Online Automatic Post-editing for MT in a Multi-Domain Translation Environment

Rajen Chatterjee[1,2], **Gebremedhen Gebremelak**[1,2], **Matteo Negri**[2], **Marco Turchi**[2]
[1]University of Trento
[2]Fondazione Bruno Kessler
{chatterjee,gebremelak,negri,turchi}@fbk.eu

Abstract

Automatic post-editing (APE) for machine translation (MT) aims to fix recurrent errors made by the MT decoder by learning from correction examples. In controlled evaluation scenarios, the representativeness of the training set with respect to the test data is a key factor to achieve good performance. Real-life scenarios, however, do not guarantee such favorable learning conditions. Ideally, to be integrated in a real professional translation workflow (e.g. to play a role in computer-assisted translation framework), APE tools should be flexible enough to cope with continuous streams of diverse data coming from different domains/genres. To cope with this problem, we propose an online APE framework that is: *i)* robust to data diversity (i.e. capable to learn and apply correction rules in the right contexts) and *ii)* able to evolve over time (by continuously extending and refining its knowledge). In a comparative evaluation, with English-German test data coming in random order from two different domains, we show the effectiveness of our approach, which outperforms a strong batch system and the state of the art in online APE.

1 Introduction

Automatic post-editing (APE) systems for machine translation (MT) aim to correct the errors present in a machine-translated text before showing it to the user, thereby reducing human workload and eventually increase translation productivity. The choice of having such post-processing systems is well motivated in (Bojar et al., 2015) and becomes a must when the MT engine used to translate is not directly accessible for retraining or for more radical internal modifications (e.g. when working with a third party MT system). As pointed out by (Parton et al., 2012; Chatterjee et al., 2015b), from the application point of view an APE system can help to: *i)* Improve MT output by exploiting information unavailable to the decoder, or by performing deeper text analysis that is too expensive at the decoding stage; *ii)* Provide professional translators with improved MT output quality to reduce (human) post-editing effort and *iii)* Adapt the output of a general-purpose MT system to the lexicon/style requested in a specific application domain. Similar to what is usually done in MT, APE components learn post-editing rules from "parallel" corpora consisting of machine-translated text (*mt*, optionally with its corresponding source text – *src*) and its post-edits (*pe*) provided by human post-editors. The effectiveness of learning from relatively small amounts of post-edited data is evident from the impressive outcomes of the recently held APE shared task at WMT 2016 (Bojar et al., 2016). Different APE paradigms, like neural (Junczys-Dowmunt and Grundkiewicz, 2016), hybrid (Chatterjee et al., 2016), and phrase-based (Pal et al., 2016) were all able to significantly improve MT output quality in the IT domain, with gains ranging from 2.0 to 5.5 BLEU points. Nevertheless, this success and the positive outcomes of previous work on automatic MT error correction build on a problem formulation that assumes to operate in a controlled lab environment, where the systems are trained and evaluated across a coherent/homogeneous data set. Moving from this controlled scenario to real-world translation workflows, 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 inherently more challenging, because the APE systems have to adapt to

Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers, pages 525–535,
Valencia, Spain, April 3-7, 2017.

all these diversities in real-time. In addition to the problem that training data provide a fraction of the possible error correction examples (a normal issue when learning from finite, often small training data), the additional complexity derives from two concurrent factors. First, not all the learned error correction rules are universally applicable: applying them in the wrong context can damage the MT output instead of improving it. Second, once in production, the APE system should be able to process streams of diverse input data presented in random order: promptly reacting to such variability is hence crucial. We define this more complex and realistic 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 our knowledge, the evaluation of APE systems on MDTE data in real-time/online translation scenarios is still unexplored. This paper represents a first step along this direction: although a full-fledged evaluation centered on human translation in a computer-assisted translation (CAT) framework is out of our reach, we provide a proof of concept in which we simulate the MDTE scenario by running different APE solutions on a stream of data coming from two different domains. By analysing alternative solutions, we discuss the limitations not only of batch APE methods (insensitive to domain shifts), but also of state-of-the-art online translation systems evaluated in the APE task in MDTE conditions. Thot (Ortiz-Martınez and Casacuberta, 2014), the online system used as term of comparison, shows in fact the inability to discern which of the learned correction rules is suitable for a specific context. In practice, all rules are created equal, for any given domain.

To overcome this limitation, we proceed incrementally. First, we propose an approach based on an instance selection strategy, which learns local, sentence-specific APE models from small amounts of relevant data for each translation to be post-edited. Then, on top of it, we add an improved way to estimate the parameters of the sentence-specific APE models. To this aim, we exploit a dynamic knowledge base that keeps track of global statistics computed over all the previously seen data (i.e. it does not rely only on those computed from the selected instances). Finally, the dynamic knowledge base gives us the possibility to experiment with new features in addition to those used by current APE systems based on the phrase-based MT paradigm. Such features incorporate in the translation models also the negative feedback collected from human post-editors. Instead of continuously expanding our knowledge base of correction rules (i.e. only considering the positive feedback about how to correct a given error), we also stepwise refine it by weighing the acquired correction rules according to their reliability (e.g. demoting those that led to corrections eventually modified by the human). Positive evaluation results reflect this incremental approach. To summarize, our contribution is a fully automated online APE system that does not rely on pre-trained models or tuned weights (unlike Thot that needs to be pre-trained and tuned) and incorporates for the first time both positive and negative post-editors' feedback to set the state-of-the-art in the difficult task of APE in the MDTE scenario.

2 Related work

Most of the previous works on APE cast the problem as a phrase-based statistical MT task[1] and operate in a batch framework where systems are evaluated on static test sets that are homogeneous with the training data (Simard et al., 2007; Dugast et al., 2007; Terumasa, 2007; Pilevar, 2011; Béchara et al., 2011; Chatterjee et al., 2016). These systems, however, are not able to leverage the feedback of the post-editors in an online translation scenario. The capability to evolve by learning from human feedback has been addressed by several online translation systems but mainly focusing on the MT task (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). From these several online MT systems, we discuss the two that have been used also for the APE task.

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 an MT output sequence. To perform post-edit propagation, the system is trained incrementally us-

[1]Only recently, the wave of neural models has also reached the APE task (Junczys-Dowmunt and Grundkiewicz, 2016), setting the new state of the art at WMT (Bojar et al., 2016). The problem addressed in this paper (dealing with MDTE data), as well as the proposed online solution are still too computationally intensive to experiment with neural models. We hence leave this aspect as a future work direction.

ing pairs of machine-translated *(mt)* and human post-edited *(pe)* segments as they were produced. When receiving a new *(mt, pe)* pair, word alignments are obtained 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 and, for every new document, the APE system is initialised with an "empty" model. This represents a possible limitation of the approach: although document-specific correction rules show a relatively high precision, some of them might in fact be useful also in other contexts and should be retained. Our approach avoids this limitation by maintaining a global knowledge base to store all the processed documents, still being able to retrieve post-editing rules specific to a document to be translated.[2]

Thot. The Thot toolkit (Ortiz-Martınez and Casacuberta, 2014) is developed to support automatic and interactive statistical machine translation.[3] It was also successfully used by Lagarda et al. (2015) to experiment in an online APE setting with several data sets for multiple language pairs, with base MT systems built using different technologies (rule-based MT, statistical MT). In order to incorporate user feedback in the underlying translation and language models, the systems maintains and incrementally updates all the required statistics. For the language model, it simply updates n-gram counts. In the case of the translation model, the process exploits an incremental version of expectation maximization algorithm to obtain word alignments and extract the phrase pairs whose counts are continuously updated. Other features, like source/target phrase-length models or the distortion model, are extracted considering geometric distributions with fixed parameters. 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. This makes our online APE approach independent from any pre-trained model or pre-tuned feature weights. Moreover, while in Thot the correction rules are learned in real-time from all the data processed, our system only learns from the most relevant data samples. Nevertheless, considering Thot as the state-of-the-art in online APE, we will use it as a term of comparison in our experiments.

[2]In our experiments we do not compare against PEPr since, being designed for document-level translation it is unable to operate in the MDTE scenario.

[3]https://github.com/daormar/thot

3 Online APE system

The backbone of our online APE system is similar to the state-of-the-art statistical batch APE approach proposed in (Chatterjee et al., 2015b). The system is trained on (*src*, *mt*, *pe*) triplets, and learns correction rules in the form of (*mt#src*, *pe*) pairs. The first element of each pair consists of MT phrases (single or multiple words) that are associated to their corresponding source words by using a word alignment model. This "joint representation" helps to restrict the applicability of each rule to the appropriate context, and was shown to perform better than using only the *mt* words as the left-hand side of the rules (Béchara et al., 2011). Our migration to the online scenario builds on incrementally extending this backbone with an instance selection mechanism (§3.1), a dynamic knowledge base (§3.2) and new features (§3.3).

3.1 Instance selection

Current batch and online APE methods estimate parameters of the models over all the available training data. This strategy may not be effective in the MDTE scenario, since the model will tend to become more and more generic by incorporating knowledge from several domains. In the long-run, this can complicate the selection of domain-specific correction rules suitable for a particular MT segment. One of the possible solutions is to constrain the system to work at document level as proposed by Simard and Foster (2013). In their approach, however, the models are reset back to their original state once the entire document is processed, due to which knowledge gained from the current document is lost. Our instance selection technique aims to overcome this issue, allowing the system to preserve all the knowledge acquired during the online learning process, still being able to apply specific post-editing rules when needed.

The instance selection mechanism consists in retrieving *ad-hoc* training sentence pairs for each MT output to be post-edited. In practice, the creation of the APE model and the estimation of its parameters are performed on-the-fly by processing relevant instances retrieved from the previously processed data. In the MDTE scenario, this will come from heterogeneous domains. The rele-

vance of a training sample is measured in terms of a similarity score based on term frequency-inverse document frequency (tf-idf[4]) computed using the Lucene software library.[5] Indexing and retrieving training triplets (*src, mt, and pe*) in this mechanism is fast, which makes it perfectly suitable to use in an online learning scenario. The cut-off similarity score is empirically estimated over a held-out development set. Input segments not having training samples above the threshold are left untouched to avoid any possible damage resulting from the application of unreliable correction rules learned from unrelated contexts. This is in contrast with the strategy adopted by current APE systems, which tend to always "translate" the given input segment, independently from the reliability of the applicable correction rules.

Once the training samples are selected for an input segment, several models are built on-the-fly. A tri-gram local language model (LM) is built over the target side of the training corpus with the IRSTLM toolkit (Federico et al., 2008). Along with the local LM a tri-gram global LM is also used, which is updated whenever a human post-edition (*pe*) is received. Local translation and reordering models are built with the Moses toolkit (Koehn et al., 2007), computing word alignment for each sentence pair using the incremental GIZA++ software.[6]

The feature weights of the log-linear model are optimized over a subset of the selected instances. The size of this development set is critical: if it is too large, then 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 (Crammer and Singer, 2003) are run in parallel on multiple development sets (Tange, 2011). For this purpose, the retrieved samples are randomly split three times into training and development. The tuned weights resulting from the three development runs are then averaged and used to decode the input MT segment.

To summarize, our training/tuning scheme requires a minimum number of retrieved sentence pairs. Following an 80%-20% distribution over training and development data, and setting to 5 the minimum number of samples needed for tuning, the complete process requires the retrieval of at least 25 samples. If this number is not reached, all the retrieved samples are used for training, the optimization step is skipped and the previously computed weights are used. If no sample is selected, then the MT output will be left untouched.

3.2 Dynamic knowledge base

The APE system described so far is built by considering only the most similar retrieved sentences, which we hypothesize to be the most useful to learn reliable corrections for a given MT output. On one hand, this strategy avoids to end up with correction options that are not appropriate to post-edit the MT output. On the other hand, it computes the statistics of the models (i.e. translation and lexicalized reordering probabilities) using only few parallel sentences, resulting in potentially unreliable values that can penalise the work of the decoder. To address this issue, we complement instance selection with a dynamic knowledge base able to keep track of all the previous observations relevant for post-editing. Such a component provides the decoder with all the translation options extracted from the retrieved sentences but, instead of computing the probabilities only on these segments, it takes advantage of all the occurrences of a translation option in the previously processed sentences. This allows our online APE system to use only the most useful translation options, associated with more reliable statistics.

The dynamic knowledge base is implemented by a distributed, scalable and real-time inverted index that, after insertion, makes all data immediately available for search and update. The ElasticSearch[7] engine is used for this purpose. Once the post-edit is made available to our system, the word alignment between the *mt* and the *pe* is computed, the sentence pair is split in phrases and then added to the dynamic model. If a translation option is already present, then the phrase translation and the orientation counts are updated, otherwise it is inserted for the first time. This is run in multi-threading, by also managing possible conflicts (i.e. the access to the same translation option by different threads). Word lexical information and phrase

[4] In MT, tf-idf was previously used by Hildebrand et al. (2005) to create a pseudo in-domain corpus from a large out-of-domain corpus. Our work is the first to investigate it for the APE task in an online learning scenario.

[5] https://lucene.apache.org/

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

[7] http://www.elastic.co/products/elasticsearch

counts are stored apart. At decoding time, the IDs of the samples retrieved by instance selection and the *mt* sentences are used to query the dynamic knowledge base. The translation options that satisfy the query are retrieved and supplied to the decoder in the form of translation and reordering model information. All the feature scores (four for the translation model and six for the reordering model) are computed on-the-fly.

Compared to the suffix arrays used to implement MT dynamic models (Germann, 2014; Denkowski et al., 2014), in which the whole sentence pairs are stored, our technique needs to save more information (all the translation options) but: *i)* the amount of data in APE is much less that in MT so it can be easily managed by *ad hoc* solutions, and *ii)* it allows us to collect global information at translation option level that can result in useful additional features for the model. This last aspect is explored in the next section, in which the reliability of the translation options is measured by looking at the behavior of the post-editors.

3.3 Negative feedback in-the-loop

Similar to the APE systems mentioned in Section 2, the one described so far stores only post-editors' positive feedback. Its knowledge base of correction rules and the statistics to estimate the model parameters are in fact continuously updated only based on alignment information between (*mt, pe*) pairs. Post-edits, however, can also be used to infer negative feedback and use it to penalize unreliable correction options (i.e. those resulting in post-edits eventually modified by the human). The dynamic knowledge base allows us to easily integrate this kind of information, in the form of two additional "negative feedback" features:

- F1. This feature penalizes the correction rules that are selected by the decoder but eventually modified by the post-editor. This can be due to the application of a rule in the wrong context (e.g. in case of domain changes in the input stream of data) but, most likely, to the fact that the learned rule is wrong (e.g. as the result of ambiguous/wrong word alignment). It is computed as the ratio of the number of times the post-editors modified a correction made by the APE decoder to the total number of times the decoder has made the correction. The information about which correction rules have been applied by the APE system is obtained from the Moses decoder `trace` option. The information about which of them has been modified is derived by string matching the target side of the rule in the final human post-edit.

- F2. This rule penalizes the correction rules that, even if not used, were available to the decoder (i.e. translation options discarded during decoding). Assuming that the application of these options would have been eventually corrected by the post-editor, all the rules in the phrase table are scanned to check if their target side (i.e. the correction) is present in the final human post-edit (again by string matching). If not, then the corresponding rule is penalised. This feature is computed as the ratio of the number of times the correction in the phrase table is (assumed to be) modified by the post-editor to the total number of time the correction rule has been seen in the local phrase table for all the segments processed so far. Since the decoder operates with a segment-specific local phrase table containing only the options relevant to the segment to be post-edited, computing this feature is not expensive.

We also evaluate system performance by using the two features together. As we will see in Section 5, although our use of negative feedback is still at a preliminary stage, its integration in our online APE framework yields some improvements.

4 Evaluation setting

Data. We experiment with two English-German data sets: *i)* the data released for the APE shared task organised within the first Conference on Machine Translation (WMT16) (Bojar et al., 2016), and *ii)* the data used in (Chatterjee et al., 2015b), which is a subset of the Autodesk Post-Editing Data corpus.[8] Although they come from the same category (IT), they feature variability in terms of vocabulary, MT engines used for translation, MT errors and post-editing style. According to our broad notion of "domain", these variations contribute to make the two data sets different enough to emulate an MDTE scenario for testing online APE capabilities. The data are pre-processed to obtain a joint representation that

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

	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
WMT16	210,573	214,720	32211	16,388	17.54	17.89	4.907	26.22

Table 1: Data statistics

links each MT word with its corresponding source word/s (*mt#src*). This representation, proposed by Béchara et al. (2011), leverages the source information to disambiguate post-editing rules and foster their application only in appropriate contexts (the matching condition is defined both on the source and on the target language). The joint representation is used as a source corpus to train all the APE systems compared in this paper and it is obtained by concatenating words in the source (*src*) and in the MT (*mt*) segments after aligning them with MGIZA++ (Gao and Vogel, 2008).

The Autodesk training and development sets consist of 12,238, and 1,948 segments respectively, while the WMT16 data contains 12,000, and 1,000 segments. Table 1 provides additional statistics of the source (*mt#src*) and target (*pe*) training sets, 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), as an indicator of the original translation quality. Looking at these statistics, there are several indicators that suggest that the WMT16 corpus provides a more difficult scenario for APE than the Autodesk one. First, the WMT16 corpus has on average longer sentences, which generally increases the complexity of the rule extraction and decoding processes. Second, although the two data sets have a similar repetition rate, the WMT16 has more tokens indicating the higher sparsity of the data. Finally, the lower TER of WMT16 suggests that there are less corrections to perform and, in turn, a higher chance to deteriorate the original MT output if the learned rules are applied in the wrong context.

To conclude, we measure the diversity of the two data sets 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 case ($\sim$15%) indicating that the information to share between the two data sets is minimal.

In our MDTE experiments, the training data is first merged, then shuffled and then split in two halves of 12,119 segments. The same procedure is applied to the development sets.

Evaluation metrics. The performance of the different systems is evaluated in terms of 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 output and the corresponding references (in this case human post-edits) by looking at the word/n-gram overlaps. Precision is the ratio of the number of sentences an APE system improves (with respect to the MT output) over all the sentences it modifies.[9] Higher precision indicates that the APE system is able to improve the quality of most of the sentences it changed. The statistical significance of BLEU results is computed using paired bootstrap resampling (Koehn, 2004). For TER, we use stratified approximate randomization (Clark et al., 2011).

Terms of comparison. We evaluate our online learning approach against: *i)* the MT baseline (i.e. the MT output left untouched), *ii)* the batch APE system described in Section 3, on top of which we incrementally add our online learning extensions, and *iii)* the Thot toolkit.

5 Experiments and results

The batch APE system is trained on the first half of the shuffled training set, tuned with the development set (2,948 segments), and evaluated over the second half of the training data. Since the batch APE only learns from the training set, we expect its performance to give us a lower bound estimate, which should be outperformed by the online APE systems that can learn from the test data too. To run the online experiments with Thot, the system first needs to estimate the feature weights of the log-linear model on a develop-

[9] For each sentence in the test set, if the TER score of the APE system is different from the baseline then it is considered as a modified sentence.

ment set. For this purpose, it is trained and tuned off-line like a batch APE system. Three online extensions of the batch backbone architecture described in Section 3 are evaluated. These are: *i)* the instance selection system (IS); *ii)* the dynamic knowledge base system (IS+DynKB) and *iii)* the dynamic knowledge base system enhanced with the negative feedback features, both alone and in combination (IS+DynKB+F1, IS+DynKB+F2 and IS+DynKB+F1+F2). For all of them, the cut-off similarity score is obtained by grid search and is set to 0.8. The results achieved by each system are reported in Table 2.

	BLEU↑	TER↓	Prec. (%)
MT	52.31	34.52	N/A
Batch APE	52.52	34.45	42.67
Thot	52.51	34.37	42.22
IS	53.35†	33.36†	58.47
IS+DynKB	53.60†	33.23†	59.69
IS+DynKB+F1	53.56†	33.29†	58.97
IS+DynKB+F2	53.21†	33.48†	54.64
IS+DynKB+F1+F2	**53.77**†	**33.20**†	**60.93**

Table 2: Results on the mixed data. ("†" indicates statistically significant difference wrt. the MT baseline with $p<0.05$).

As can be seen from the table, the batch APE system is able to slightly improve over the baseline even if it damages most of the translations it changes (its precision is in fact lower than 45%). Although it learns also from the test data, Thot achieves similar results. This is probably due to its inability to identify domain-specific correction rules needed to improve the translations, thus ending up with damaging the majority of the modified MT segments. A significant gain in performance (+1.04 BLEU, -1.16 TER points) is obtained by our online IS system that, by using the instance selection technique, is able to isolate only the most useful training samples. This mechanism also improves precision up to 58.4% ($\sim$16% above Thot), indicating that the applied post-editing rules are correct in the majority of the cases. The analysis of the performance of the two online systems reveals that our approach modifies less segments compared to Thot, due to the fact that it builds sentence-specific models only if it finds relevant data, leaving the MT segment untouched otherwise. In several cases, when modified by the Thot system, these untouched segments result in deteriorated sentences.

Further performance improvements are yield by the dynamic knowledge base (IS+DynKB), which provides the decoder with a better estimation of the APE model parameters. Although the BLEU and TER gains are minimal, the dynamic knowledge base helps to significantly increase the precision of the APE system avoiding unnecessary changes, thus confirming the effectiveness of keeping track of the whole past history of each translation option. Our implementation of the dynamic knowledge base also allows us to add the two "negative feedback" features that model the reliability of the translation options by looking at the changes made by the post-editors. When used in combination, the two negative feedback features (IS+DynKB+F1+F2) yield visible gains in performance over (IS+DynKB) with small but statistically significant improvement in BLEU score, along with a precision gain of 1.24%. This suggests their possible complementarity with the translation and reordering features and the need of further investigation in future work. Overall our full-fledged system achieves state-of-the-art results with significant improvement over Thot by 1.26 BLEU, 1.17 TER, and 18.71% Precision.

	BLEU↑	TER↓	Prec. (%)
	Autodesk		
MT	40.01	45.42	N/A
Batch APE	43.13†	43.19†	58.86
Thot	43.40†	42.96†	59.04
IS+DynKB+F1+F2	**44.56**†	**41.86**†	**73.37**
	WMT16		
MT	61.04	26.24	N/A
Batch APE	59.24†	27.81†	22.18
Thot	59.05†	27.84†	20.06
IS+DynKB+F1+F2	**60.39**†	**26.62**†	**36.67**

Table 3: Performance analysis of each domain.

6 Analysis

To understand and compare the behavior of the batch APE, Thot and our best online system in the long-run, the plot in Figure 1 shows the TER moving average (window of 750 data points) at each segment of the test set (second half of the shuffled training data). As can be seen, our approach successfully maintains the best performance across the entire test set. Moreover, looking at the first

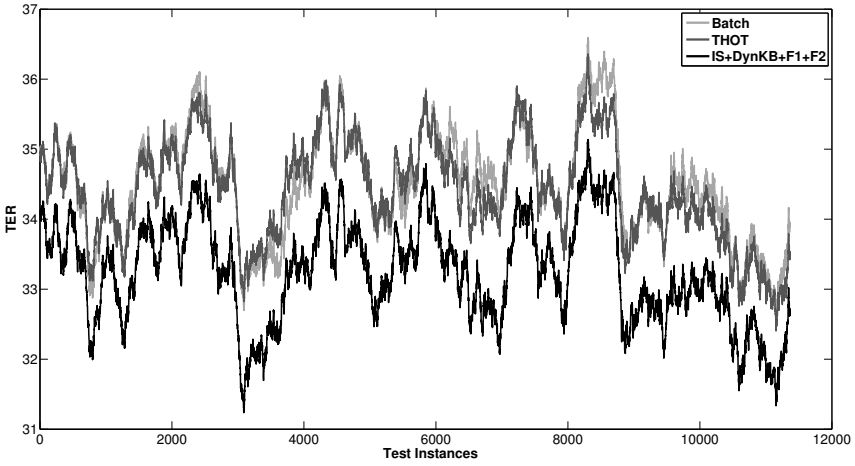

Figure 1: Performance comparison of different online APE systems (TER moving average).

SRC	Specifies the value to define the mid-ordinate distance by which to tessellate baseline alignment curves .
MT	Gibt den Wert für den krzesten Abstand vom Sekantenmittelpunkt zu Kreisbogen für die Tessellation Basislinienachse Kurven .
MT-Top1	Gibt den Wert für den krzesten Abstand vom Sekantenmittelpunkt zu Kreisbogen für die Tessellation Basislinienachse Kurven .
PE-Top1	Gibt den Wert zum Definieren des krzesten Abstands vom Sekantenmittelpunkt zum Kreisbogen an , um den Basislinienachsen-Bogen ausgerundet werden sollen .
THOT	Gibt den Wert **für den Versatzzielbogen Abstand** vom Sekantenmittelpunkt **zu** Kreisbogen **für die Tessellation Basislinienachse Kurven** .
IS+DynKB +F1+F2	Gibt den Wert zum Definieren des krzesten Abstands vom Sekantenmittelpunkt zum Kreisbogen an , um den **Basislinienachse Versatzzielbogen** ausgerundet werden sollen .
REF	Gibt den Wert zum Definieren des krzesten Abstands vom Sekantenmittelpunkt zum Kreisbogen an , um den Basislinienachsen-Bogen ausgerundet werden sollen .

Table 4: Sample outputs where our approach outperform THOT (erroneous words are in bold)

and the last 250 points in the test set, we notice that the performance gap between our best system and Thot increases on average from 0.8 to 1.6 TER points. This shows that, during processing, our approach is able to self-adapt in real-time towards the domain-shifts in the input stream of data. To better understand their behavior with respect to data coming from the two domains, systems' output has been separately evaluated per domain. The results of this evaluation are reported in Table 3.

For the Autodesk and the APE shared task domain there are 6,166 and 5,953 segments respectively. It is interesting to see that all the APE systems improve the translations belonging to the Autodesk domain by a large margin, with our approach being the best. The same does not hold for the other domain, which, as discussed in Section 5, is more challenging due to several factors like longer sentence length, higher data sparsity and, most of all, lower translation quality. For this challenging domain, however, our approach has the least degradation compared to the other APE methods that severely damage the translations. Overall, compared to other APE approaches, our best system has the best performance in both the domains when evaluated in isolation.

To evaluate the efficiency of our approach, we computed the average time in seconds to perform a full online cycle over the test set (*i.e.* the time needed for post-editing the MT output and updating the models) for Thot, IS and IS+DynKB+F1+F2. Thot spends on average 4.75 seconds per cycle. The IS system, which builds its models on-the-fly by leveraging only the selected

SRC	Drag to the left and then click to place .
MT	Ziehen Sie nach links , und klicken Sie dann , um sie zu platzieren .
MT-Top1	Ziehen Sie nach links , und klicken Sie dann , um sie zu platzieren .
PE-Top1	Ziehen Sie sie nach links , und klicken Sie dann , um sie zu platzieren .
MT-Top4	Ziehen Sie den Cursor nach unten und nach rechts , und klicken Sie dann , um sie zu platzieren .
PE-Top4	Ziehen Sie nach unten und nach rechts , und klicken Sie dann zum Platzieren .
THOT	Ziehen Sie nach links , und klicken Sie dann , um sie zu platzieren .
IS+DynKB +F1+F2	Ziehen Sie nach links , und klicken Sie dann **zum Platzieren** .
REF	Ziehen Sie sie nach links , und klicken Sie dann , um sie zu platzieren .

Table 5: Sample output where our approach performs poor than THOT (erroneous words are in bold)

data and optimises the weights before post-editing, is faster than Thot, with a gain of 1.03 seconds (3.62" on average). The use of the dynamic model, that is faster in updating and dumping the tables, allows our best system to perform a full online cycle in 3.05", showing that our approach is not only better in terms of performance but also in computation time.

Tables 4 and 5 respectively show examples where our approach performs better/worse than Thot. Both tables report the source sentence (SRC), the MT output to be post-edited (MT), the MT and the PE segment of the top training samples retrieved based on cosine similarity (MT-TopX/PE-TopX, where X is the rank of the training sample), the output of Thot, the output of our best system (IS+DynKB+F1+F2) and the reference (REF). Table 4 seems to confirm our intuition that learning from the most similar examples yields better translation quality. An interesting counter example is shown in Table 5, where despite having access to a training sample (MT-Top1 and PE-Top1) that is exactly the same as the MT segment to be post-edited, our system deteriorates the translation by selecting a translation option (*"zu platzieren"* → *"zum Platzieren"*) learned from a lower ranked training sample (MT-Top4 and PE-Top4), which probably received a higher weight from the local models. In future work, we will try to extend our system to address this issue by prioritizing the translation rules according to the rank of the training samples.

7 Conclusion

In recent years, APE systems achieved impressive results in fixing recurrent errors in machine-translated texts. Such gains, however, were mostly observed in controlled lab environments, where systems are trained, tuned, and evaluated with repetitive and homogeneous training/test data. These favorable learning conditions may not hold in real-world professional translation workflow, in which streams of data to be processed in real-time may feature a high diversity in terms of domain, post-editing style and MT systems that generated the translations. In this paper, we investigated for the first time the challenges posed to APE technology by such multi-domain translation environments. Our study revealed that the existing online and batch solutions are not robust enough for this scenario due to their inability to discern which of the learned rules is suitable for a specific context (in fact, a correction rule learned from one domain may not be valid for other domains). We addressed this problem incrementally, first by proposing an instance selection technique that learns rules from contexts that are similar to the MT segment to be post-edited. The gains achieved by this solution over the existing batch APE methods were further increased by the addition of a dynamic knowledge base that stores more reliable statistics about the learned translation options, also improving the computation time. The adoption of this dynamic knowledge base allowed us to further extend our online approach by including features that capture negative human feedback, giving to the system the capability to learn from the mistakes it made in the past. Our evaluation results indicate that our approach improves state of the art performance on an English-German MDTE data set.

8 Acknowledgement

This work has been partially supported by the EC-funded H2020 project QT21 (grant no. 645452)

References

Hanna Béchara, Yanjun Ma, and Josef van Genabith. 2011. Statistical Post-Editing for a Statistical MT System. In *Proceedings of the 13th Machine Translation Summit*, pages 308–315.

Nicola Bertoldi, Mauro Cettolo, and Marcello Federico. 2013. Cache-based Online Adaptation for Machine Translation Enhanced Computer Assisted Translation. *Proceedings of the XIV Machine Translation Summit*, pages 35–42.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Barry Haddow, Matthias Huck, Chris Hokamp, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Carolina Scarton, Lucia Specia, and Marco Turchi. 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, September.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurelie Neveol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. 2016. Findings of the 2016 Conference on Machine Translation. In *Proceedings of the First Conference on Machine Translation*, pages 131–198, Berlin, Germany, August. Association for Computational Linguistics.

Rajen Chatterjee, Marco Turchi, and Matteo Negri. 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.

Rajen Chatterjee, Marion Weller, Matteo Negri, and Marco Turchi. 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, July.

Rajen Chatterjee, José G. C. de Souza, Matteo Negri, and Marco Turchi. 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, August. Association for Computational Linguistics.

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 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.

Koby Crammer and Yoram Singer. 2003. Ultraconservative Online Algorithms for Multiclass Problems. *The Journal of Machine Learning Research*, 3:951–991.

Michael Denkowski, Chris Dyer, and Alon Lavie. 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, April.

Loïc Dugast, Jean Senellart, and Philipp Koehn. 2007. Statistical Post-Editing on SYSTRAN's Rule-Based Translation System. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 220–223.

Marcello Federico, Nicola Bertoldi, and Mauro Cettolo. 2008. IRSTLM: an Open Source Toolkit for Handling Large Scale Language Models. In *Proceedings of the Interspeech*, pages 1618–1621.

Qin Gao and Stephan Vogel. 2008. Parallel Implementations of Word Alignment Tool. In *Proceedings of Software Engineering, Testing, and Quality Assurance for Natural Language Processing*, pages 49–57.

Ulrich Germann. 2014. Dynamic Phrase Tables for Machine Translation in an Interactive Post-editing Scenario. In *Proceedings of the Workshop on interactive and adaptive machine translation*, pages 20–31.

Daniel Hardt and Jakob Elming. 2010. Incremental Re-training for Post-editing SMT. In *Proceedings of the Ninth Conference of the Association for Machine Translation in the Americas*.

Almut Silja Hildebrand, Matthias Eck, Stephan Vogel, and Alex Waibel. 2005. Adaptation of the Translation Model for Statistical Machine Translation based on Information Retrieval. In *Proceedings of the 10th Annual Conference of the European Association for Machine Translation*, pages 133–142.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2016. Log-linear Combinations of Monolingual and Bilingual Neural Machine Translation Models for Automatic Post-Editing. In *Proceedings of the First Conference on Machine Translation*, pages 751–758, Berlin, Germany, August. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, 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.

Philipp Koehn. 2004. Statistical Significance Tests for Machine Translation Evaluation. In *Proceedings of the Empirical Methods on Natural Language Processing*, pages 388–395.

Antonio L Lagarda, Daniel Ortiz-Martínez, Vicent Alabau, and Francisco Casacuberta. 2015. Translating without In-domain Corpus: Machine Translation Post-Editing with Online Learning Techniques. *Computer Speech & Language*, 32(1):109–134.

Prashant Mathur, Mauro Cettolo, Marcello Federico, and FBK-Fondazione Bruno Kessler. 2013. Online Learning Approaches in Computer Assisted Translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 301–308.

Daniel Ortiz-Martınez and Francisco Casacuberta. 2014. The New THOT Toolkit for Fully-Automatic and Interactive Statistical Machine Translation. In *14th Annual Meeting of the European Association for Computational Linguistics*, pages 45–48.

Santanu Pal, Marcos Zampieri, and Josef van Genabith. 2016. USAAR: An Operation Sequential Model for Automatic Statistical Post-Editing. In *Proceedings of the First Conference on Machine Translation*, pages 759–763, Berlin, Germany, August. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 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.

Kristen Parton, Nizar Habash, Kathleen McKeown, Gonzalo Iglesias, and Adrià de Gispert. 2012. Can Automatic Post-Editing Make MT More Meaningful? In *Proceedings of the 16th Annual Conference of the European Association for Machine Translation*, pages 111–118.

Abdol Hamid Pilevar. 2011. Using Statistical Post-editing to Improve the Output of Rule-based Machine Translation System. *International Journal of Computer Science and Communication*.

Michel Simard and George Foster. 2013. Pepr: Post-edit Propagation Using Phrase-based Statistical Machine Translation. In *Proceedings of the XIV Machine Translation Summit*, pages 191–198.

Michel Simard, Cyril Goutte, and Pierre Isabelle. 2007. 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.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A Study of Translation Edit Rate with Targeted Human Annotation. In *Proceedings of the Association for Machine Translation in the Americas*, pages 223–231.

O. Tange. 2011. GNU Parallel - The Command-Line Power Tool. *;login: The USENIX Magazine*, 36(1):42–47, Feb.

Ehara Terumasa. 2007. Rule Based Machine Translation Combined with Statistical Post Editor for Japanese to English Patent Translation. In *Proceedings of the XI Machine Translation Summit*, pages 13–18.

Joern Wuebker, Spence Green, and John DeNero. 2015. Hierarchical Incremental Adaptation for Statistical Machine Translation. In *Proceedings of the Empirical Methods in Natural Language Processing*, pages 1059–1065, September.