Assessing linguistically aware fuzzy matching in translation memories

Tom Vanallemeersch, Vincent Vandeghinste
Centre for Computational Linguistics, University of Leuven
Blijde Inkomststraat 13
B-3000 Leuven, Belgium
{tom,vincent}@ccl.kuleuven.be

Abstract

The concept of fuzzy matching in translation memories can take place using linguistically aware or unaware methods, or a combination of both.

We designed a flexible and time-efficient framework which applies and combines linguistically unaware or aware metrics in the source and target language.

We measure the correlation of fuzzy matching metric scores with the evaluation score of the suggested translation to find out how well the usefulness of a suggestion can be predicted, and we measure the difference in recall between fuzzy matching metrics by looking at the improvements in mean TER as the match score decreases. We found that combinations of fuzzy matching metrics outperform single metrics and that the best-scoring combination is a non-linear combination of the different metrics we have tested.

1 Introduction

Computer-aided translation (CAT) has become an essential aspect of translators' working environments. CAT tools speed up translation work, create more consistent translations, and reduce repetitiveness of the translation work. One of the core components of a CAT tool is the translation memory system (TMS). It contains a database of already translated fragments, called the translation memory (TM), which consists of translation units: segments of a text together with their translation.

© 2015 The authors. This article is licensed under a Creative Commons 3.0 licence, no derivative works, attribution, CC-BY-ND.

Given a sentence to be translated, the traditional TMS looks for source language sentences in a TM which are identical (exact matches) or highly similar (fuzzy matches), and, upon success, suggests the translation of the matching sentence to the translator (Sikes, 2007).

Formally, a TM consists of a set of source sentences $S_1, ..., S_n$ and target sentences $T_1, ..., T_n$, where (S_i, T_i) form a translation unit. Let us call the sentence that we want to translate Q (the query sentence).

The TMS checks whether Q already occurs in the TM, i.e. whether $\exists S_i \in S_1, ..., S_n : Q = S_i$. If this is the case, Q needs no new translation and the translation T_i can be retrieved and used as the translation of Q. This is an exact match. If the TMS cannot find a perfect match, fuzzy matching is applied using some function Sim, which calculates the best match S_b of Q in the TM, i.e. the most similar match, as in (1):

$$S_b = \max_{S_i} Sim(Q, S_i) \quad (1)$$

If $Sim(Q, S_b) >= \theta$ (a predefined minimal threshold, which is typically 0.7 in CAT tools)[1]), T_b, the translation of S_b, is retrieved from the TM and provided as a suggestion for translating Q. If the threshold is not reached, the TMS assumes that T_b is of no value for the translator and does not provide it as a translation suggestion.

Similarity calculation can be done in many ways. In current TMS systems, fuzzy matching techniques mainly consider sentences as simple sequences of words and contain very limited linguistic knowledge. The latter is for instance present in

[1] In CAT tool interfaces, this is usually expressed as a percentage, 70%, which may be modified by the user. Developers may determine the threshold empirically, but their use of a 70% threshold may also be a matter of convention.

the form of stop word lists. Few tools use more elaborate linguistic knowledge.[2]

2 Related work

There is a large variety of methods that can be used for comparing sentences to each other. They are designed for comparing any pair of sequences or trees (not necessarily sentences and their parse trees), for fuzzy matching in a TM, or for comparing machine translation (MT) output to a reference translation. As pointed out by Simard and Fujita (2012), the third type, MT automatic evaluation metrics, can also be used in the context of TMs, both as fuzzy matching metric and as metric for comparing the translation of a fuzzy match with the desired translation. Some matching methods specifically support the integration of fuzzy matches within an MT system (example-based or statistical MT); see for instance Aramaki et al. (2005), Smith and Clark (2009), Zhechev and van Genabith (2010), Ma et al. (2011).

Some matching methods are linguistically unaware. Levenshtein distance (Levenshtein, 1966), which calculates the effort needed to convert one sequence into another using the operations insertion, deletion and substitution, is the most commonly used fuzzy matching method (Bloodgood and Strauss, 2014). Tree edit distance (Klein, 1998) applies this principle to trees; another tree comparison method is tree alignment (Jiang et al., 1995).[3]

To allow using string-based matching methods on trees, there are several ways of converting trees into strings without information loss, as described in Li et al. (2008), who applies a method designed by Prüfer (1918) and based on post-order tree traversal.

Examples of matching methods specifically designed for fuzzy matching are percent match and ngram precision (Bloodgood and Strauss, 2014), which act on unigrams and longer ngrams. Baldwin (2010) compares bag-of-words fuzzy matching metrics with order-sensitive metrics, and word-based with character-based metrics. Examples of well-known MT evaluation metrics are BLEU (Papineni et al., 2002) and TER, i.e. Translation Error Rate[4] (Snover et al., 2006).

Linguistically aware matching methods make use of several layers of information. The "subtree metric" of Liu and Gildea (2005) compares subtrees of phrase structure trees. We devised a similar method, *shared partial subtree matching*, described in Section 3.1.2. Matching can also involve dependency structures, as in the approach of Smith and Clark (2009), head word chains (Liu and Gildea, 2005), semantic roles, as in the HMEANT metric (Lo and Wu, 2011), and semantically similar words or paraphrases, as in the MT evaluation metric Meteor (Denkowski and Lavie, 2014). The latter aligns MT output to one or more reference translations, not only by comparing word forms, but also through shallow linguistic knowledge, i.e. by calculating the stem of words (in some cases using language-specific rules), and by using lists with function words, synonyms and paraphrases.

Some MT evaluation metrics, such as VERTa (Comelles et al., 2014) and LAYERED (Gautam and Bhattacharyya, 2014), and some fuzzy matching methods, like the one of Gupta et al. (2014), are based on multiple linguistic layers. The layers are assigned weights or combined using a support vector machine.

Different types of metrics can be combined in order to join their strengths. For instance, the Asiya toolkit (Giménez and Márquez, 2010) contains a large number of matching metrics of different origins and applies them for MT evaluation. An optimal metric set is determined by progressively adding metrics to the set if that increases the quality of the translation.

3 Experimental setup

3.1 Independent variables

In Sections 3.1.1, 3.1.2, and 3.1.3, we describe the independent variables of our experiment.

3.1.1 Linguistically unaware metrics

Levenshtein (baseline) Given the Levenshtein distance $\Delta_{LEV}(S, T_i)$, we define Levenshtein score (i.e. similarity) as in (2):

$$Sim_{LEV}(Q, S_i) = 1 - \frac{\Delta_{LEV}(Q, S_i)}{\max(|Q|, |S_i|)} \quad (2)$$

[2] One example of such a tool is *Similis* (http://www.similis.org), which determines constituents in sentences and allows to retrieve (S_i, T_i) when S_i shares constituents with Q.

[3] We implemented the Tree Edit Distance algorithm of Klein from its description in Bille (2005), as well as the Tree Alignment Distance algorithm. However, both were too slow to be useful for parse trees unless severe optimization takes place.

[4] While the developers of TER call it *Translation Edit Rate*, the name *Translation Error Rate* is often used, through the influence of the metric name *Word Error Rate*, which is used in automatic speech recognition.

Levenshtein distance, which is based on three operation types (insertion, deletion and substitution), and its variants, assign a specific cost to each type of operation. Typically, each type has a cost of 1. Certain costs may be changed in order to obtain a specific behaviour. For instance, the cost of a substitution may depend on the similarity of words.

Translation Error Rate Given a sentence Q output by an MT system and a reference translation R, TER keeps on applying shifts to Q as long as $\Delta_{LEV}(Q,R)$ keeps decreasing.[5] The TER distance $\Delta_{TER}(Q,R)$, which equals $\Delta_{LEV}(Q,R)$ plus the cost of the shifts, is normalized as in (3):

$$Score_{TER}(Q,R) = \frac{\Delta_{TER}(Q,R)}{|R|} \quad (3)$$

We convert $Score_{TER}$ into a similarity score between 0 and 1 as in (4). This formula assumes a very high upper bound for $Score_{TER}$.[6]

$$Sim_{TER}(Q,R) = 1 - \frac{log(1 + Score_{TER}(Q,R))}{3} \quad (4)$$

Percent match calculates the percent of unigrams in Q that are found in S_i, as in (5):[7]

$$Sim_{PM}(Q,S_i) = \frac{|Q_{1grams} \cap S_{i,1grams}|}{|Q_{1grams}|} \quad (5)$$

Ngram precision compares $n$grams, i.e. subsequences of one or more elements, of length 1 up till N, as in (6), where the precision for $n$grams of length n is calculated as in (7).

$$Sim_{NGP}(Q,S_i) = \sum_{n=1}^{N} \frac{1}{N} p_n \quad (6)$$

$$p_n = \frac{|Q_{ngrams} \cap S_{i,ngrams}|}{Z * |Q_{ngrams}| + (1 - Z) * |S_{i,ngrams}|} \quad (7)$$

[5] An implementation of TER can be found here: http://www.cs.umd.edu/~snover/tercom. We used version 0.7.25 for our experiment.

[6] Setting the denominator to 3 ensures that Sim_{TER} is a non-negative number unless $Score_{TER}$ exceeds the upper bound of 19. We chose this arbitrary bound in order to have an integer as denominator in the formula.

[7] This metric is similar to the metric PER, *position-independent word error rate* (Tillmann et al., 1997). The difference between both metrics lies in the fact that PER takes account of multiple occurrences of a token in a sentence, does not calculate a normalized value between 0 and 1, and does not ignore words which are present in S_i but not in Q.

Q_{ngrams} is the set of $n$grams in Q. $S_{i,ngrams}$ is the set of $n$grams in S_i, and Z is a parameter to control normalization. Setting Z to a high value prefers longer translations.[8]

Bloodgood and Strauss propose weighted variants for the Sim_{PM} and Sim_{NGP} metrics, using IDF weights, which reflect the relevance of the matching words. We will refer to one of these variants later on, calling it Sim_{PMIDF}.

3.1.2 Linguistically aware metrics

Adaptations of linguistically unaware metrics We investigated Levenshtein, percent match, and TER not only on sequences of word forms, but also on sequences of lemmas. We will refer to these lemma-based metrics as Sim_{LEVLEM}, Sim_{PMLEM} and Sim_{TERLEM}.

Shared partial subtree matching We devised a method which aims specifically at comparing two parse trees. In order to perform this comparison in an efficient way, we apply the following steps: (1) check whether pairs of subtrees in the two parses share a partial subtree; (2) determine the scores of the shared partial subtrees, based on lexical and non-lexical similarity of the nodes, on the relevance of the words, and on the number of nodes in the shared partial subtree; (3) perform a greedy search for the best combination of shared partial subtrees.

Based on the scores of the partial subtrees in the final combination, we determine the shared partial subtree similarity, as in (8). In this equation, $Score_{SPS}(Q,S_i)$ stands for the sum of the scores of the partial subtrees in the combination and $MaxScore_{SPS}(Q,S_i)$ stands for the score we obtain if Q and S_i are equal.

$$Sim_{SPS}(Q,S_i) = \frac{Score_{SPS}(Q,S_i)}{MaxScore_{SPS}(Q,S_i)} \quad (8)$$

Levenshtein for Prüfer sequences Extracting information from a tree and gathering it into a sequence allows us to apply string-based methods, which are less time-costly than tree-based methods, and which come in a great variety.

When comparing the structures in two Prüfer sequences, we may use either a cost of 0 (identity of structures) or 1. However, some structures which

[8] For our experiment, we set N to 4 and Z to 0.5. Setting its value experimentally, however, would be more appropriate.

are not identical may have some degree of similarity (for instance, a terminal node with equal part-of-speech but different lemmas). Therefore, we assign costs between 0 and 1 when calculating the Levenshtein distance. We refer to Levenshtein calculation on Prüfer sequences as $Sim_{LEVPRFC}$.

Ngram precision for head word chains Head word chains can be considered as ngrams. Therefore, we apply a variant of ngram precision to them which we call Sim_{NGPHWC}.

Meteor For brevity's sake we do not provide the formulas on which Sim_{METEOR} is based. We use the standard settings including shallow linguistic knowledge and paraphrases.[9]

3.1.3 Combinations of metrics

Could a combination of matching metrics perform better than the metrics on their own? We checked this by creating regression trees.[10] The training examples provided for building the tree are the matches of sentences to translate, the features (independent variables) are matching metrics, and their values are the matching score. The regression trees model decisions for predicting the evaluation score of the translation of the match (the dependent variable) in a non-linear way. We consider the predicted evaluation score as a new fuzzy match score.

3.2 Dependent variable

The dependent variable of our experiment is the evaluation score of the translation suggestion. We use Sim_{TER} as evaluation metric. It reflects the effort required to change a translation suggestion into the desired translation. It should be noted that the usefulness of a translation suggestion should ultimately be determined by a translator working with a CAT tool. However, human evaluation is time-consuming. We therefore use an automatic evaluation metric as a proxy for human evaluation, similarly to the modus operandi in the development of MT systems.

In order to assess the usefulness of an indvidual or combined fuzzy matching metric, we apply a leave-one-out test to a set of parallel sentences and investigate how well each metric correlates with the evaluation score. For each $Q_i \in Q_1, \ldots, Q_n$, we select the best match produced by the metric, which we call $S_{b,i}$. We call its match score $M_{b,i}$ and its translation in the TM $T_{b,i}$. We call Q_i's translation R_i. The evaluation score of the translation is $E_{b,i} = Sim_{TER}(T_{b,i}, R_i)$. We compute the Pearson correlation coefficient between M and E. A higher coefficient indicates a more useful fuzzy matching metric.

A second way for assessing the usefulness of metrics is considering their mean evaluation score and investigating the significance of the difference between metrics through bootstrap resampling. This approach consists of taking a large number of subsets of test sentences and comparing the mean evaluation score of their best matches across metrics. For instance, if one metric has a higher mean in at least 95% of the subsets than another one, the first metric is significantly better than the second one at confidence level 0.05.

A third way we study the usefulness of metrics is by investigating the degree to which the mean evaluation score decreases as we keep adding sentences with diminishing match score. If the decrease in mean evaluation score is slower in one metric than in another, the first metric has a higher recall than the second metric, as we need to put less effort in editing the translation suggestions to reach the desired translation.

3.3 Speed of retrieval

We developed a filter called *approximate query coverage* (AQC). Its purpose is to select candidate sentences in the TM which are likely to reach a minimal matching threshold when submitting them to a fuzzy matching metric, in order to increase the speed of matching. A candidate sentence is a sentence which shares one or more $n$grams of a minimal length N with Q, and which shares enough $n$grams with Q so as to cover the latter sufficiently.

The implementation of the filter uses a suffix array (Manber and Myers, 1993), which allows for a very efficient search for sentences sharing $n$grams with Q.[11] This approach is similar to the one used in the context of fuzzy matching by Koehn and Senellart (2010).

In order to measure the usefulness of the AQC filter, we measured the tradeoff between the gain

[9] We use version 1.5 of Meteor. See http://www.cs.cmu.edu/~alavie/METEOR.

[10] We used complexity parameter 0.001, retained 500 competitor splits in the output and applied 100 cross-validations.

[11] We used the SALM toolkit (Zhang and Vogel, 2006) for building and consulting suffix arrays in our experiment.

in speed and the loss of potentially useful matches. We used a sample of about 30,000 English-Dutch sentence pairs selected from Europarl (Koehn, 2005), and a threshold of 0.2. After applying a leave-one-out test, which consists of considering each S_i in the sample as a Q and comparing it to all the other S_i in the sample, it appeared that the AQC filter selected about 9 candidate sentences per Q. The gain in speed is very significant: after filtering, a fuzzy matching metric like Sim_{LEV} only needs to be applied to 0.03% of the sentences in the sample. As for the loss of potentially useful matches, we considered each S_i for which $Sim_{LEV}(Q, S_i) >= 0.3$ to be such a match. It appears that most of these S_i are still available after filtering: 93% of all pairs (Q, S_i) with a Sim_{LEV} value between 0.3 and 0.4 have an AQC score $>= 0.2$. For pairs between 0.4 and 0.6, this is 98%, and for pairs above 0.6 100%. Hence, there is a very good tradeoff between gain in speed and loss of potentially useful matches.

3.4 Preprocessing data

We use the Stanford parser (Klein and Manning, 2003) to parse English sentences. We divide a sample of sentences into two equally sized sets: a training set, from which regression trees are built, and a test set, to which individual metrics and combined metrics derived from regression trees are applied. We derive IDF weights from the full sample.

4 Results

We tested the setup described in the previous section on a sample of 30,000 English-Dutch sentence pairs from Europarl. We built regression trees for different combinations of metrics. The combined metrics either involve the baseline and an individual metric or a larger set of metrics. The results are shown in Table 1. The leftmost column shows the metric used:

- Individual metrics: LEV (Levenshtein), TER, METEOR, PM (percent match), PMIDF (percent match with weights), NGP (ngram precision), NGPHWC (head word chains), LEVLEM (lemma-based Levenshtein), PMLEM, TERLEM, SPS (shared partial subtree matching)
- Combination of baseline and individual metric: TER+LEV, SPS+LEV, ...
- Combination of all linguistically aware metrics: LING

Table 1: Comparison of metrics with baseline

Sim	$Corr(M_{b,i}, E_{b,i})$	$Score_{TER}$
Baseline		
LEV	0.278	1.007
Linguistically aware metrics		
LEVLEM	0.279	1.009
LEVPRFC	0.283	0.983
METEOR	0.058	1.066
NGPHWC	0.291	1.028
PMLEM	0.420	0.927*
SPS	0.275	0.987
TERLEM	**0.500**	**0.926***
Linguistically unaware metrics		
NGP	0.222	1.035
PM	0.424	**0.926***
PMIDF	0.335	0.963*
TER	**0.502**	**0.926***
Metrics combined using regression tree		
LEVLEM+LEV	0.362	**0.869***
LEVPRFC+LEV	0.386	0.905*
METEOR+LEV	0.391	0.910*
NGPHWC+LEV	0.347	**0.869***
PMLEM+LEV	0.478	0.916*
SPS+LEV	0.363	0.908*
TERLEM+LEV	0.562	0.894*
NGP+LEV	0.376	0.903*
PM+LEV	0.455	0.906*
PMIDF+LEV	0.405	0.906*
TER+LEV	0.561	0.894*
LING	0.564	0.899*
NONLING	**0.571**	0.889*
ALL	0.563	0.899*
		$^*p < 0.05$

- Combination of all linguistically unaware metrics (except for the baseline): NONLING
- Combination of all metrics (including the baseline): ALL

The middle column of Table 1 shows the Pearson correlation coefficient between the match score and the evaluation score (Sim_{TER}). The rightmost column shows the means of $Score_{TER}$ values (which reflect the estimated editing effort) instead of the means of the Sim_{TER} values. We used the latter primarily to facilitate the calculation of certain statistics regarding TER, such as correlations.

Let us first have a look at the individual metrics in Table 1. The Sim_{PM} and Sim_{TER} metrics, and their lemma-based variants, have the highest correlation with the evaluation score; their correlation is markedly higher than that of the baseline. Interestingly, IDF weights do not seem to help percent match, on the contrary. The correlation of most other individual metrics is close to that of the baseline. Looking at the worst-performing two metrics, Sim_{NGP} and Sim_{METEOR}, it is striking that the latter has an extremely low correla-

tion compared to the baseline. This needs further investigation. The high score of Sim_{TER} and Sim_{TERLEM} raises the question whether an evaluation metric favors a fuzzy matching metric which is identical or similar to it.

The means of the $Score_{TER}$ values for individual metrics more or less confirm the differences observed for correlation. Sim_{PM}, Sim_{TER} and their lemma-based variants have the lowest mean. As shown by the asterisks in the table, the difference in mean with the baseline is significant at the 0.05 level for about half of the individual metrics.

Looking at the combined metrics in Table 1, we see that all of them have a higher correlation with the evaluation score than the baseline, and a lower $Score_{TER}$ mean; the difference in mean with the baseline is always significant. Of all two-metric combinations, the ones involving Sim_{TER} and its lemma-based variant perform the best. The combinations Sim_{LING} and $Sim_{NONLING}$ perform slightly better than the best two-metric combinations. Sim_{ALL}, which includes the baseline itself, comes close to Sim_{LING} and $Sim_{NONLING}$ but does not exceed their performance. From the regression tree involving the combination of all metrics, it appears that it uses 9 of the 12 individual metrics, including the baseline, to predict the evaluation score. There is no clearcut association between the correlation values of the combined metrics and their $Score_{TER}$ mean. For instance, Sim_{NGPHWC} has the lowest correlation but also the lowest $Score_{TER}$ mean.

Figure 1 shows the mean $Score_{TER}$ increase (i.e. increase in editing effort) that we obtain when adding baseline matches with decreasing match score. When we order all test sentences according to the baseline score of their best match, the mean $Score_{TER}$ of the first 1000 sentences (the 1000 top sentences) is 0.74. When we order the test sentences according to Sim_{ALL}, the mean $Score_{TER}$ of the 1000 top sentences is 0.67. As we add more sentences to the top list, the $Score_{TER}$ mean for the baseline increases more strongly than that of Sim_{ALL}. The recall of Sim_{ALL} increases, as we need to put less effort in editing the translation suggestions of the top list. For instance, the recall for 1000 sentences is 10% lower for the baseline (0.74/0.67=1.10). For 2000 sentences, the difference increases to 11%, and for 3000 to 13%. The *oracle* line in Figure 1 indicates the mean $Score_{TER}$ increase in case we know the evaluation score of the best match beforehand; this is the upper bound for a matching metric.

From the results in Table 1, we can conclude that, though linguistically unaware metrics help a long way in improving on the baseline, linguistic metrics clearly have added value. A question that arises here, and to which we already pointed previously, is whether the use of an identical metric for fuzzy matching and for evaluation favors that fuzzy matching metric with respect to others. If that is the case, it may be better to optimize fuzzy matching methods towards a combination of evaluation metrics rather than a single metric. Ideally, human judgment of translation should also be involved in evaluation.

5 Conclusion and future

Our comparison of the baseline matching metric, Levenshtein distance, with linguistically aware and unaware matching metrics, has shown that the use of linguistic knowledge in the matching process provides clear added value. This is especially the case when several metrics are combined into a new metric using a regression tree. The correlation of combined metrics with the evaluation score is much stronger than the correlation of the baseline. Moreover, significant improvement is observed in terms of mean evaluation score, and the difference in recall with the baseline increases as match scores decrease.

Considering the fact that there is added value in linguistic information, we may further improve the performance of matching metrics by testing more metric configurations, by using additional metrics or metric combinations built for MT evaluation, and by building regression trees using larger training set sizes. Testing on an additional language, for instance a highly inflected one, may also shed light on the value of fuzzy metrics.

Our experiments were performed using a single evaluation metric, TER. We may also use other metrics for evaluation, such as percent match, Meteor or shared partial subtree matching, in order to assess to which degree the use of an identical metric for fuzzy matching and for evaluation affects results. In this respect, we will also investigate the low correlation between Meteor as a fuzzy matching metric and TER as an evaluation score, and select a new metric which we use for evaluation only and which applies matching techniques absent from the other metrics. An example of such a

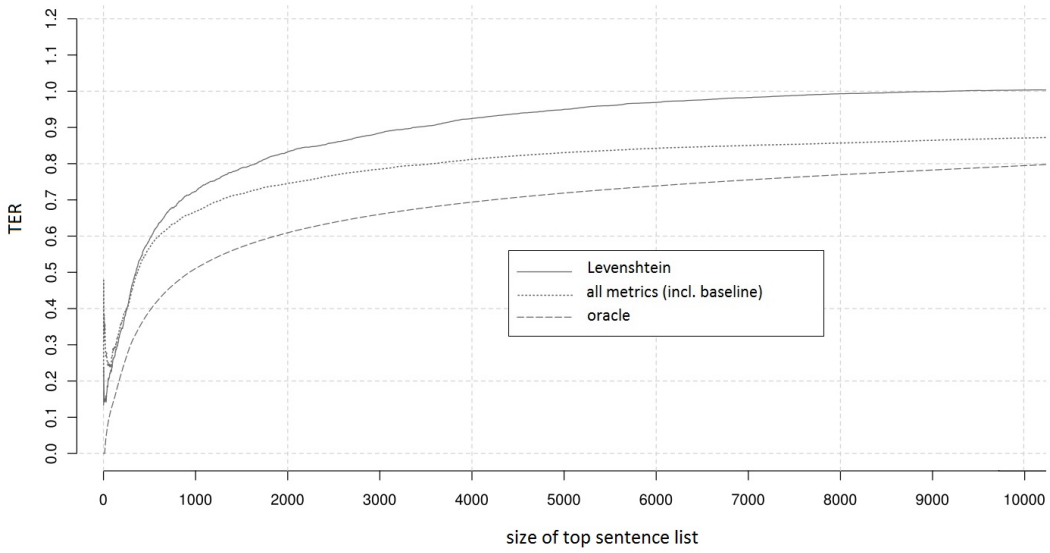

Figure 1: Mean $Score_{TER}$ increase

metric is the recently developed BEER (Stanojević and Sima'an, 2014), which is based on permutation of tree nodes. Human judgment of translation suggestions will also be taken into account.

Last but not least, we would like to point out that we have created an innovative fuzzy matching framework with powerful features: integration of matching metrics with different origins and levels of linguistic information, support for different types of structures (sequences, trees, trees converted into sequences), combination of metrics using regression trees, use of any metric in the source or target language (fuzzy matching metric or evaluation metric), and fast filtering through a suffix array.

6 Acknowledgements

This research is funded by the Flemish government agency IWT (project 130041, SCATE). See http://www.ccl.kuleuven.be/scate.

References

Aramaki, Eiji, Sadao Kurohashi, Hideki Kashioka and Naoto Kato. 2005. Probabilistic Model for Example-based Machine Translation. *Proceedings of the 10th Machine Translation Summit*, Phuket, Thailand. pp. 219–226.

Baldwin, Timothy. 2010. The Hare and the Tortoise: Speed and Accuracy in Translation Retrieval. *Machine Translation*, 23(4):195–240.

Bille, Philip. 2005. A Survey on Tree Edit Distance and Related Problems. *Theoretical Computer Science*, 337(1-3):217–239.

Bloodgood, Michael and Benjamin Strauss. 2014. Translation Memory Retrieval Methods. *Proceedings of the 14th Conference of the European Association for Computational Linguistics*, Gothenburg, Sweden. pp. 202–210.

Comelles, Elisabet, Jordi Atserias, Victoria Arranz, Irene Castellón, and Jordi Sesé. 2014. VERTa: Facing a Multilingual Experience of a Linguistically-based MT Evaluation. *Proceedings of the 9th International Conference on Language Resources and Evaluation*, Reykjavik, Iceland. pp. 2701–2707.

Denkowski, Michael and Alon Lavie. 2014. Meteor Universal: Language Specific Translation Evaluation for Any Target Language. *Proceedings of the 9th Workshop on Statistical Machine Translation*, Baltimore, Maryland, USA. pp. 376–380.

Gautam, Shubham and Pushpak Bhattacharyya. 2014. LAYERED: Metric for Machine Translation Evaluation. *Proceedings of the 9th Workshop on Statistical Machine Translation*, Baltimore, Maryland, USA. pp. 387-393.

Giménez, Jesús and Lluís Márquez. 2010. Asiya: An Open Toolkit for Automatic Machine Translation (Meta-)Evaluation. *The Prague Bulletin of Mathematical Linguistics*, 94:77–86.

Gupta, Rohit, Hanna Bechara and Constantin Orasan. 2014. Intelligent Translation Memory Matching and Retrieval Metric Exploiting Linguistic Technology. *Proceedings of Translating and the Computer 36*, London, UK. pp. 86–89.

Jiang, Tao, Lushen Wang, and Kaizhong Zhang. 1995. Alignment of Trees – An Alternative to Tree Edit. *Theoretical Computer Science*, 143(1):137-148.

Klein, Dan and Christopher Manning. 2003. Fast Exact Inference with a Factored Model for Natural Language Parsing. *Advances in Neural Information Processing Systems 15 (NIPS)*, MIT Press. pp. 3–10.

Klein, Philip. 1998. Computing the Edit Distance between Unrooted Ordered Trees. *Proceedings of the 6th Annual European Symposium on Algorithms*, Venice, Italy. pp. 91–102.

Koehn, Philipp. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. *Proceedings of the 10th Machine Translation Summit*, Phuket, Thailand. pp. 79–86.

Koehn, Philipp and Jean Senellart. 2010. Fast Approximate String Matching with Suffix Arrays and A*Parsing. *Proceedings of the 9th Conference of the Association for Machine Translation in the Americas*, Denver, Colorado. 9 pp. [http://www.mt-archive.info/AMTA-2010-Koehn.pdf]

Levenshtein, Vladimir I. 1966. Binary Codes Capable of Correcting Deletions, Insertions, and Reversals. *Soviet Physics Doklady*, 10(8):707–710.

Li, Guoliang, Xuhui Liu, Jianhua Feng, and Lizhu Zhou. 2008. Efficient Similarity Search for Tree-Structured Data. *Proceedings of the 20th International Conference on Scientific and Statistical Database Management*, Hong Kong, China. pp. 131–149.

Liu, Ding and Daniel Gildea. 2005. Syntactic Features for Evaluation of Machine Translation. *Proceedings of ACL 2005 Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, Ann Arbor, Michigan, USA. pp. 25–32.

Lo, Chi-kiu and Dekai Wu. 2011. MEANT: An Inexpensive, High-accuracy, Semi-automatic Metric for Evaluating Translation Utility via Semantic Frames. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies – Volume 1*, Portland, Oregon, USA. pp. 220–229.

Ma, Yanjun, Yifan He, Andy Way, and Josef van Genabith. 2011. Consistent Translation using Discriminative Learning: a Translation Memory-inspired Approach. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies – Volume 1*, Portland, Oregon. pp. 1239–1248.

Manber, Udi and Gene Myers. 1993. Suffix Arrays: A New Method for On-line String Searches. *SIAM Journal on Computing*, 22:935–948.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, Philadelphia, Pennsylvania, USA. pp. 311–318.

Prüfer, Heinz. 1918. Neuer Beweis eines Satzes über Permutationen. *Archiv der Mathematik und Physik*, 27:742–744.

Sikes, Richard. 2007. Fuzzy Matching in Theory and Practice. *Multilingual*, 18(6):39–43.

Simard, Michel and Atsushi Fujita. 2012. A Poor Man's Translation Memory Using Machine Translation Evaluation Metrics. *Proceedings of the 10th Conference of the Association for Machine Translation in the Americas*, San Diego, California, USA. 10 pp. [http://www.mt-archive.info/AMTA-2012-Simard.pdf]

Smith, James and Stephen Clark. 2009. EBMT for SMT: a new EBMT-SMT hybrid. *Proceedings of the 3rd International Workshop on Example-Based Machine Translation*, Dublin, Ireland. pp. 3–10.

Snover, Matthew, Bonnie Dorr, Richard Schwartz, Linnea Micciula, and John Makhoul. 2006. A Study of Translation Edit Rate with Targeted Human Annotation. *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas*, Cambridge, Massachusetts, USA. pp. 223–231.

Stanojević, Miloš and Khalil Sima'an. 2014. BEER: BEtter Evaluation as Ranking. *Proceedings of the 9th Workshop on Statistical Machine Translation*, Baltimore, Maryland, USA. pp. 414–419.

Tillmann, Christoph, Stephan Vogel, Hermann Ney, Alex Zubiaga, and Hassan Sawaf. 1997. Accelerated Dp Based Search For Statistical Translation. *Proceedings of the 5th European Conference on Speech Communication and Technology*, Rhodes, Greece. pp. 2667–2670.

Zhang, Ying and Stephan Vogel. 2006. Suffix Array and its Applications in Empirical Natural Language Processing. *Technical Report CMU-LTI-06-010*, Language Technologies Institute, School of Computer Science, Carnegie Mellon University.

Zhechev, Ventsislav and Josef van Genabith. 2010. Maximising TM Performance through Sub-Tree Alignment and SMT. *Proceedings of the 9th conference of the Association for Machine Translation in the Americas*, Denver, Colorado, USA. 10 pp. [http://www.mt-archive.info/AMTA-2010-Zhechev.pdf]