Seeding Statistical Machine Translation with Translation Memory Output through Tree-Based Structural Alignment

Ventsislav Zhechev
EuroMatrixPlus, CNGL
School of Computing, Dublin City University
contact@VentsislavZhechev.eu

Josef van Genabith
EuroMatrixPlus, CNGL
School of Computing, Dublin City University
josef@computing.dcu.ie

Abstract

With the steadily increasing demand for high-quality translation, the localisation industry is constantly searching for technologies that would increase translator throughput, with the current focus on the use of high-quality Statistical Machine Translation (SMT) as a supplement to the established Translation Memory (TM) technology. In this paper we present a novel modular approach that utilises state-of-the-art sub-tree alignment to pick out pre-translated segments from a TM match and seed with them an SMT system to produce a final translation. We show that the presented system can outperform pure SMT when a good TM match is found. It can also be used in a Computer-Aided Translation (CAT) environment to present almost perfect translations to the human user with markup highlighting the segments of the translation that need to be checked manually for correctness.

1. Introduction

As the world becomes increasingly interconnected, the major trend is to try to deliver ideas and products to the widest audience possible. This requires the localisation of products for as many countries and cultures as possible, with translation being one of the main parts of the localisation process. Because of this, the amount of data that needs professional high-quality translation is continuing to increase well beyond the capacity of the world's human translators.

Thus, current efforts in the localisation industry are mostly directed at the reduction of the amount of data that needs to be translated from scratch by hand. Such efforts mainly include the use of Translation Memory (TM) systems, where earlier translations are stored in a database and offered as suggestions when new data needs to be translated. As TM systems were originally limited to providing translations only for (almost) exact matches of the new data, the integration of Machine Translation (MT) techniques is seen as the only feasible development that has the potential to significantly reduce the amount of manual translation required.

At the same time, the use of SMT is frowned upon by the users of CAT tools as they still do not trust the quality of the SMT output. There are two main reasons for that. First, currently there is no reliable way to automatically ascertain the quality of SMT-generated translations, so that the user could at a glance make a judgement as to the amount of effort that might be needed to post-edit the suggested translation (Simard and Isabelle, 2009). Not having such automatic quality metrics also has the side effect of it being impossible for a Translation-Services Provider (TSP) company to reliably determine in advance the increase in translator productivity due to the use of MT and to adjust their resources-allocation and cost models correspondingly.

The second major problem for users is that SMT-generated translations are as a rule only obtained for cases where the TM system could not produce a good-enough translation (cf. Heyn, 1996). Given that the SMT system used is usually trained only on the data available in the TM, expectedly it also has few examples from which to construct the translation, thus producing low quality output.

In this paper, we combine a TM, SMT and an automatic Sub-Tree Alignment (STA) backends in a single integrated tool. When a new sentence that needs to be translated is supplied, first a Fuzzy-Match Score (FMS – see Section 2.2) is obtained from the TM backend, together with the suggested matching sentence and its translation. For sentences that receive a reasonably high FMS, the STA backend is used to find the correspondences between the input sentence and the TM-suggested translation, marking up the parts of the input that are correctly translated by the TM. The SMT backend is then employed to obtain the final translation from the marked-up input sentence. In this way we expect to achieve a better result compared to using pure SMT.

In Section 2, we present the technical details of the design of our system, together with motivation for the particular design choices. Section 3 details the experimental setup and the data set used for the evaluation results in Section 4. We present improvements that we plan to investigate in further work in Section 5, and provide concluding remarks in Section 6.

2. System Framework

We present a system that uses a TM-match to pre-translate parts of the input sentence and guide an SMT system to the generation of a higher-quality translation.

2.1. Related Approaches

We are not aware of any published research where TM output is used to improve the performance of an SMT system in a manner similar to the system presented in this paper.

Most closely related to our approach are the systems by Biçici and Dymetman (2008) and Simard and Isabelle (2009), where the authors use the TM output to extract new phrase pairs that supplement the SMT phrase table. Such an approach, however, does not guarantee that the SMT system will select the TM-motivated phrases even if a heavy bias is applied to them.

Another related system is presented in (Smith and Clark, 2009). Here the authors use a syntax-based EBMT system to pre-translate and mark-up parts of the input sentence and then supply this marked-up input to an SMT system. This differs to our system in two ways. First, Smith and Clark use EMBT techniques to obtain partial translations of the input from the complete example base, while we are only looking at the best TM match for the given input. Second, the authors use dependency structures for EMBT matching, while we employ phrase-based structures.

2.2. Translation Memory Backend

Although the intention is to use a full-scale TM system as the translation memory backend, to have complete control over the process for this initial research we decided to build a simple prototype TM backend ourselves.

We employ a database setup using the PostgreSQL v.8.4.3[1] relational database management (RDBM) system. The segment pairs from a given TM are stored in this database and assigned unique IDs for further reference. When a new sentence is supplied for translation, the database is searched for (near) matches, using an FMS based on normalised character-level Levenshtein edit distance (Levenshtein, 1965).

Thus for each input sentence, from the database we obtain the matching segment with the highest FMS, its translation and the score itself.

2.3. Sub-Tree Alignment Backend

The system presented in this paper uses phrase-based sub-tree structural alignment (Zhechev, 2010) to discover parts of the input sentence that correspond to parts of the suggested translation extracted from the TM database. We chose this particular tool, because it can produce aligned phrase-based-tree pairs from unannotated (i.e. unparsed) data. It can also function fully automatically without the need for any training data. The only auxiliary requirement it has is for a probabilistic dictionary for the languages that are being aligned. As described later in this section, in our case this is obtained automatically from the TM data during the training of the SMT backend.

The matching between the input sentence and the TM-suggested translation is done in a three-step process. First, the plain TM match and its

[1] http://www.postgresql.org/

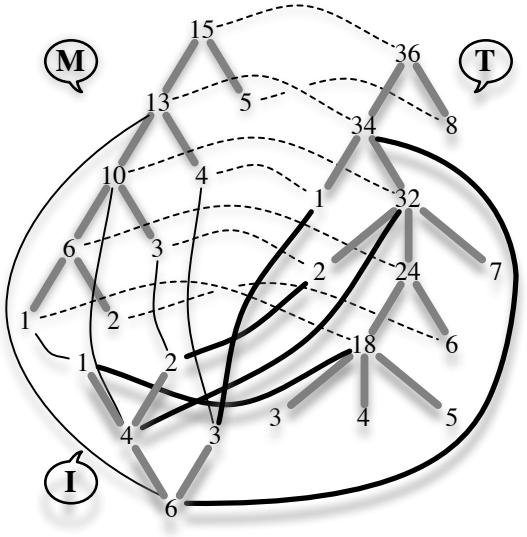
translation are aligned, which produces a sub-tree-aligned phrase-based tree pair with all non-terminal nodes labelled '*X*' (cf. Zhechev, 2010). As we are only interested in the relations between the lexical spans of the non-terminal nodes, we can safely ignore their labels. We call this first step of our algorithm *bilingual alignment*.

In the second step, called *monolingual alignment*, the phrase-based tree-annotated version of the TM match is aligned to the unannotated input sentence. The reuse of the tree structure for the TM match allows us to use it in the third step as an intermediary to establish the available sub-tree alignments between the input sentence and the translation suggested from the TM.

During this final alignment, we identify matched and mismatched portions of the input sentence and their possible translations in the TM suggestion and, thus, this step is called *matching*. Additionally, the sub-tree alignments implicitly provide us with reordering information, telling us where the portions of the input sentence that we translate should be positioned in the final translation.

The alignment process is exemplified in Figure 1. The tree marked 'I' corresponds to the input sentence, the one marked 'M' to the TM match and the one marked 'T' to the TM translation. Due to space constraints, we only display the node ID numbers of the non-terminal nodes in the phrase-structure trees — in reality all nodes carry the label '*X*'. These IDs are used to identify the sub-sentential alignment links. The lexical items corresponding to the leaves of the trees are presented in the table below the graph.

The alignment process can be visually represented as starting at a linked node in the **I** tree and following the link to the **M** tree. Then, if available, we follow the link to the **T** tree and this leads us to the **T**-tree node corresponding to the **I**-tree node we started from. In Figure 1, this results in the **I**–**T** alignments *I1–T18*, *I2–T2*, *I3–T1*, *I4–T32* and *I6–T34*. The first three links are matches, because the lexical items covered by the **I** nodes correspond exactly to the lexical items covered by their **M** node counterparts. Such alignments provide us with direct TM translations for our input. The last two links in the group are mismatched, because there is no lexical correspondence between the **I** and **M** nodes (node *I4* corresponds to the phrase *sender email*, while the linked node *M10* corresponds to *sender 's email*). Such alignments can only be used to infer reordering information. In particular in this case, we can infer that the target word order for the input sentence is *address email sender*, which produces the translation *adresse électronique de l' expéditeur*.



I	1	2	3					
input	sender	email	address					
M	1	2	3	4	5			
match	sender	's	email	address	.			
T	1	2	3	4	5	6	7	8
trans-lation	adresse	électro-nique	de	l'	expé-diteur	du	mes-sage	.

Figure 1. Example of sub-tree alignment between an input sentence, TM match and TM translation

We decided to use sub-tree-based alignment, rather than plain word alignment (e.g. GIZA++ – Och and Ney, 2003), due to a number of factors. First, sub-tree-based alignment provides much better handling of long-distance reorderings, while word– and phrase-based alignment models always have a fixed limit on reordering distance that tends to be relatively low to allow efficient computation.

The alignments produced by a sub-tree alignment model are also precision-oriented, rather than recall-oriented (cf. Tinsley, 2010). This is important in our case, where we want to only extract those parts of the translation suggested by the TM for which we are most certain that they are good translations.

As stated earlier, the only resource necessary for the operation of this system is a probabilistic bilingual dictionary covering the data that needs to be aligned. For the *bilingual alignment* step, such a bilingual dictionary is produced as a by-product of the training of the SMT backend and therefore available. For the *monolingual alignment* step, the required probabilistic dictionary is generated by simply listing each unique token seen in the source-language data in the TM as translating only as itself with probability 1.

2.4. Statistical Machine Translation Backend

Once the *matching* step is completed, we have identified and marked-up the parts of the input sentence for which translations will be extracted from the TM suggestions, as well as the parts that need to be translated from scratch. The lengths of the non-translated segments vary depending on the FMS, but are in general relatively short (one to three tokens).

The further processing of the input relies on a specific feature of the SMT backend we use, namely the Moses system (Koehn et al., 2007). We decided to use this particular system as it is the most widely adopted open-source SMT system, both for academic and commercial purposes. In this approach, we annotate the segments of the input sentence for which translations have been found from the TM suggestion using XML tags with the translation corresponding to each segment given as an attribute to the encapsulating XML tag, similarly to the system described in (Smith and Clark, 2009). The SMT backend is supplied with marked-up input in the form of a string consisting of the concatenation of the XML-enclosed translated segments and the plain non-translated segments in the target-language word order, as established by the alignment process. The SMT backend is instructed to translate this input, while keeping the translations supplied via the XML annotation. This allows the SMT backend to produce translations informed by and conforming to actual examples from the TM, which should result in improvements in translation quality.

2.5. Auxilliary Tools

It must be noted that in general the SMT backend sees the data it needs to translate in the target-language word order (e.g. it is asked to translate an English sentence that has French word order). This, however, does not correspond to the data found in the TM, which we use for deriving the SMT models. Because of this discrepancy, we developed a pre-processing tool that goes over the TM data performing *bilingual alignment* and outputting reordered versions of the sentences it processes by using the information implicitly encoded in the sub-tree alignments. In this way we obtain the necessary reordered data to train a translation model where the source language already has the target-language word order. In our system we than use this model — together with the proper-word-order model — for translation.

One specific aspect of real-world TM data that we need to deal with is that they often contain meta-tag annotations of various sorts. Namely, annotation tags specific to the file format used for storing the TM data, XML tags annotating parts of the text as appearing in Graphical User Interface (GUI) elements, formatting tags specific to the file format the TM data was originally taken from, e.g. RTF, OpenDoc, etc. Letting any MT system try to deal with these tags in a probabilistic manner can easily result in ill-formed, mistranslated and/or out-of-order meta-tags in the translation.

This motivates the implementation of a rudimentary handling of meta-tags in the system presented in this paper, in particular handling the XML tags found in the TM data we work with, as described in Section 3. The tool we developed for this purpose simply builds a map of all unique XML tags per language and replaces them in the data with short placeholders that are designed in such a way that they would not interfere with the rest of the TM data.[2] A special case that the tool has to take care of is when an XML tag contains an attribute whose value needs to be translated. In such situations, we decided to not perform any processing, but rather leave the XML tag as is, so that all text may be translated as needed. A complete treatment of meta-tags, however, is beyond the scope of the current paper.

[2] In the current implementation, the XML tags are replaced with the string <tag_id>, where <tag_id> is a unique numeric identifier for the XML tag that is being replaced.

We also had to build a dedicated tokeniser/detokeniser pair to handle real world TM data containing meta-tags, e-mail addresses, file paths, etc., as described in Section 3. Both tools are implemented as a cascade of regular expression substitutions in Perl.

Finally, we use a tool to extract the textual data from the TM. That is, we strip all tags specific to the format in which the TM is stored, as they can in general be recreated and thus do not need to be present during translation. In our particular case the TM is stored in the XML-based TMX format.[3]

2.6. Complete Workflow

Besides the components described above, we also performed two further transformations on the data. First, we lowercase the TM data before using it to train the SMT backend models. This also means that the alignment steps from Section 2.3 are performed on lowercased data, as the bilingual dictionary used there is obtained during the SMT training process.[4]

Additionally, the SMT and sub-tree alignment systems that we use cannot handle certain characters, which we need to mask in the data. For the SMT backend, this includes '|', '<' and '>' and for the sub-tree aligner, '(' and ')'. The reason these characters cannot be handled is that the SMT system uses '|' internally to separate data fields in the trained models and '<' and '>' cannot be handled whilst using XML tags to annotate pre-translated portions of the input. The sub-tree aligner uses '(' and ')' to represent the phrase-based tree structures it generates and the presence of these characters in the data may create ambiguity when parsing the tree structures. All these characters are masked by substituting in high-Unicode counterparts, namely '|', '<', '>', '(' and ')'. Visually, there is a very slight distinction and this is intentionally so to simplify debugging. However, the fact that the character codes are different alleviates the problems discussed above. Of course, in the final output, the masking is reversed and the translation contains the regular versions of the characters.

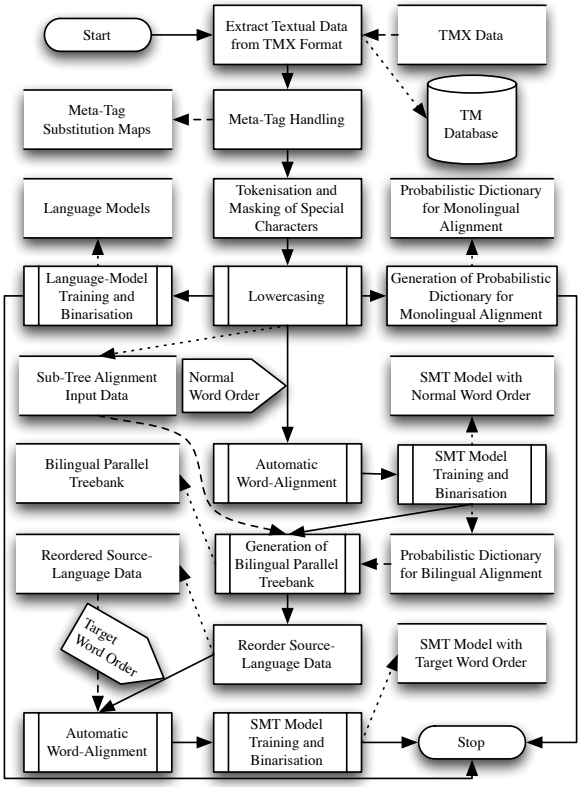


Figure 2. Pre-Processing Workflow

The complete pre-processing workflow is presented in Figure 2, where the rectangles with vertical bars represent the use of open-source tools, while the plain rectangles represent tools developed by the authors of this paper.

First, the textual data is extracted from the original TM format, producing one plain-text file for each language side. These data can either be pre-loaded in a PostgreSQL database at this time, or during the first run of the translation system.

Next, the meta-tag-handling tool is used to generate the substitution tables for the source and target languages, as well as new files for each language with the tags substituted by the corresponding identifiers (cf. Section 2.5). These files are then tokenised, lowercased and all conflicting characters are masked, as described above.

The pre-processed files are then used to produce a file containing pairs of sentences in the input format of the sub-tree aligner, as well as to generate the probabilistic dictionary required for

[3] http://www.lisa.org/fileadmin/standards/tmx1.4/tmx.htm

[4] Currently, we do not use a recaser tool and the translations produced are always in lowercase. This component, however, will be added in a future version of the system.

the *monolingual alignment* and to train the SMT model on the data in the proper word order. The SMT training produces the necessary bilingual dictionary for use by the sub-tree aligner, which is run to obtain a parallel-treebank version of the TM data. The parallel treebank is then used to retrieve bilingual alignments for the TM data, rather than generate them on the fly during translation. This is an important design decision, as the complexity of the alignment algorithm is high for plain-text alignment (cf. Zhechev, 2010).

Once we have generated the bilingual parallel treebank, we run the reordering tool, which generates a new plain-text file for the source language, where the sentences are modified to conform to the target-language word order, as implied by the data in the parallel treebank. This is then matched with the proper-order target-language file to train the SMT backend for the actual use in the translation process.

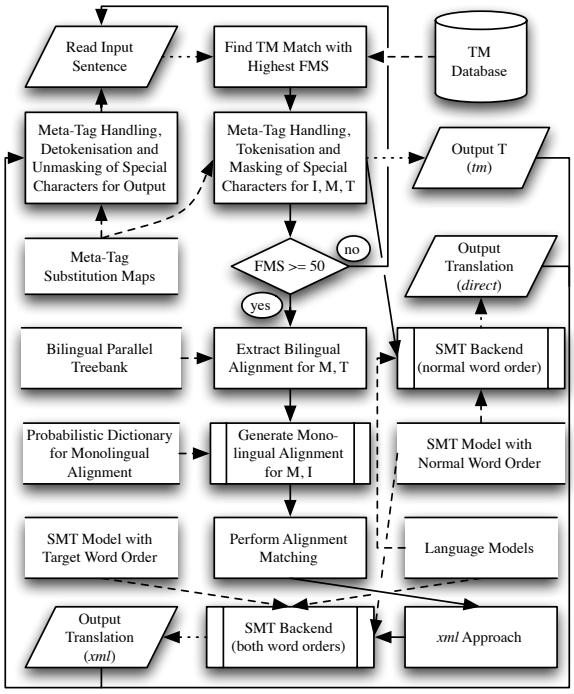


Figure 3. Translation Workflow

Once all the necessary files have been generated and all pre-processing steps have been completed, the system is ready for use for translation. The translation workflow is shown in Figure 3, 'I', 'M' and 'T' having the same meanings as in Figure 1. Here, the first step after an input sentence has been read in is to find the TM match with the highest FMS. This is done using the original plain non-pre-processed data to simulate real-life operation with a proper TM backend.

After the best TM match and its translation are extracted from the TM, they — together with the input sentence — are pre-processed by tokenisation, lowercasing, meta-tag and special-character substitution. Next, the corresponding tree pair is extracted from the bilingual parallel treebank to establish the tree structure for the TM source-language match. This tree structure is then used to perform the *monolingual alignment*, which allows us to perform the *matching* step next. After the *matching* is complete, we generate a final translation as described in Section 2.4. Finally, the translations are de-tokenised and the XML tags and special characters are unmasked.

3. Experimental Setup

We use real-life TM data from an industrial partner. The TM was generated during the translation of RTF-formatted customer support documentation. The data is in TMX format and originally contains 108 967 English–French translation segments, out of which 14 segments either have an empty language side or have an extreme discrepancy in the number of tokens for each language side and were therefore discarded.

A particular real-life trait of the data is the presence of a large number of XML tags. Running the tag-mapping tool described in Section 2.6, we gathered 2 049 distinct tags for the English side of the data and 2 653 for the French side. Still, there were certain XML tags that included a `label` argument whose value was translated from one language to the other. These XML tags were left intact so that our system could handle the translation correctly.

The TM data also contain a large number of file paths, e-mail addresses, URLs and others, which makes bespoke tokenisation of the data necessary. Our tokenisation tool ensures that none of these elements are tokenised, keeps RTF formatting sequences non-tokenised and properly handles non-masked XML tags, minimising their fragmentation.

As translation segments rarely occur more than once in a TM, we observe a high number of unique tokens (measured after pre-processing) — 41 379 for English and 49 971 for French — out of

108 953 segment pairs. The average sentence length is 13.2 for English and 15.0 for French.

For evaluation, we use a data set of 4977 English–French segments from the domain of the TM. The sentences in the test set are significantly shorter on average, compared to the TM — 9.2 tokens for English and 10.9 for French.

It must be noted that we used SMT models with maximum phrase length of 3 tokens, rather than the standard 5 tokens, and for decoding we used a 3-gram language model. This results in much smaller models than the ones usually used in mainstream SMT applications. (The standard for some tools goes as far as 7-token phase-length limit and 7-gram language models)

4. Evaluation Results

For the evaluation of our system, we used a number of widely accepted automatic metrics, namely BLEU (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005), TER (Snover et al., 2006) and inverse F-Score based on token-level precision and recall.

We setup our system to only fully process input sentences for which a TM match with an FMS over 50% was found, although all sentences were translated directly using the SMT backend to check the overall pure SMT performance. The TM-suggested translations were also output for all input sentences.

The results of the evaluation are given in Figure 4, where the *tm* and *direct* scores are also given for the FMS range [0%; 50%)∪{100%}. Across all metrics we see a uniform drop in the quality of TM-suggested translations, which is what we expected, given that these translations contain one or more wrong words. We believe that the relatively high scores recorded for the TM-suggested translations at the high end of the FMS scale are a result of the otherwise perfect word order and lexical choice. For *n*-gram-match-based metrics like the ones we used such a result is expected and predictable. Although the inverse F-score results show the potential of our setup to translate the outstanding tokens in a 90%–100% TM match, it appears that the SMT system produces word order that does not correspond to the reference translation and because of this receives lower scores on the other metrics.

The unexpected drop in scores for perfect TM matches is due to discrepancies between the reference translations in our test set and the translations stored in the TM. We believe that this issue

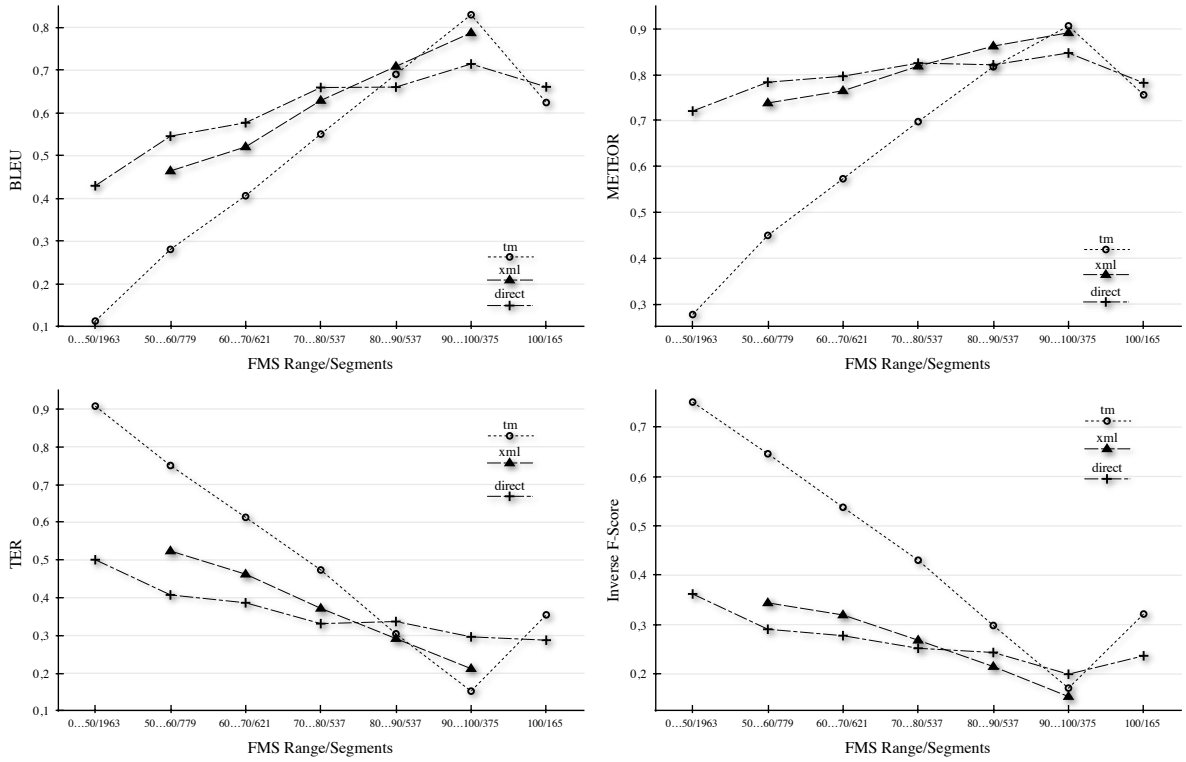


Figure 4. Evaluation results for English-to-French translation, broken down by FMS range

affects all FMS ranges, albeit to a lower extent for non-perfect matches. Unfortunately, the exact impact cannot be ascertained without human evaluation.

We observe a significant drop-off in translation quality for the *direct* output below FMS 50%. This suggests that sentences with such low FMS should be translated either by a human translator from scratch, or by an SMT system trained on different/more data.

Our system (i.e. the *xml* setup) clearly outperforms the *direct* SMT translation for FMS between 80 and 100 and has comparable performance between FMS 70 and 80. Below FMS 70, the SMT backend has the best performance. Although these results are positive, we still need to investigate why our system has poor performance at lower FMS ranges. Theoretically, it should outperform the SMT backend across all ranges, as its output is generated by supplying the SMT backend with good pre-translated fragments. The Inverse F-Score graph suggest that this is due to worse lexical choice, but only manual evaluation can provide us with clues for solving the issue.

The discrepancy in the results in the Inverse F-Score graph with the other metrics suggest that the biggest problem for our system is producing output in the expected word-order.

5. Future Work

There are a number of possible directions for improvement that can be explored.

As mentioned earlier, we plan to integrate our system with a full-featured open-source or commercial TM product that will supply the TM matches and translations. We expect this to improve our results, as the quality of the TM matches will better correspond to the reported FMS.

Such an integration will also be the first necessary step to perform a user study evaluating the effect of the use of our system on post-editing speeds. We expect the findings of such a study to show a significant increase of throughput that will significantly reduce the costs of translation for large-scale projects.

It would be interesting to also conduct a user study where our system is used to additionally mark up the segments that need to be edited in the final SMT translation. We expect this to provide additional speedup to the post-editing process. Such a study will require tight integration between our system and a CAT tool and the modular design we presented will facilitate this significantly.

The proposed treatment of meta-tags is currently very rudimentary and may be extended with additional features and to handle additional types of tags. The design of our system currently allows the meta-tag-handling tool to be developed independently, thus giving the user the choice of using a different meta-tag tool for each type of data they work with.

In addition, the reordering tool needs to be developed further, with emphasis on properly handling situations where the appropriate position of an input-sentence segment cannot be reliably established. In general, further research is needed into the reordering errors introduced by the SMT system into otherwise good translations.

6. Conclusions

In this paper, we presented a novel modular approach to the utilisation of Translation Memory data to improve the quality of Statistical Machine Translation.

The system we developed uses precise subtree-based alignments to reliably determine and mark up correspondences between an input sentence and a TM-suggested translation, which ensures the utilisation of the high-quality translation data stored in the TM database. An SMT backend then translates the marked-up input sentence to produce a final translation with improved quality.

Our evaluation shows that the system presented in this paper significantly improves the quality of SMT output when using TM matches with FMS above 80 and produces results on par with the pure SMT output for SMT between 70 and 80. TM matches with FMS under 70 seem to provide insufficient reordering information and too few matches to improve on the SMT output. Still, further investigation is needed to properly diagnose the drop in quality for FMS below 70.

We expect further improvements to the reordering functionality of our system to result in higher-quality output even for lower FMS ranges.

Acknowledgements

This research is funded under the 7th Framework Programme of the European Commission within the EuroMatrixPlus project (grant № 231720). The data used for evaluation was generously provided by Symantec Ireland.

References

Banerjee, Satanjeev and Alon Lavie. 2005. METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgements. In *Proceedings of the Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization at the 43rd Annual Meeting of the Association for Computational Linguistics (ACL '05)*, pp. 65–72. Ann Arbor, MI.

Biçici, Ergun and Marc Dymetman. 2008. Dynamic Translation Memory: Using Statistical Machine Translation to improve Translation Memory Fuzzy Matches. In *Proceedings of the 9th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing '08)*, ed. Alexander F. Gelbukh, pp. 454–465. Vol. 4919 of *Lecture Notes in Computer Science*. Haifa, Israel: Springer Verlag.

Heyn, Matthias. 1996. Integrating Machine Translation into Translation Memory Systems. In *Proceedings of the EAMT Machine Translation Workshop, TKE '96*, pp. 113–126. Vienna, Austria.

Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the Demo and Poster Sessions of the 45th Annual Meeting of the Association for Computational Linguistics (ACL '07)*, pp. 177–180. Prague, Czech Republic.

Levenshtein, Vladimir I. 1965. Двоичные коды с исправлением выпадений, вставок и замещений символов (Binary Codes Capable of Correcting Deletions, Insertions, and Reversals). *Доклады Академий Наук СССР*, 163 (4): 845–848. [reprinted in: Soviet Physics Doklady, 10: 707–710.].

Och, Franz Josef and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29 (1): 19–51.

Papineni, Kishore, 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 of Computational Linguistics (ACL '02)*, pp. 311–318. Philadelphia, PA.

Simard, Michel and Pierre Isabelle. 2009. Phrase-based Machine Translation in a Computer-assisted Translation Environment. In *The Twelfth Machine Translation Summit (MT Summit XII)*, pp. 120–127. Ottawa, ON, Canada.

Smith, James and Stephen Clark. 2009. EBMT for SMT: A New EBMT–SMT Hybrid. In *Proceedings of the 3rd International Workshop on Example-Based Machine Translation (EBMT '09)*, eds. Mikel L. Forcada and Andy Way, pp. 3–10. Dublin, Ireland.

Snover, Matthew, Bonnie J. Dorr, Richard Schwartz, Linnea Micciulla and John Makhoul. 2006. A Study of Translation Edit Rate with Targeted Human Annotation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas (AMTA '06)*, pp. 223–231. Cambridge, MA.

Tinsley, John. 2010. Resourcing Machine Translation with Parallel Treebanks. School of Computing, Dublin City Univercity: PhD Thesis. Dublin, Ireland.

Zhechev, Ventsislav. 2010. Automatic Generation of Parallel Treebanks. An Efficient Unsupervised System: Lambert Academic Publishing.