDE4 data. Since a large portion of the generic training data belongs to the IT domain we observe a reasonably high performance by the generic system in the studied domains, in particular on the single-word terms (79.58 and 73.70 on Gnome and KDE4 domains, respectively). However, translating multi-word terms is more challenging for all the systems as it involves producing sequences of words that might have several translations in different context. For example the English words *bar*, *path*, and *mouse* are usually translated into *bar*, *indirizzo*, and *topo* while their contextual translations in the technical terms *title bar*, *full path* and *mouse pointer* is *barra*, *path*, and *mouse*. This makes the translation more difficult for the systems, resulting in a significant performance drop compared to the case of single-word terms.

5.3 Instance selection effect

In addition to the similarity of the retrieved samples to the test discussed in (Farajian et al., 2017b), the presence of domain terms in the retrieved sentence pairs is another important factor for instance-based adaptation. As Table 7 shows, in about 30% of the cases the retrieved English sentence does not contain the desired term. This proportion is even higher if we look at the target side of the retrieved instances, in which around 40% of the desired term translations are missing. However, this is expected since the retrieval is performed only based on the source side information (*i.e.* in our experiments English), with no additional filters based on the target side of the retrieved instance. Measuring the performance of the adaptive system in correcting the terms which are missed by the generic system shows that the instance-based system effectively learns the vocabulary of the application domain, correcting up to 76.64% of the mistakes made by the generic system if the desired term translation exists in the retrieved instance (Table 8).

6 Further analysis

In addition to the automatic evaluations we performed further manual analysis on the outputs of the instance-based adaptive system. The results of this analysis indicate that, compared to the generic system, its behavior differs in two main aspects: *i)* learning to translate the terms that are missed or wrongly translated by the generic system, *ii)* adapting to different style of the translation. When run on new domains, for which it has not seen any

	Single-word	Multi-word	Overall
Gnome	64.33	52.94	63.22
KDE4	76.92	73.91	76.64

Table 8: Percentage of the terms corrected by the instance-based adaptive system.

in-domain training data, it is highly probable that the generic system receives translation requests containing terms which are OOV or infrequently observed in the training data. In such cases, even after applying BPE, it might not be able to produce proper translations. As an example, the English word *dolphin*, which is rarely observed in the generic training data, is always translated in the Italian word *delfino* which refers to the animal. However, in the KDE4 domain it corresponds to a proper noun that indicates a file manager application. As we see in Table 9, the generic system wrongly translates it into *delphin*. By accessing in-domain training data (*i.e.* either the full corpus or just one single, highly similar instance), both the adaptive systems are able to correctly translate it.

The English terms *Control Center* and *mouse cursor* are two interesting examples of learning domain-specific translation styles. While in the generic training data these terms are usually translated into *Control Center* and *cursore del mouse*, in the KDE4 domain the human translators prefer them to be translated into *centro di controllo* and *puntatore del mouse*. As we see in the examples of Table 9, the generic system produces their commonly used translations, while the instance-based system is able to learn and produce the desired domain-specific translations.

We also observed a few cases in which the instance-based approach learns to properly generate Italian terms in the translation while there is no corresponding source English term in the given test sentence. The Italian word *pulsante* in the fourth example provided in Table 9 is one of these cases. As we see, the input English sentence does not contain the word *button*, hence both the generic and corpus-based adapted NMT systems do not produce any translation for it. On the contrary, the instance-based system, being trained on a very similar instance which contains the word *pulsante*, learns the pattern and produces a translation that is closer to the reference.

Finally, we noticed that inconsistent translations of the terms can affect the instance-based adaptive system, resulting in translations which are different than the manually produced references. The last example provided in Table 9 shows one of these cases. As we see, the English term *packages* can be translated into either *pacchetti* or *package*. So, based on the suggestion provided by the retrieval module, the instance-based system learns to translate it into *package* which is another valid translation of this term. This, however, does not affect the global performance of the system due to the small amount of similar situations.

7 Conclusions

We investigated the application of instance-based adaptive NMT in a real-world scenario where translation requests come from new domains that contain many technical terms. In particular, we analyzed its ability to properly handle domain terminology, comparing its output against the translations produced by a generic (unadapted) NMT system and a corpus-based specialized NMT system. Overall, our experiments with Gnome and KDE4 data reveal that the two adaptation methods significantly improve the performance of the generic system both in terms of global BLEU score and *term translation accuracy*. Unsurprisingly, by performing a computationally intensive fine tuning on the full in-domain training data, corpus-based adaptation produces specialized NMT systems that achieve better results at the cost of reduced scalability. However, the less demanding instance-based adaptation (performed on one parallel sentence pair retrieved from a pool of data based on its similarity to the test sentence), is able to effectively learn domain terms' translations, even for expressions that were never observed by the generic model. Such capability allows instance-based adaptation to significantly reduce the gap between generic and corpus-based specialized NMT models at manageable costs.

Acknowledgements

This work has been partially supported by the EC-funded H2020 projects QT21 (grant no. 645452) and ModernMT (grant no. 645487). This work was also supported by The Alan Turing Institute under the EPSRC grant EP/N510129/1 and by a donation of Azure credits by Microsoft.

References

[Arčan and Buitelaar2017] Arčan, Mihael and Paul Buitelaar. 2017. Translating domain-specific expressions in knowledge bases with neural machine translation. CoRR. http://arxiv.org/abs/1709.02184.

[Arčan et al.2014] Arčan, Mihael, Marco Turchi, Sara Tonelli, and Paul Buitelaar. 2014. Enhancing statistical machine translation with bilingual terminol-

Source	[...]Files may be dragged and dropped onto & kwrite; from the Desktop, the filemanager & dolphin;[...]
Reference	[...]I file possono essere trascinati e rilasciati su & kwrite; dal Desktop, & dolphin;[...]
Ret. Src.	[...]Files may be dragged and dropped onto & kate; from the Desktop, the filemanager & dolphin;[...]
Ret. Trg.	[...]I file possono essere trascinati e rilasciati in & kate; dal Desktop, dal gestore di file & dolphin;[...]
Generic	[...]I file possono essere trascinati e rilasciati su & kwrite; dal Desktop, dal file manager e dal delphin;[...]
Instance-based	[...]I file possono essere trascinati e rilasciati in & kwrite; dal Desktop, dal gestore di file & dolphin;[...]
Corpus-based	[...]I file possono essere trascinati e caduti su kwrite; dal desktop, dal gestore file & dolphin;[...]
Source	[...]The mouse cursor is identified in the status bar.[...]
Reference	[...]Al puntatore del mouse è identificato nella barra di stato.[...]
Ret. Src.	[...]When you hold the mouse cursor still for a moment[...]
Ret. Trg.	[...]Mantenendo fermo per qualche istante il puntatore del mouse[...]
Generic	[...]Il cursore del mouse viene identificato nella barra di stato.[...]
Instance-based	[...]Il puntatore del mouse viene identificato nella barra di stato.[...]
Corpus-based	[...]Il cursore del mouse è identificato nella barra di stato.[...]
Source	Exiting the kde Control Center
Reference	Uscire dal centro di controllo di kde
Ret. Src.	The kde Control Center Screen
ret. Trg.	Lo schermo del centro di controllo di kde;
Generic	Uscita da kde Control Center
Instance-based	Uscita dal centro di controllo di kde
Corpus-based	Uscita dal centro di controllo di kde
Source	This saves the settings and closes the configuration dialog.
Reference	Questo pulsante salva le impostazioni e chiude la finestra di configurazione.
Ret. Src.	This saves the settings without closing the configuration dialog.
Ret. Trg.	Questo pulsante salva le impostazioni senza chiudere la finestra di configurazione.
Generic	~~pulsante~~ Salva le impostazioni e chiude la finestra di configurazione.
Instance-based	Questo pulsante salva le impostazioni e chiude la finestra di configurazione.
Corpus-based	Questo ~~pulsante~~ salva le impostazioni e chiude la finestra di configurazione.
Source	Automatically scan project's packages
Reference	Analizzare automaticamente i pacchetti del progetto
Ret. Src.	View or modify the UML system's packages.
Ret. Trg.	Consente di visualizzare o modificare i package di sistema UML.
Generic	Scansione automatica dei pacchetti del progetto
Instance-based	Scansione automatica dei package di progetto
Corpus-based	Scansiona automaticamente i pacchetti del progetto

Table 9: Translation examples produced by generic and adaptive NMT systems.

ogy in a CAT environment. In Proc. of AMTA'14, Vancouver, BC, Canada, October.

[Bahdanau et al.2015] Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In Proc. of ICLR'15, San Diego, CA, USA, May.

[Chatterjee et al.2017] Chatterjee, Rajen, Matteo Negri, Marco Turchi, Marcello Federico, Lucia Specia, and Frederic Blain. 2017. Guiding neural machine translation decoding with external knowledge. In Proc. of WMT'17, pages 157–168, Copenhagen, Denmark, September.

[Chen and Cherry2014] Chen, Boxing and Colin Cherry. 2014. A systematic comparison of smoothing techniques for sentence-level BLEU. In Proc. of WMT'14, pages 362–367, Baltimore, Maryland, USA, June.

[Chen et al.2016] Chen, Wenhu, Evgeny Matusov, Shahram Khadivi, and Jan-Thorsten Peter. 2016. Guided Alignment Training for Topic-Aware Neural Machine Translation. In Proc. of AMTA'16, pages 121–134, Austin, Texas, October.

[Chu et al.2017] Chu, Chenhui, Raj Dabre, and Sadao Kurohashi. 2017. An empirical comparison of simple domain adaptation methods for neural machine translation. In Proc. of ACL'17-Short Papers, pages 385–391, Vancouver, Canada, July, August.

[Farajian et al.2017a] Farajian, M. Amin, Marco Turchi, Matteo Negri, Nicola Bertoldi, and Marcello Federico. 2017a. Neural vs. phrase-based machine translation in a multi-domain scenario. In Proc. of EACL'17, pages 280–284, Valencia, Spain, April.

[Farajian et al.2017b] Farajian, M. Amin, Marco Turchi, Matteo Negri, and Marcello Federico. 2017b. Multi-domain neural machine translation through unsupervised adaptation. In Proc. of WMT'17, pages 127–137, Copenhagen, Denmark, September.

[Freitag and Al-Onaizan2016] Freitag, Markus and Yaser Al-Onaizan. 2016. Fast domain adaptation for neural machine translation. CoRR. https://arxiv.org/abs/1612.06897.

[Goodfellow et al.2016] Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. 2016. Deep Learning. MIT Press.

[Hildebrand et al.2005] Hildebrand, Almut Silja, Mattias Eck, Stephan Vogel, and Alex Waibel. 2005. Adaptation of the translation model for statistical machine translation based on information retrieval. In Proc. of EAMT'05, pages 133–142, Budapest, Hungary, May.

[Kingscott2002] Kingscott, Geoffrey. 2002. Technical translation and related disciplines. Perspectives, 10(4):247–255.

[Klein et al.2017] Klein, Guillaume, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. In Proc. of ACL'17, pages 67–72, Vancouver, Canada, July, August.

[Kobus et al.2017] Kobus, Catherine, Josep Maria Crego, and Jean Senellart. 2017. Domain Control for Neural Machine Translation. In Proc. of RANLP'17, pages 372–378, Varna, Bulgaria, September.

[Koehn and Knowles2017] Koehn, Philipp and Rebecca Knowles. 2017. Six challenges for neural machine translation. In Proc. of 1^{st} Workshop on Neural Machine Translation, pages 28–39, Vancouver, Canada, August.

[Luong and Manning2015] Luong, Minh-Thang and Christopher D Manning. 2015. Stanford Neural Machine Translation Systems for Spoken Language Domains. In Proc. of IWSLT'15, pages 76–79, Da Nang, Vietnam, December.

[McCandless et al.2010] McCandless, Michael, Erik Hatcher, and Otis Gospodnetic. 2010. Lucene in Action. Manning Publications Co., Greenwich, CT, USA.

[Papineni et al.2002] Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In Proc. of ACL'02, pages 311–318, Philadelphia, USA, July.

[Sennrich et al.2016] Sennrich, Rico, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In Proc. of ACL'16, pages 1715–1725, Berlin, Germany, August.

[Zhang et al.2016] Zhang, Jian, Liangyou Li, Andy Way, and Qun Liu. 2016. Topic-informed neural machine translation. In Proc. of COLING'16, pages 1807–1817, Osaka, Japan, December.