TMop: a Tool for Unsupervised Translation Memory Cleaning

Masoud Jalili Sabet$^{(1)}$, **Matteo Negri**$^{(2)}$, **Marco Turchi**$^{(2)}$,
José G. C. de Souza$^{(2)}$, **Marcello Federico**$^{(2)}$
$^{(1)}$ School of Electrical and Computer Engineering, University of Tehran, Iran
$^{(2)}$ Fondazione Bruno Kessler, Trento, Italy
jalili.masoud@ut.ac.ir
{negri,turchi,desouza,federico}@fbk.eu

Abstract

We present TMop, the first open-source tool for automatic Translation Memory (TM) cleaning. The tool implements a fully unsupervised approach to the task, which allows spotting unreliable translation units (sentence pairs in different languages, which are supposed to be translations of each other) without requiring labeled training data. TMop includes a highly configurable and extensible set of filters capturing different aspects of translation quality. It has been evaluated on a test set composed of 1,000 translation units (TUs) randomly extracted from the English-Italian version of MyMemory, a large-scale public TM. Results indicate its effectiveness in automatic removing "bad" TUs, with comparable performance to a state-of-the-art supervised method (76.3 vs. 77.7 balanced accuracy).

1 Introduction

Computer-assisted translation (CAT) refers to a framework in which the work of human translators is supported by machines. Its advantages, especially in terms of productivity and translation consistency, have motivated huge investments both economic (by the translation industry) and intellectual (by the research community). Indeed, the high market potential of solutions geared to speed up the translation process and reduce its costs has attracted increasing interest from both sides.

Advanced CAT tools currently integrate the strengths of two complementary technologies: translation memories (TM - a high-precision mechanism for storing and retrieving previously translated segments) and machine translation (MT - a high-recall technology for translating unseen segments). The success of the integration has determined the quick growth of market shares that are held by CAT, as opposed to fully manual translation that became a niche of the global translation market. However, differently from MT that is constantly improving and reducing the distance from human translation, core TM technology has slightly changed over the years. This is in contrast with the fact that TMs are still more widely used than MT, especially in domains featuring high text repetitiveness (e.g. software manuals).

Translation memories have a long tradition in CAT, with a first proposal dating back to (Arthern, 1979). They consist of databases that store previously translated segments, together with the corresponding source text. Such (*source*, *target*) pairs, whose granularity can range from the phrase level to the sentence or even the paragraph level, are called translation units (TUs). When working with a CAT tool, each time a segment of a document to be translated matches with the *source* side of a TU, the corresponding *target* is proposed as a suggestion to the user. The user can also store each translated (*source*, *target*) pair in the TM for future use, thus increasing the size and the coverage of the TM. Due to such constant growth, in which they evolve over time incorporating users style and terminology, the so-called private TMs represent an invaluable asset for individual translators and translation companies. Collaboratively-created public TMs grow in a less controlled way but still remain a practical resource for the translators' community at large.

The usefulness of TM suggestions mainly depends on two factors: the matching process and the quality of the TU. To increase recall, the retrieval is based on computing a "fuzzy match" score. Depending on how the matching is performed, its output can be a mix of perfect and partial matches requiring variable amounts of corrections by the user. For this reason, most prior works on TM technology focused on improving this aspect (Gupta et al., 2014; Bloodgood and Strauss, 2014; Vanallemeersch and Vandeghinste, 2015; Chatzitheodoroou, 2015; Gupta et al., 2015).

The other relevant factor, TU quality, relates to the reliability of the *target* translations. Indeed, a perfectly matching *source* text associated to a wrong translation would make the corresponding suggestion useless or, even worse, an obstacle to productivity. On this aspect, prior research is limited to the work proposed in (Barbu, 2015), which so far represents the only attempt to automatically spot false translations in the bi-segments of a TM. However, casting the problem as a supervised binary classification task, this approach highly depends on the availability of labelled training data.

Our work goes beyond the initial effort of Barbu (2015) in two ways. First, **we propose a configurable and extensible open source framework for TM cleaning**. In this way, we address the demand of easy-to-use TM management tools whose development is out of the reach of individual translators and translation companies. Such demand is not only justified by productivity reasons (remove bad suggestions as a cause of slow production), but also for usability reasons. Loading, searching and editing a TM are indeed time-consuming and resource-demanding operations. In case of very large databases (up to millions of TUs) the accurate removal of useless units can significantly increase usability. Though paid, the few existing tools that incorporate some data cleaning methods (e.g. Apsic X-Bench[1]) only implement very simple syntactic checks (e.g. repetitions, opening/closing tags consistency). These are insufficient to capture the variety of errors that can be encountered in a TM (especially in the public ones).

Second, **our approach to TM cleaning is fully unsupervised**. This is to cope with the lack of labelled training data which, due to the high acquisition costs, represents a bottleneck rendering supervised solutions unpractical. It is worth remarking that also current approaches to tasks closely related to TM cleaning (e.g. MT quality estimation (Mehdad et al., 2012; C. de Souza et al., 2014)) suffer from the same problem. Besides not being customised for the specificities of the TM cleaning scenario (their usefulness for the task should be demonstrated), their dependence on labelled training data is a strong requirement from the TM cleaning application perspective.

2 The TM cleaning task

The identification of “bad” TUs is a multifaceted problem. First, it deals with the recognition of *a variety of errors*. These include:

- Surface errors, such as opening/closing tags inconsistencies and empty or suspiciously long/short translations;
- Language inconsistencies, for instance due to the inversion between the *source* and *target* languages;
- Translation fluency issues, such as typos and grammatical errors (e.g. morpho-syntactic disagreements, wrong word ordering);
- Translation adequacy issues, such as the presence of untranslated terms, wrong lexical choices or more complex phenomena (e.g. negation and quantification errors) for which a syntactically correct *target* can be a semantically poor translation of the *source* segment.

The *severity* of the errors is another aspect to take into account. Deciding if a given error makes a TU useless is often difficult even for humans. For instance, judging about the usefulness of a TU whose *target* side has missing/extra words would be a highly subjective task.[2] For this reason, identifying “bad” TUs with an automatic approach opens a number of problems related to: *i)* defining when a given issue becomes a real error (e.g. the ratio of acceptable missing words), *ii)* combining potentially contradictory evidence (e.g. syntactic and semantic issues), and *iii)* making these actions easily customisable by different users having different needs, experience and quality standards.

What *action* to take when one or more errors are identified in a TU is also important. Ideally, a TM cleaning tool should allow users either to simply flag problematic TUs (leaving the final decision to a human judgment), or to automatically remove them without further human intervention.

Finally, two critical aspects are the external *knowledge* and *resources* required by the TM-cleaning process. On one side, collecting evidence

[1]`http://www.xbench.net/`

[2]Likely, the perceived severity of a missing word out of n perfectly translated terms will be inversely proportional to n.

for each TU can involve processing steps that access external data and tools. On the other side, decision making can require variable amounts of labelled training data (i.e. positive/negative examples of "good"/"bad" TUs). For both tasks, the recourse to external support can be an advantage in terms of performance due to the possibility to get informed judgments taken from models trained in a supervised fashion. At the same time, it can be a limitation in terms of usability and portability across languages. When available, external resources and tools (e.g. syntactic/semantic parsers) can indeed be too slow to process huge amounts of data. Most importantly, labelled training data are usually difficult to acquire. In case of need, a TM cleaning tool should hence minimise the dependence of its performance from the availability of external resources.

All these aspects were considered in the design of TMop, whose capability to cope with a variety of errors, customise its actions based on their severity and avoid the recourse to external knowledge/resources are described in the next section.

3 The TMop framework

TMop (Translation Memory open-source purifier) is an open-source TM cleaning software written in Python. It consists of three parts: core, filters and policy managers. The core, the main part of the software, manages the workflow between filters, policy managers and input/output files. The filters (§3.2) are responsible for detecting "bad" TUs. Each of them can detect a specific type of problems (e.g. formatting, fluency, adequacy) and will emit an *accept* or *reject* judgment for each TU. Policy managers (§3.3) collect the individual results from each filter and take a final decision for each TM entry based on different possible strategies. Filters, policies and basic parameters can be set by means of a configuration file, which was structured by keeping ease of use and flexibility as the main design criteria.

TMop implements a fully unsupervised approach to TM cleaning. The *accept/reject* criteria are learned from the TM itself and no training data are required to inform the process.[3] Nevertheless, the filters' output could be also used to instantiate feature vectors in any supervised learning scenario supported by training data.

[3]The tool has been recently used also in the unsupervised approach by Jalili Sabet et al. (2016).

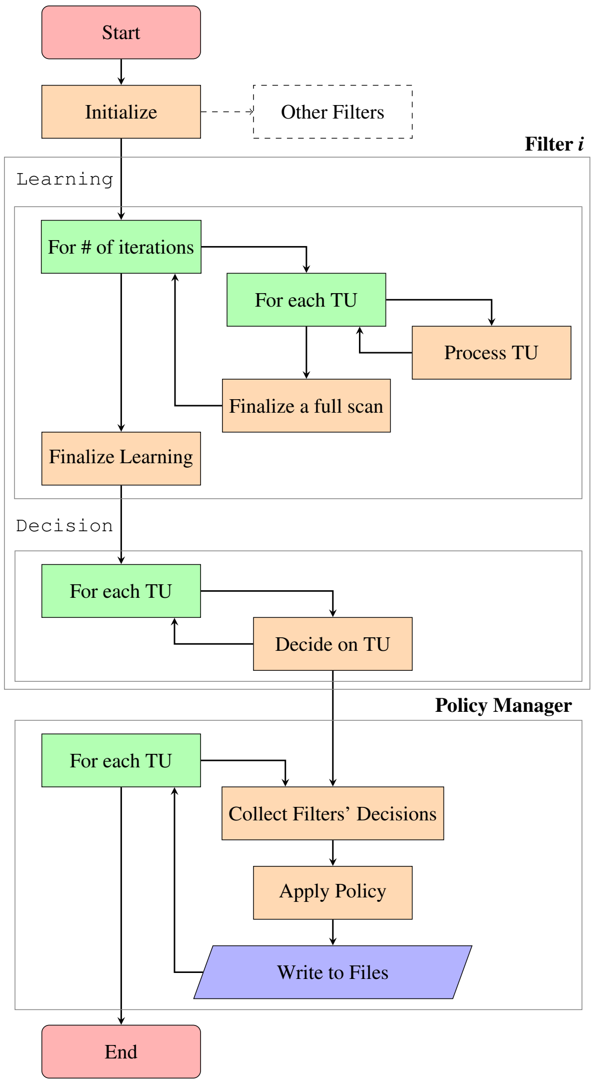

Figure 1: TMop workflow

3.1 Workflow

The input file of TMop is a TM represented as a text file containing one TU per line in the form (*ID*, *source*, *target*). The output consists of several files, the most important of which are the *accept* and *reject* files containing the TUs identified as "good"/"bad", in the same format of the input. As depicted in Figure 1, TMop **filters** operate in two steps. In the first one, the `learning` step, each filter i iterates over the TM or a subset of it to gather the basic statistics needed to define its *accept/reject* criteria. For instance, by computing mean and standard deviation values for a given indicator (e.g. sentence length ratio, proportion of aligned words), quantiles or std counts in case of normal value distributions will be used as decision boundaries. Then, in the `decision` step, each filter uses the gathered information to decide about each TU. At the end of this process, for each

TU the **policy manager** collects all the decisions taken by the filters and applies the policy set by the user in the configuration file to assign an *accept* or *reject* judgment. The final labels, the TUs and the filters outputs are saved in different files.

3.2 Filters

Our filters capture different aspects of the similarity between the source and the target of a TU. The full set consists of 23 filters, which are organized in four groups.

Basic filters (8 in total). This group (**B**) extends the filters proposed by Barbu (2015) and substantially covers those offered by commercial TM cleaning tools. They capture translation quality by looking at surface aspects, such as the possible mismatches in the number of dates, numbers, URLs, XML tags, ref and image tags present in the source and target segments. Other filters model the similarity between source and target by computing the direct and inverse ratio between the number of characters and words, as well as the average word length in the two segments. Finally, two filters look for uncommon character or word repetitions.

Language identification filter (1). This filter (**LI**) exploits the Langid tool (Lui and Baldwin, 2012) to verify the consistency between the source and target languages of a TU and those indicated in the TM. Though simple, it is quite effective since often the two languages are inverted or even completely different from the expected ones.

QE-derived filters (9). This group (**QE**) contains filters borrowed from the closely-related task of MT quality estimation, in which the complexity of the source, the fluency of the target and the adequacy between source and target are modeled as quality indicators. Focusing on the adequacy aspect, we exploit a subset of the features proposed by C. de Souza et al. (2013). They use word alignment information to link source and target words and capture the quantity of meaning preserved by the translation. For each segment of a TU, word alignment information is used to calculate: *i)* the proportion of aligned and unaligned word n-grams (n=1,2), *ii)* the ratio between the longest aligned/unaligned word sequence and the length of the segment, *iii)* the average length of the aligned/unaligned word sequences, and *iv)* the position of the first/last unaligned word, normalized by the length of the segment. Word alignment models can be trained on the whole TM with one of the many existing word aligners. For instance, the results of WE filters reported in §4 were obtained using MGIZA++ (Gao and Vogel, 2008).

Word embedding filters (5). Cross-lingual word embeddings provide a common vector representation for words in different languages and allow looking at the source and target segments at the same time. In TMop, they are computed using the method proposed in (Søgaard et al., 2015) but, instead of considering bilingual documents as atomic concepts to bridge the two languages, they exploit the TUs contained in the TM itself. Given a TU and a 100-dimensional vector representation of each word in the source and target segments, this group of filters (**WE**) includes: *i)* the cosine similarity between the source and target segment vectors obtained by averaging (or using the median) the source and target word vectors; *ii)* the average embedding alignment score obtained by computing the cosine similarity between each source word and all the target words and averaging over the largest cosine score of each source word; *iii)* the average cosine similarity between source/target word alignments; *iv)* a score that merges features *(ii)* and *(iii)* by complementing word alignments (also in this case obtained using MGIZA++) with the alignments obtained from word embedding and averaging all the alignment weights.

3.3 Policies

Decision policies allow TMop combining the output of the active filters into a final decision for each TU. Simple decision-making strategies can consider the number of *accept* and *reject* judgments, but more complex methods can be easily implemented by the user (both filters and policy managers can be easily modified and extended by exploiting well-documented abstract base classes).

TMop currently implements three policies: *OneNo*, *20%No* and *MajorityVoting*. The first one copies a TU in the *reject* file if at least one filter rejects it. The second and the third policy take this decision only if at least twenty or fifty percent of the filters reject the TU respectively.

These three policies reflect different TM cleaning strategies. The first one is a very aggressive (recall-oriented) solution that tends to flag more TUs as “bad”. The third one is a more conservative (precision-oriented) solution, as it requires at least half of the judgments to be negative for pushing a TU in the *reject* file. Depending on the user needs and the overall quality of the TM, the choice of the policy will allow keeping under control the number of false positives ("bad" TUs accepted) and false negatives ("good" TUs rejected).

4 Benchmarking

We test TMop on the English-Italian version of MyMemory,[4] one of the world's largest collaborative public TMs. This dump contains about 11M TUs coming from heterogeneous sources: aggregated private TMs, either provided by translators or automatically extracted from the web/corpora, as well as anonymous contributions of (*source*, *target*) bi-segments. Its uncontrolled sources call for accurate cleaning methods (e.g. to make it more accurate, smaller and manageable).

From the TM we randomly extracted a subset of 1M TUs to compute the statistics of each filter and a collection of 2,500 TUs manually annotated with binary labels. Data annotation was done by two Italian native speakers properly trained with the same guidelines prepared by the TM owner for periodic manual revisions. After agreement computation (Cohen's kappa is 0.78), a reconciliation ended up with about 65% positive and 35% negative examples. This pool is randomly split in two parts. One (1,000 instances) is used as test set for our evaluation. The other (1,500 instances) is used to replicate the supervised approach of Barbu (2015), which leverages human-labelled data to train an SVM binary classifier. We use it as a term of comparison to assess the performance of the different groups of filters.

To handle the imbalanced (65%-35%) data distribution, and equally reward the correct classification on both classes, we evaluate performance in terms of balanced accuracy (BA), computed as the average of the accuracies on the two classes (Brodersen et al., 2010).

In Table 1, different combinations of the four groups of filters are shown with results aggregated with the *20%No* policy, which, on this data, results to be the best performing policy among the ones implemented in TMop. Based on the statistics collected in the `learning` phase of each filter, the *accept/reject* criterion applied in these experiments considers as "good" all the TUs for which the filter value is below one standard deviation from the mean and "bad" otherwise.

[4]`http://mymemory.translated.net`

Filters	**BA**$\uparrow$
(Barbu, 2015)	77.7
B	52.8
LI	69.0
QE	71.2
WE	65.0
B + LI	55.4
B + QE	70.1
B + WE	68.7
QE + LI	71.7
QE + WE	67.9
LI + WE	68.1
B + QE + LI	72.9
B + WE + LI	70.3
B + QE + WE	73.3
B + QE + LI + WE	76.3

Table 1: Balanced accuracy of different filter combinations on a 1,000 TU, EN-IT test set. B=Basic, LI=language identification, QE=quality estimation, WE=word embedding.

Looking at the results, it is worth noting that the LI, QE and WE groups, both alone and in combination, outperform the basic filters (B), which substantially represent those implemented by commercial tools. Although relying on an external component (the word aligner), QE filters produce the best performance in isolation, showing that word alignment information is a good indicator of translation quality. The results obtained by combining the different groups confirm their complementarity. In particular, when using all the groups, the performance is close to the results achieved by the supervised method by Barbu (2015), which relies on human-labelled data (76.3 vs. 77.7).

The choice of which filter combination to use strongly depends on the application scenario and it is often a trade-off. A first important aspect concerns the type of user. When the expertise to train a word aligner is not available, combining B, WE and LI is the best solution, though it comes at the cost of lower accuracy. Another aspect is the processing time that the user can afford. TM cleaning is an operation conceived to be performed once in a while (possibly overnight), once the TM has grown enough to justify a new sanity check. However, although it does not require real-time processing, the size of the TM can motivate the selection of faster filter combinations. An analysis of the efficiency of the four groups, made by counting the number of processed TUs per second,[5] indicates that B and QE are the fastest filters (processing on average ∼2,000 TUs/sec.). The LI filter is slower, processing ∼300 TUs per second, while the large number of times the cosine similarity score is computed does not allow the WE filter to process more than 50 TUs per second.

5 Conclusion

We presented TMop, the first open-source tool for automatic Translation Memory (TM) cleaning. We summarised its design criteria, workflow and main components, also reporting some efficiency and performance indicators. TMop is implemented in Python and can be downloaded, together with complete documentation, from `https://github.com/hlt-mt/TMOP`. Its license is FreeBSD, a very open permissive non-copyleft license, compatible with the GNU GPL and with any use, including commercial.

Acknowledgments

This work has been partially supported by the EC-funded project ModernMT (H2020 grant agreement no. 645487). The work carried out at FBK by Masoud Jalili Sabet was sponsored by the EAMT summer internships 2015 program and supported by Prof. Heshaam Faili (University of Tehran). The authors would also like to thank Translated for providing a dump of MyMemory.

References

Peter Arthern. 1979. Machine Translation and Computerized Terminology Systems: a Translator's Viewpoint. In *Translating and the computer. Proc. of a seminar*, pages 77–108, London, UK.

Eduard Barbu. 2015. Spotting False Translation Segments in Translation Memories. In *Proc. of the Workshop Natural Language Processing for Translation Memories*, pages 9–16, Hissar, Bulgaria.

Michael Bloodgood and Benjamin Strauss. 2014. Translation Memory Retrieval Methods. In *Proc. of the 14th Conference of the EACL*, pages 202–210, Gothenburg, Sweden.

Kay Henning Brodersen, Cheng Soon Ong, Klaas Enno Stephan, and Joachim M. Buhmann. 2010. The Balanced Accuracy and Its Posterior Distribution. In *Proc. of the 2010 20th International Conference on Pattern Recognition*, ICPR '10, pages 3121–3124.

José G. C. de Souza, Christian Buck, Marco Turchi, and Matteo Negri. 2013. FBK-UEdin Participation to the WMT13 Quality Estimation Shared Task. In *Proc. of the Eighth Workshop on Statistical Machine Translation*, pages 352–358, Sofia, Bulgaria. Association for Computational Linguistics.

José G. C. de Souza, Jesús González-Rubio, Christian Buck, Marco Turchi, and Matteo Negri. 2014. FBK-UPV-UEdin Participation in the WMT14 Quality Estimation Shared-task. In *Proc. of the Ninth Workshop on Statistical Machine Translation*, pages 322–328, Baltimore, Maryland, USA.

Konstantinos Chatzitheodoroou. 2015. Improving Translation Memory Fuzzy Matching by Paraphrasing. In *Proc. of the Workshop Natural Language Processing for Translation Memories*, pages 24–30, Hissar, Bulgaria.

Qin Gao and Stephan Vogel. 2008. Parallel Implementations of Word Alignment Tool. In *In Proc. of the ACL 2008 Software Engineering, Testing, and Quality Assurance Workshop*.

Rohit Gupta, Hanna Bechara, and Constantin Orasan. 2014. Intelligent Translation Memory Matching and Retrieval Metric Exploiting Linguistic Technology. In *Proc. of Translating and the Computer: Vol. 36.*, pages 86–89.

Rohit Gupta, Constantin Orasan, Marcos Zampieri, Mihaela Vela, and Josef Van Genabith. 2015. Can Translation Memories afford not to use paraphrasing? In *Proc. of the 18th Annual Conference of the European Association for Machine Translation*, pages 35–42, Antalya, Turkey.

Masoud Jalili Sabet, Matteo Negri, Marco Turchi, and Eduard Barbu. 2016. An Unsupervised Method for Automatic Translation Memory Cleaning. In *Proc. of the 54th Annual Meeting of the Association for Computational Linguistics*, Berlin, Germany.

Marco Lui and Timothy Baldwin. 2012. langid.py: An Off-the-shelf Language Identification Tool. In *Proc. of the ACL 2012 system demonstrations*, pages 25–30. Association for Computational Linguistics.

Yashar Mehdad, Matteo Negri, and Marcello Federico. 2012. Match without a Referee: Evaluating MT Adequacy without Reference Translations. In *Proc. of the Machine Translation Workshop (WMT2012)*, pages 171–180, Montréal, Canada.

Anders Søgaard, Željko Agić, Héctor Martínez Alonso, Barbara Plank, Bernd Bohnet, and Anders Johannsen. 2015. Inverted indexing for cross-lingual NLP. In *The 53rd Annual Meeting of the Association for Computational Linguistics (ACL 2015)*.

Tom Vanallemeersch and Vincent Vandeghinste. 2015. Assessing Linguistically Aware Fuzzy Matching in Translation Memories. In *Proc. of the 18th Annual Conference of the European Association for Machine Translation*, pages 153–160, Antalya, Turkey.

[5]Experiments were run with a PC with an Intel Core i5 M540 @ 2.53GHz and 6 GB RAM.