Project funded by the People Programme (Marie Curie Actions) of the European Union's Seventh Framework Programme FP7/2007-2013/ under REA grant agreement no. 317471.

Project reference: 317471

Project full title: EXPloiting Empirical appRoaches to Translation

D4.1: Language resources for matching and retrieval

Authors: Rohit Gupta (UoW)

Contributors: Hanna Bechara (UoW), Joachim Daiber (UvA), Constantin Orasan (UoW)

Document Number: EXPERT_D4.1_20150519

Distribution Level: Public

Contractual Date of Delivery: 30.10.15

Actual Date of Delivery: 19.05.15

Contributing to the Deliverable: WP4

WP Task Responsible: USAAR

EC Project Officer: Concepcion Perez-Camaras

Contents

1 Introduction

This document reports the deliverable D4.1 of the Work Package 4. We present the work on developing language resources for translation memory matching and retrieval carried out by Early Stage Researcher 4 (ESR4) under the EXPloiting Empirical appRoaches to Translation (EXPERT) project.

The concept of TM can be traced back to 1978 when Peter J. Arthern proposed the use of a translation archive (Arthern, 1978). A TM tool helps translators by retrieving previously translated segments to extract the relevant match for reuse. TMs also help in maintaining consistency with previous work with the use of appropriate terminology. In a broad perspective, Translation Memory (TM) simply is an archive of previous translations. In addition, TM also refers to a tool responsible for storing and retrieving these previous translations in an efficient manner and are commonly used by professional translators to speed up the translation process.

TM tools are the most popular tools among professional translators, reviewers and post-editors. Lagoudaki (2006) surveyed the use of TMs by professional translators in 2006, and 721 out of 874 (82.5%) replies confirmed the use of a TM. Due to this popularity, TM tools were improved over the years. The focus has been on providing a good graphical user interface to the translators, developing different filters to handle different file formats (e.g. pdf, xml, txt, html, word, xliff, subtitle etc) and, project management features and standardisation of TM storing and sharing formats. There are various exchange formats proposed for this storing and sharing purpose (TMX[1], TBX, GMX, XML:TM, XLIFF etc.). Apart from this, current TMs are also equipped with some tools like terminology managers and plugins to support machine translation from MT service providers. So far, most of the research in TM has been carried out in the industry and received less attention from academia. Although, extensive research has been done in Natural Language Processing (NLP) with emphasis on improving the performance of automatic Machine Translation (MT), there is not much research on improving the TM systems by using NLP techniques.

TM tools process the input file for translation and extract the text segments to be translated. These segments are checked against previously stored segments in a TM. In a typical TM, three types of matches are displayed to the user: in context exact matches; exact matches; and fuzzy matches. Exact matches are those matches for which 100% match is found in TM; a segment that completely matches the segment in TM. The 'in context exact matches' refers to the matches when previous and followed segments are also exact matches. Different TM vendors call *in context exact match* differently. For example Memsource[2] call it *101% match* and XTM [3] call it *ICE match*. When a TM tool is unable to find any *in context match* or *exact match* in the TM, it retrieves similar segments for post-editing in order to avoid translating from scratch. These similar segments are called fuzzy matches. However, this retrieval process is still limited to edit-distance based measures that work on surface form (or sometimes stem) matching. Most of the commercial systems use edit distance (Levenshtein, 1966) or some variation of it with some extra preprocessing. The

[1]http://www.ttt.org/oscarstandards/
[2]https://www.memsource.com
[3]http://xtm-intl.com

preprocessing typically involves tokenisation, removing punctuation, removing stop words and stemming. Although these measures provide a strong baseline, they are not sufficient to capture the semantic similarity between the segments as judged by humans. This also results in uneven post-editing time for the same fuzzy match scored segments and non-retrieval of semantically similar segments. Our work presented here tries to mitigate such problems.

The structure of this document is as follows: Section 2 provides an overview of related research in Translation Memory and related NLP techniques, Section 3 details our proposed approach to improve matching and retrieval using paraphrasing, Section 4 presents an approach to TM matching inspired from textual similarity techniques used in NLP, and Section 5 presents our conclusion.

2 Literature Survey

The last decade has shown interest in involving TM in machine translation, however mostly for the benefit of machine translation and very little has been devoted to improving translation memory. Other areas in NLP like textual entailment and semantic similarity computation between texts relate to similarity computation for TM matching and retrieval. In these areas, the ways similarities are computed substantially differs from the way similarity is calculated in TM. However, some of the techniques which work at the sentence level can be leveraged by TM matching and retrieval. The related research can be divided into three types: related NLP techniques for similarity computation; combining TM with MT; and research in improving TM matching and retrieval.

2.1 Related NLP techniques for similarity computation

Not much research in TM specifically focuses on similarity calculation for TM, but there are various areas in NLP that require calculating some kind of similarity between two texts. Different similarity measures have different characteristics, and their effectiveness depends on the type of text, length of text, domain and application. In speech recognition similarity between two string of phonemes is typically measured for error correction. In a typical information retrieval system, a query is matched against a large collection of documents, whereas in TM, query and target document have the similar length. A TM segment generally corresponds to sentence length. In machine translation evaluation, a sentence is typically evaluated against one or more references available and the final score over the test set is obtained by using the statistics obtained from all the sentences in the test set. We explore related NLP areas: semantic similarity and textual entailment; and machine translation evaluation.

2.1.1 Semantic similarity and textual entailment

In this section, we discuss some of the techniques used in NLP for measuring similarity between texts, particularly at the sentence level. We can divide these measures into three types: sentence similarity using word similarity, edit distance based approaches, and approaches combining various similarity measures.

2.1.1.1 *Sentence similarity using word similarity*

Li, McLean, Bandar, O'shea, and Crockett (2006) presented semantic similarity metric based on semantic similarity of words in a sentence. As opposed to bag-of-words approach, they also considered word order in a sentence. Word similarity and word order similarity were used to calculate sentence similarity. Word order similarity was calculated by considering index of the words appearing in a sentence as explained by the example below. For example (example taken from: Li et al. (2006)), the pair of sentence is as follows:

T1: RAM keeps things being worked with

T2: The CPU uses RAM as a short-term memory store

For the given pair of sentences, a joint word vector is formed by assigning unique words from both the sentences and preserving the order wherever possible. For the above pair of sentences T1 and T2, the joint vector T is given below:

T= {RAM keeps things being worked with The CPU uses as a short-term memory store}

Using the joint word vector with each of the sentences, corresponding word order vectors are formed respectively for each sentences. The method assigns a unique index number for each word in T1 and T2. The index numbers are nothing more than the order numbers in which the words appears in the sentence. The order vector is created using each sentence vector and joint word vector as follows, taking T1 as an example. For each word w_i in T, this method selects either the same or the most similar word above a predefined threshold in T1. If an identical or similar word is found, the index number of the corresponding word is given; otherwise 0 is given for the corresponding entry. For the above example, word order vectors for T1 and T2 are r1 and r2, respectively, given below:

$$r_1 = \{\,1\;2\;3\;4\;5\;6\;0\;3\;3\;0\;0\;0\;1\;1\,\} \qquad r_2 = \{\,4\;0\;3\;0\;0\;0\;1\;2\;3\;5\;6\;7\;8\;9\,\}$$

The word order similarity is determined by normalising the difference of word order. The word order similarity (S_r) is defined as follows:

$$S_r = 1 - \frac{||r_1 - r_2||}{||r_1 + r_2||} \tag{1}$$

The similarity measure to calculate similarity between individual words using WordNet is proposed as follows:

$$s(w1, w2) = e^{-\alpha l}\frac{e^{\beta h} - e^{-\beta h}}{e^{\beta h} + e^{-\beta h}} \tag{2}$$

where $s(w1, w2)$ is similarity between words $w1$ and $w2$, l is the shortest path length in WordNet between the synsets containing compared words $w1$ and $w2$, h is the depth of the subsumer in the hierarchical semantic nets (subsumer refers to the synset that represents hypernym of both words $w1$ and $w2$ in hierarchical semantic nets, for example, the synset of *person* is called the subsumer for words *girl* and *boy*), $\alpha \in [0, 1]$ and $\beta \in (0, 1]$ are parameters scaling the contribution of shortest path length and depth, respectively. Using the word similarity, semantic similarity (S_s) is calculated as follows: the joint vector (T) is represented as a row and the sentence (T1) is represented as a column. Each column of this matrix is filled with either the value 1 if both word are same or with the similarity for the pair of words. This matrix is used to create a lexical vector. This lexical vector is formed by taking the maximum value from each column. Furthermore, the values in the lexical vector is weighted using information gain calculated using Brown Corpus for pair of words whose score are used in the lexical vector. For the example given above, the lexical vector corresponding to T1 is as follows:

$$s_1 = \{0.390\ 0.330\ 0.179\ 0.146\ 0.239\ 0.074\ 0\ 0.082\ 0.1\ 0\ 0\ 0\ 0.263\ 0.288\}$$

Similarly, for another sentence T2, a different lexical vector is formed.

$$s_2 = \{0.390\ 0\ 0.1\ 0\ 0\ 0\ 0.023\ 0.479\ 0.285\ 0.075\ 0.043\ 0.354\ 0.267\ 0.321\}$$

Using these lexical vectors the semantic similarity is calculated as follows:

$$S_s = \frac{s_1 \cdot s_2}{||s_1|| \cdot ||s_2||} \tag{3}$$

Sentence similarity is defined as follows:

$$S(T1, T2) = \delta S_s + (1 - \delta) S_r \tag{4}$$

where $S(T1, T2)$ is the similarity between sentences $T1$ and $T2$, S_s denotes the semantic similarity calculated using WordNet by taking words occurring in sentences $T1$ and $T2$, S_r denotes word order similarity between sentences $T1$ and $T2$ and δ is a weighing parameter. For the example case above, the semantic similarity between sentences is $S_s = 0.6139$ and the order similarity $S_r = 0.2023$. By empirically determining the value of $\delta = 0.85$ we determine that the similarity between sentences is 0.5522. The approach was designed to be suitable for short texts of certain sentence length. For evaluation purposes, a dataset was created using definitions of 65 noun word pairs. 32 participants tagged this dataset for similarity ratings ranging from 0 to 4. The system performed well with 0.816 Pearson correlation coefficient against human rated dataset.

Islam and Inkpen (2008) use context and statistical information from a corpus. The authors propose a method which uses string similarity between words, semantic similarity between words and a word order similarity. The string similarity is used to capture the minute difference like misspelled words and is calculated using a modified longest common subsequence algorithm. Semantic similarity between words is calculated by a method inspired from point wise mutual information (PMI) using the British National Corpus. Word order similarity is calculated using the normalised difference of common-word order between two sentences. Word order similarity is used to incorporate syntactic information.

The overall sentence similarity is calculated as follows. Given two sentences having m and n words ($m \geq n$) with c tokens in common, the algorithm creates a joint matrix of $(m-c) \times (n-c)$ size using the string similarity matrix and semantic word similarity matrix. The string similarity and semantic word similarity matrices are created by calculating the respective similarity measures among the uncommon words between the two sentences ($(m-c)$ and $(n-c)$ words). The algorithm selects the biggest positive element (> 0) and stores it in ρ and removes the corresponding row and column elements from the matrix.

The process continues until either the matrix contains no positive element or the matrix is empty. The whole sentence similarity between sentences P and R is computed as follows:

$$S(P,R) = \frac{(c(1 - w_f + w_f S_0) + \sum_{i=1}^{|\rho|} \rho_i) \times (m+n)}{2mn}$$

In the equation above, S_0 is the common word order similarity score and w_f is the weight deciding the contribution of the common word order similarity score. The evaluation was carried out on two different datasets; Microsoft paraphrase corpus (Dolan, Quirk, & Brockett, 2004) and the dataset by Li et al. (2006). The system performed reasonably well with a Pearson correlation coefficient of 0.853 compared to 0.816 Pearson correlation coefficient obtained by Li et al. (2006) on the same dataset. On the test set from Microsoft paraphrase corpus (1725 sentence pairs), the method achieves a 81.3 F-measure.

2.1.1.2 Edit distance based approaches Tree edit distance is one of the prominent measures used to calculate similarity and textual entailment between two sentences. Basic tree edit distance signifies minimum cost of transforming the source sentence tree into the target sentence tree. Transformation is done by a sequence of edit operations on nodes and cost is calculated by summing associated cost with each of these operations. Bille (2005) surveyed various tree edit distance metrics. The standard algorithm uses dynamic programming technique (Zhang & Shasha, 1989). The basic tree edit distance calculation involves three operations: insertion of a node; deletion of a node; and substitution of a node. The three operations are given below:

1. Substitution (or Change): This operation involves relabelling a node.

2. Deletion (or Remove): This operation involves deletion of a node. All children of the removed node are assigned to the parent node.

3. Insertion: This operation involves insertion of a new node. Insertion of a node *A* under the parent node *B*, makes *A* the parent of a consecutive subsequence of the children of *B*.

Tree edit distance is extensively used in the textual entailment problem. Heilman and Smith (2010) extend the tree edit distance by adding more operations. The method implement nine operations involving edges, subtrees, siblings and root. The authors use greedy best first search to find the edit sequence which may not be the minimal edit sequence. They used Microsoft Paraphrase corpus to train a classifier using the edit sequences. They used tree edit models for three tasks: textual entailment; paraphrasing; and an answer selection task for a question answering system. The method obtained 61.8% accuracy on textual entailment task, 73.2% accuracy for paraphrase identification task and 0.6091 mean average precision and 0.6917 mean reciprocal rank for question selection task.

Recently, Alabbas and Ramsay (2013) presented a modified tree edit distance approach, which extends tree edit distance to the level of subtrees. The approach extends Zhang-Shasha's algorithm (Zhang & Shasha, 1989) by allowing costs of operations that insert, delete, and exchange subtrees. The costs of operations are derived by some appropriate function of the costs of the operations on their parts. This method was used to check entailment between two Arabic text snippets. The method got 0.636 F-score compared to 0.578 F-score using a bag-of-words approach and 0.597 F-score using Levenshtein edit distance.

M. Wang and Cer (2012) presented an approach that uses probabilistic edit-distance as a measure of semantic similarity. The approach uses probabilistic finite state automata and pushdown automata to model weighted edit-distance where state transitions correspond to edit-operations. The system obtained an overall Pearson correlation coefficient of 0.5589, which was calculated using all datasets (and models) of the task.

2.1.1.3 Combining various similarity measures

Bär, Biemann, Gurevych, and Zesch (2012) and Marsi et al. (2013) presented approach, which combine various text similarity measures. The similarity system of Bär et al. (2012) participated in the Semantic Textual Similarity (STS) task at SemEval-2012 (Agirre, Diab, Cer, & Gonzalez-Agirre, 2012). The system got an overall Pearson correlation coefficient of 0.823 on human annotated datasets used in the task. Marsi et al. (2013) extend the system by (Bär et al., 2012) with additional features. The system (Gupta, Béchara, Maarouf, & Orăsan, 2014) used in the approach presented in Section 4 is most similar to these systems.

2.1.2 Machine translation evaluation

There are several machine translation evaluation metrics. In fact, every year there are several new metrics proposed in WMT workshop (Bojar et al., 2014). Some of the most popular and widely used metrics are BLEU, METEOR, NIST, TER and WER. We used BLEU and METEOR for automatic evaluation of retrieved segments. Furthermore, the metrics which calculate a score on the basis of individual sentences can also work as a similarity metric for TM. We analyse some of the most popular metrics in MT and TM evaluations.

BLEU (Papineni, Roukos, Ward, & Zhu, 2002) is based on N-gram counts. The metric does not involve any linguistic processing. The N-grams are collected over the sentences and matched against the N-grams collected from the references.

NIST (Doddington, 2002) is similar to BLEU as it is also based on N-gram counts. In BLEU, each N-gram has the equal contribution whereas in NIST N-grams are weighted. The importance of an N-gram is decided by the use of this N-gram in the test corpus.

METEOR (Denkowski & Lavie, 2014) evaluate on the basis of phrase to phrase alignment between the given hypothesis translation and reference. Words and stems are considered as phrases of length one. In the case where multiple references are available, the hypothesis is scored against the each reference and the maximum scoring reference is used. The algorithm performs alignment on the basis of four matchers; exact words matcher, stem matcher, synonyms matcher and paraphrase matcher. The aligner constructs a search space by applying these matchers sequentially to get the all possible matches between the hypothesis and the reference. Once all possible matches are identified, the aligner obtains the largest subset of these matches fulfilling the following criteria in the order of preference given below:

1. Each word in each sentence is covered by zero or one matches

2. Largest number of words covered
3. Smallest number of chunks, where chunk is not more than a series of contiguously matched and identically ordered phrases
4. Smallest sum of absolute differences between match start positions

After getting the alignment, the precision (P) and recall (R) are calculated as follows and is used in the calculation of the Meteor score.

$$P = \frac{\Sigma_i w_i \cdot m_i(t)}{|t|} \qquad R = \frac{\Sigma_i w_i \cdot m_i(r)}{|r|}$$

In the above equations, $|t|$ is the number of words in the hypothesis translation t, $|r|$ is the number of words in the reference r. $m_i(t)$ is the number of words covered by matcher m_i for hypothesis translation t and $m_i(r)$ is the number of words covered by the matcher m_i for reference. w_i is the weight for each matcher. Using precision and recall, F_{mean} is calculated as follows:

$$F_{mean} = \frac{P \cdot R}{\alpha \cdot P + (1-\alpha) \cdot R}$$

Finally, the METEOR score is calculated as follows:

$$Score = (1 - Penalty) \cdot F_{mean} \tag{5}$$

Where, $Penalty$ is used to account for word order, is given as follow:

$$Penalty = \gamma \cdot \left(\frac{ch}{m}\right)^{\beta}$$

In the equation above, ch is the minimum number of chunks and m is the number of matched phrases.

Word error rate (WER[4]) is used as one of the basic metrics for machine translation evaluation. This metric computes scores on the basis of insertions, deletions and substitutions required divided by the number of word in the reference.

TER (Snover, Dorr, Schwartz, Micciulla, & Makhoul, 2006) metric tries to improve on WER. TER combines knowledge from multiple references and reordering of words and phrases. In TER, block movement of words called *shifts* give the same penalty as of inserting, deleting and substituting a single word.

TERp(Snover, Madnani, Dorr, & Schwartz, 2008) extends TER. TERp also includes stemming, synonyms and paraphrases. TERp uses all the edit operations of TER viz *insertions*, *deletions*, *substitutions* and *shifts* along with three additional operations viz *stem* matches, *synonym* matches and *phrase* substitutions. Stemming is implemented using the Porter stemming algorithm (Porter, 1980), synonym operation is implemented using WordNet and a paraphrase table is generated using the approach proposed by Bannard and Callison-Burch (2005). Sequences of words in the reference are searched to get the paraphrases from this table and used for phrase substitution operation.

[4]http://en.wikipedia.org/wiki/Word_error_rate

Recently, Simard and Fujita (2012) presented an interesting paper where the MT evaluation metrics are considered as TM similarity functions. The authors used English-French, English-German, English-Spanish language pairs (both ways) from three different corpora; Europarl, ECB, EMEA and JRC-Acquis Tiedemann (2009). 1000 segments from each corpus are selected for testing and rest of them are taken as the TM. Evaluation of the retrieved matches is also performed on the MT evaluation metrics. The authors find that all metrics except NIST give the matches that retrieved the best scores. Furthermore, the authors use two versions of Meteor: without paraphrases; and with paraphrases. The authors find that the matches retrieved using without paraphrases Meteor scores better most of the time over matches retrieved using with paraphrases Meteor. The authors point out that the semantic processing and matching performed on the source side English does not reflect on the target side because of the necessary resources (e.g. lack of paraphrases and unavailability of WordNet on target side) to evaluate on the target side.

2.2 Combining TM with MT

Early research in improving translation memory has focused on example based machine translation (EBMT) techniques and benefiting the translation memory where needed using EBMT. In this direction of research, Carl and Hansen (1999) concluded that linking of TM and EBMT may enhance the EBMT system. The hypothesis was that more generalisation and decomposition like EBMT system leads to better coverage, while surface form comparison like the one used by TM systems lead to better precision.

Recent techniques proposed exploit SMT in various ways. Some techniques proposed use SMT as an alternative (Simard & Isabelle, 2009; Dandapat, 2012) when no match is found in TM, some techniques (Koehn & Senellart, 2010; Tezcan & Vandeghinste, 2011) use SMT to complete the fuzzy match extracted from TM.

Simard and Isabelle (2009) used quality estimation and machine learning approaches to select the better translation between TM and SMT. The quality estimation system was trained on features like source and target segments lengths, source and target language model probabilities, language model probability ratios, machine translation probabilities from IBM model 2 and various similarity measures (Levenshtien edit-distance, Longest common subsequences etc). A SVM regression model was used to train these features and predict the quality of translation. The SMT system was also improved using some features picked from the TM technology. The SMT system re-scoring was achieved using Levenshtein edit-distance, unigram and bigram precision. The SMT system with TM features was combined with TM to get the overall system. For any input sentence the selection between TM or MT output was performed using a quality estimation system. The combined system having TM and MT with TM features was checked on three different datasets viz Europarl, Hansard (Roukos, Graff, & Melamed, 1995) and Acquis corpus. When compared to the baseline MT system, for Hansard and JRC-Acquis corpus, this approach got 1.8 and 2.6 BLEU points improvements respectively, whereas for Europarl a slight decrease of 0.3 BLEU points was observed.

Koehn and Senellart (2010) proposed an approach which completes the fuzzy matches and showed that for more than 70% fuzzy match score. This approach performs better than a single system SMT or TM. However, the focus was more on improving machine translation so testing was done on MT evaluation metric BLEU with baseline as either of MT or TM alone. Tezcan and Vandeghinste (2011) pointed out the various issues when integrating SMT and TM, mainly focussing on the XML and markup issues. They propose generalisation of XML tags before SMT training at the time of tokenisation and replacing the tags back after the translation. They tested on a TM from the automotive domain on two language pairs, English-Spanish and English-French. 912 segments were used as test set for English-Spanish, 871 segments were used as test set for English-French and 400K segments were used for training SMT system for each language pair. The approach achieved 0.7 BLEU points improvement on English-Spanish language pair and 0.88 BLEU points improvement on English-French. In our work, we do not focus on these issues because they are not relevant for our research.

There are recent trends (Bertoldi, Cettolo, & Federico, 2013; K. Wang, Zong, & Su, 2014) in adaptation of SMT systems using a TM system. Bertoldi et al. (2013) built the small local (dynamic) model using the translations by translators along with the typical static SMT model. This local translation model is updated every time the translator does a new translation. Both the local as well as the global static SMT models perform the overall translation. It was analysed that this technique can be used to improve machine translation systems if the text is very repetitive in nature. K. Wang et al. (2014) proposed an approach to dynamically incorporate TM generated segments in SMT systems. This approach adds parameters from a TM system to improve decoding in SMT as well as update the SMT model with additional phrases from TM.

2.3 Research in Improving Matching and Retrieval

Several researchers have used semantic or syntactic information in TM but the evaluations they performed were shallow and most of the time limited to subjective evaluation by authors. This makes it hard to judge how much a semantically informed TM matching system can benefit a translator. Existing research (Planas & Furuse, 1999; Hodász & Pohl, 2005; Pekar & Mitkov, 2007; Mitkov, 2008) pointed out the need for similarity calculation in TM beyond surface form comparison. Both, Planas and Furuse (1999) and Hodász and Pohl (2005) proposed using lemma and parts of speech along with surface form comparison. Planas and Furuse (1999) proposed that considering the different levels of representation of text in similarity computation is useful. The structure has eight different levels:

1. Text characters: This layer consists of all relevant characters for a text.

2. Surface Words: This layer consists of the surface forms of the words of the sentence

3. Lemmas: This layer consists of lemmatised words

4. POS: This layer consists of the part of speech tags of the words

5. XML content tags: This layer consists of XML content tags, for example, layout attribute tags used in XML segment

6. XML empty tags: these tags take care of objects inserted in the flow of text like images

7. Glossary entries: this layer consists of glossary entries

8. Linguistic analysis structures: this layer consists of "pivot schemata". The "pivot schemata" are no more than patterns composed of pivot keywords and variables with a link between two schemata; source language pattern and target language pattern. For example, "A and B" in English mapping to "A et B" in French; A-A, and-et, and B-B. Here the pivot keyword is 'and' and pivot variables are 'A' and 'B'.

Planas and Furuse (2000) proposed an algorithm that considers this structure and computes the similarity between two segments. The algorithm to compute similarity is similar to the edit distance algorithm but considers deletions and equality operations only i.e. the algorithm considers identical substitutions only; no insertions are allowed. This means algorithm does not retrieve segments where insertions are required. The edit distances are calculated at each layer. A vector is formed after taking into consideration edit distances calculated at each layer. The ranking of edit distance vectors is performed based on partial order defined on vectors. For example, assuming two layers with edit distances a_i and b_i with vector A=[a_1, b_1] and B=[a_2, b_2]. $A > B$ iff $(a_1 > a_2)$ or $(a_1 = a_2$ and $b_1 > b_2)$. This means that surface form characters are given more preference over words; words are given more preference over lemmas; and lemmas are given more preference over POS tags and so on. The authors tested a prototype model on 50 sentences from the software domain and 75 sentences from a journal with TM sizes of 7,192 sentences and 31,526 sentences respectively. The authors concluded that the approach yields more usable results compared to Trados Workbench used as a baseline. A fuzzy match retrieved was considered usable if less than half of the words required editing to get the input sentence.

Hodász and Pohl (2005) also included noun phrase (NP) detection and alignment in the matching process. The NPs are either tagged by a translator or by a heuristic NP aligner developed for English-Hungarian translation. The automatic NP detection and alignment is performed the following ways:

- Dictionary: The method searches all stems of detected NPs in an English-Hungarian dictionary. The search starts in a greedy manner with the longest NP. If the longest NP is not found, sub parts of it are searched in the dictionary. A similar procedure is applied on both sides; English and Hungarian. Finally, the alignment is done based on whether at least one token is found on the Hungarian side matching to NP detected on the English side.

- Cognate: Cognates are searched for the remaining NPs after the dictionary based searched. Cognates are considered words that are longer than one character, have at least one capital letter, number or special character; and either match or have the first four characters in common.

- POS: Parts of speech are matched when dictionary and cognate match is not possible. Matching score is kept low compared to dictionary and cognate match.

The matching score is calculated by combining the above three matches; dictionary, cognate and POS. The matching score (MS) computation is given below:

$$MS = \frac{a \cdot \text{DMW} - b \cdot \text{DNMW} + c \cdot \text{CMW} + d \cdot \text{PMW} - e \cdot \text{RFW}}{W - \text{RFW}}$$

In the above equation, DMW represents the number of dictionary matched words, DNMW represents the non NP words, CMW represents the number of match cognate words, PMW represents the number of words matched based on POS tags, PWF represents the number of function words and W is the total number words. a, b, c, d and e are weights. The authors tested on a sample sentence. They found that their system retrieves better matches compared to the baseline system which uses simple edit distance without any linguistic processing. Hodász and Pohl claim that their approach matches using simplified patterns based on linguistic analysis and hence makes it more probable to find a match in TM.

Pekar and Mitkov (2007) presented an approach based on syntax driven syntactic analysis. The segment is represented in a generalised form using this analysis. Three procedures are employed to carry out this generalisation; syntactic generalisation, lexico-syntactic generalisation and lexical generalisation. The goal of syntactic generalisation is to transform the equivalent constructions like active and passive constructions to a single representation. Syntactic generalisation is done using predefined syntactic transformation rules. Lexico-syntactic generalisation is used to account for the variability of syntactic constructions when using equivalent lexical expressions (For example, 'X likes Y' vs 'Y is appealing to X'). Lexical generalisation is used to generalise the lexical units to an equivalent class in a thesaurus. Lexical generalisation also recognise named entities mapped to an equivalent class. First-order logic representations of each predicate and its arguments found in the sentence is derived and compiled into a tree graph. The authors do not present the algorithm or framework to incorporate the generalisations or to compile the tree graph.

The authors tested the retrieval performance by using a test sentence obtained form proz.com website. The TM was artificially created for this experiment. TM contains eight sentences obtained by the top eight documents retrieved from Google search when using the test sentence as a query. Two sentences that would be the relevant matches are manually added; one with synonyms, and another with synonyms and syntactic variation. Three different similarity measures compared in this testing: (1) cosine similarity; (2) Tree edit distance with WordNet similarity between matching nodes; (3) same as (2) with additional paraphrases. In (3), nine more variants of paraphrases are used to extend the query. Two verbs available in the query sentence are replaced by three paraphrases for each verb, making nine more combinations ($3 \times 3 = 9$). Two verbs are *purchase* (paraphrase used: *buy*, *sell*, *take delivery of*) and *receive* (paraphrases used: *notify*, *consult*, *send*). The authors do not give details of how these extended paraphrases are used for retrieval and how the final similarity score is obtained.

The query segment used(Q) and top segment retrieved using all three approaches(1, 2 and 3) are given below:

Q: The company must purchase materials for release of goods, usually before any money is received from its customers.

1: It acknowledges receipt of the goods for shipment and is a certificate of ownership which must be produced before goods may be released for delivery.

2: The business has to acquire raw materials for production of commodities, typically prior to any payment is made by its clients.

3: The business has to take delivery of raw materials for production of commodities, typically before clients send any money.

The authors have found that method (1) retrieved one of the segments added using Google search document; method (2) retrieved the manually added segment which has synonyms; and method (3) retrieved the manually added segment which has synonyms and syntactic variation.

One of the things to notice is that authors extended the query with nine more variants. This is one of the major problems when using paraphrasing. Generating additional segments using paraphrasing increases the size exponentially. In our research, we propose an algorithm to efficiently handle this issue.

Recently, Utiyama, Neubig, Onishi, and Sumita (2011) have presented an approach which use paraphrasing in TM matching and retrieval. The authors proposed a method using a finite state transducer. The authors treated TM matching and retrieval procedure as a search process and they do not considered the fuzzy matching aspect. The authors acquire paraphrase list using (Bannard & Callison-Burch, 2005) approach. The method proposed by Utiyama et al. (2011) searches a best path in the composition of four weighted finite state transducers, given below:

1. InputFST: Accepts input sentence and output the same sentence
2. ParaFST: Accepts input sentence and outputs its paraphrases. This FST consists of all paraphrases and all words in the source language vocabulary
3. LMFST: Language model FST created using Kyoto language modelling toolkit[5].
4. TMFST: Accept input sentence and output its index

These four FSTs are composed and the best path is calculated to obtain the match as follows:

$BestPath(\textit{InputFST} \circ \textit{ParaFST} \circ \textit{LMFST} \circ \textit{TMFST})$

The InputFST, ParaFST, LMFST, TMFST are compiled and processed using OpenFST library[6]. Composition of all four FSTs and best path calculation is perform using Kyoto FST Decoder[7].

Texts from health-care domain products are used for testing. From this data, two sets are constructed: set1 consisting of 41,712 sentences who occurred more than once in the dataset; and set2 consisting of 90,498 sentences who occurred only once in the dataset. The set1 is used to create 266,519 paraphrase pairs. Furthermore, sentences in set2 are divided into development and test set having 44,817 and 44,975 sentences, respectively. The test set is used as input to retrieve paraphrased sentences. The method retrieved translations of 4.87% (2189) input segments. The authors created a sample of 100 sentences from these retrieved translations to check the precision. A Japanese-English translator judged the accuracy of the translations. 91 out of the 100 sentences in the sample set are evaluated to be correct, which makes precision to be 91%. The authors check against

[5]http://www.phontron.com/kylm/

[6]http://www.openfst.org/

[7]http://www.phontron.com/kyfd/

word accuracy based method considered as a baseline. The word accuracy is calculated as follows: count the number of insertions, deletions and substitutions required to turn reference into input and divide this count by the number of words in reference. The authors extracted same number (2189) of top sentences retrieved using the word accuracy method. Similarly, 100 sentences are sampled from the retrieved translations and judged by the translator. 76 out of 100 translations are found to be correct making it 76% precision. The downside of this approach is that it requires generation of all the additional segments based on paraphrases which is inefficient both in terms of time and space. Also, their approach limits TM matching to exact matches only.

Although these works present some insight into TM systems and their limitations, most of the approaches do not propose a feasible practical implementation. Also, the evaluations are shallow.

3 Improving Matching and Retrieval using Paraphrasing

As we have previously pointed that current TM systems work on the surface level with no or little linguistic information. Because of this often the paraphrased segments available in the TM are either not retrieved or retrieved with a very low threshold and are ranked incorrectly among the retrieved segments. The lack of semantic knowledge in the matching process also leads to cases where, for the same similarity score shown by the system, one segment may require little effort while another requires more in terms of post editing.

For example, even though segments like “the period laid down in article 4(3)” and “the duration set forth in article 4(3)” have the same meaning, the one segment may not be retrieved for another in current TM systems as having only 57% similarity based on word based Levenshtein edit distance as implemented in OmegaT[8]. In this case we can see that one segment is a paraphrase of the another segment. To mitigate this limitation of TM, we propose an approach to incorporating paraphrasing in TM matching without compromising the ease and flexibility of edit-distance which has been trusted by TM developers, translators and translation service providers over the years.

3.1 Our Approach

A trivial approach to implementing paraphrasing along with edit-distance is to generate all the paraphrases based on the paraphrases available and store these additional segments in the TM. This approach is highly inefficient both in terms of time and space. For example, for a TM segment which has four different phrases where each phrase can be paraphrased in five more possible ways, we get 1295 (6^4 -1) additional segments (still not considering that these phrases may contain paraphrases as well) to store in the TM, which is inefficient even for small TMs.

The basic steps for TM matching and retrieval are as follows:

step 1 Read the Translation Memories available

step 2 Read the file that needs to be translated

[8]OmegaT is an open source TM available form http://www.omegat.org

step 3 Preprocess the input file, apply filter for different file formats and identify the segments

step 4 For each segment in the input file search for the most similar segment in TM and retrieve the most similar segment if above a predefined threshold

step 5 For each segment in the input file display the input segment along with the most similar segment to the translator for post-editing

There are two options for incorporating paraphrasing in this pipeline: paraphrase the input or paraphrase the TM. For our approach we have chosen to paraphrase the TM. There are many reasons for this. First, once a system is set up, the user can get the retrieved matches in real time; second, TMs can be stored in company servers and all processing can be done offline; third, the TM system need not be installed on the user computer and can be provided as a service.

3.1.1 Classification of Paraphrases

We have classified paraphrases obtained from PPDB 1.0 (Ganitkevitch, Benjamin, & Callison-Burch, 2013) into four types for our implementation on the basis of the number of words in the source and target phrases. These four categories are as follows:

1. Paraphrases having one word on both the source and target sides, e.g. "period" $\Rightarrow$"duration"

2. Paraphrases having multiple words on both sides but differing in one word only, e.g. "in the period" $\Rightarrow$ "during the period"

3. Paraphrases having multiple words as well as same number of words on both sides, e.g. "laid down in article" $\Rightarrow$ "set forth in article"

4. Paraphrases in which the number of words on the source and target sides differ, e.g. "a reasonable period of time to" $\Rightarrow$ "a reasonable period to"

This classification of paraphrases is carried out to be able to implement the edit distance procedure which considers paraphrases efficiently.

In TM, the paraphrases are stored in their reduced forms. We store only the smallest substring which can capture the whole non matching words instead of the whole paraphrase. In other words, we remove the matching words form both sides of the strings. When we store matching paraphrase with the TM segment, we have already considered the context and there is no need for it to be considered again while calculating edit-distance. This prevents the redundant computation. The reduced paraphrase is stored with the source word where the uncommon substring starts. We refer to this source word as "token". Table 1 shows the TM source segment (TMS), paraphrases captured for this segment (TMP) and paraphrases stored in their reduced form (TMR). In this case, the token "period" stores the two paraphrases "duration" and "time" and the token "laid" stores the two paraphrases "referred to" and "provided for by". For Type 3 and Type 4 the paraphrase source length (represented by LS in Table 1) is also stored along with the paraphrase (represented by TP in Table 1). In this case, length “2” for “laid down” is stored with paraphrase “referred to” and length “3” for “laid down in” is stored along with paraphrase “provided for by”.

TMS	the	period	laid down in article			4(3)	of	decision	468
TMP	the	period duration time	laid down referred to provided for	in in by	article article article	4(3)	of	decision	468
TMR	the	period duration time	laid LS TP 2 referred to 3 provided for by	down in article		4(3)	of	decision	468

Table 1: Representing paraphrases in TM

Algorithm 1 Basic Edit-Distance Procedure

```
 1: procedure EDIT-DISTANCE($Input$,$TMS$)
 2:     $M \leftarrow$ length(TMS)                    ▷ Initialise $M$ with length of TM segment
 3:     $N \leftarrow$ length(Input)                  ▷ Initialise $N$ with length of Input segment
 4:     D$[i, 0] \leftarrow i$ for $0 \leq i \leq N$                         ▷ initialisation
 5:     D$[0, j] \leftarrow j$ for $0 \leq j \leq M$                         ▷ initialisation
 6:     for $j \leftarrow 1...M$ do
 7:         TMToken $\leftarrow$ TMS$_j$                        ▷ get Token of TM segment
 8:         for $i \leftarrow 1...N$ do
 9:             InputToken $\leftarrow$ InputSegment$_i$            ▷ get Token of Input segment
10:             if InputToken $=$ TMToken then      ▷ match $InputToken$ with $TMToken$
11:                 substitutionCost $\leftarrow 0$          ▷ Substitution cost if matches
12:             else
13:                 substitutionCost $\leftarrow 1$          ▷ Substitution cost if not matches
14:             end if
15:             D$[i, j] \leftarrow$ min D$[i-1, j]$ + insertionCost, D$[i, j-1]$ + deletionCost, D$[i-1, j-1]$ + substitutionCost
16:         end for
17:     end for
18:     Return D$[N, M]$                              ▷ Return minimum edit-distance
19: end procedure
```

When edit distance is calculated the paraphrases of Types 1 and 2 can be implemented in a more efficient manner compared to paraphrases of Types 3 and 4. The paraphrase of the Types 1 and 2 have a unique property that they can be reduced to a single word paraphrases by removing the other matching words. The Algorithm 1 describes the basic edit-distance procedure. This implementation calculates word-based edit distance with cost 1 for insertion, deletion and substitution. The similarity is calculated by normalising edit-distance with the length of the larger segment. The basic procedure works by comparing the each token one by one in the input segment with the each token in TM segment. To implement the paraphrases of Types 1 and 2, we compare the list of paraphrases (reduced single tokens) stored with the TM token instead of comparing only TM token with the input token. This procedure has the advantage that complexity of the algorithm is only increased by the additional searching in the list ($log(n)$, where n is the number of Types 1 and 2 paraphrases stored with the TM token). There is no approximation here and the algorithm is able to capture all the paraphrases of Types 1 and 2. For these two types, the edit-distance procedure can be optimised globally as this is a simple case of matching one of these “paraphrases” when calculating the cost of substitution. For the example given in Table 1, if a word from input segment matches any of the words “period”, “time” or “duration”, the cost of substitution will be 0. The same technique can not be applied for types 3 and 4. For types 3 and 4, we use the greedy selection of paraphrase.

3.1.2 Implementation Procedure

Our procedure to implement paraphrases can be briefly described as the following steps:

step 1 Read the Translation Memories available

step 2 Collect all the paraphrases from the paraphrase database and classify them according to the classes presented in Section 3.1.1

step 3 Store all the paraphrases for each segment in the TM in their reduced forms

step 4 Read the file that needs to be translated

step 5 For each segment in the input file get the potential segments for paraphrasing in the TM according to the filtering steps of Section 3.1.4 and search for the most similar segment based on approach described in Section 3.1.5 and retrieve the most similar segment if above a predefined threshold

3.1.3 Paraphrase Corpus

We have used the PPDB 1.0 paraphrases database (Ganitkevitch et al., 2013) for our work. This database database contains lexical, phrasal and syntactic paraphrases automatically extracted using a large collection of parallel corpora. The paraphrases in this database are constructed using bilingual pivoting method. The hypothesis is that if two different English language phrases are translated to an identical foreign language phrase, the two English phrases are paraphrases of each other. Because of the automatic extraction, not all the paraphrases are completely accurate. The paraphrase database comes in six sizes (S, M, L, XL, XXL, XXXL) where S is the smallest and XXXL is the largest. The smaller packages contain only high precision paraphrases, while the larger ones aims at more coverage. The smallest package (S) contains 0.6 million while the largest package (XXXL) contains 68 million lexical and phrasal paraphrases. In our work, we have used lexical and phrasal paraphrases of “L” size. The reason for choosing L size was to retain the quality of segments retrieved using paraphrasing and at the same time gain some coverage. It contains 3 million paraphrases. We removed the paraphrase having punctuations, numbers or any special characters and retained the remaining 2 million paraphrases for our work.

3.1.4 Filtering

Before processing begins, for each input segment certain filtering steps are applied in order to speed up the process. The purpose of this preprocessing is to filter out unnecessary candidates for participating in the paraphrasing process. Because we are generally interested in candidates above a certain threshold it is obvious to filter out candidates below a certain threshold. For this purpose, we have defined certain filtering steps. The filtering of each step can be controlled by the respective threshold. Our filtering steps for getting potential candidates for paraphrasing are as follows:

Length Threshold: Filter out the segments based on length because if segments differ considerably in length, the edit-distance will also differ. So, if the threshold for length is 49%, the TM segments which are shorter than 49% of the input will be filtered.

Similarity Threshold: Next, we filter out the segments based on baseline edit-distance similarity. The TM segments which are having a similarity below a certain threshold will be removed. So, if the similarity threshold is 49%, the candidates below 49% similarity will be discarded.

N-best Threshold: Next, after filtering the candidates with the above two steps we sort the remaining segments in decreasing order of similarity and pick the top N segments.

Beam Threshold: Finally segments within a certain range of similarity with the most similar segment were selected for paraphrasing. So, if the beam threshold is 35% and the most similar segment has 95% similarity, segments with a similarity below 60% (95-35=60) will be discarded. [9]

3.1.5 Matching and Retrieval

For matching, similarity is calculated with the potential segments for paraphrasing extracted as per Section 3.1.4. Type 1 and Type 2 paraphrases after reduction are single-word paraphrases and Type 3 and Type 4 paraphrases have multiple words.

For paraphrases of Types 3 and 4 the algorithm takes the decision locally at the point where all paraphrases finish. The algorithm elaborating our paraphrase implementation and decision-making process is given in Algorithm 2 which extends the basic algorithm given in Algorithm 1. In Algorithm 2, $Input$ is the segment that we want to translate and TMS is the TM segment. Table 2 shows the edit-distance calculation of the first five tokens of the Input and TM segment with paraphrasing. In Algorithm 2, *lines 11 to 26* executes

	j	0	1	2				3	4				5
i		#	the	period duration time	laid	down	in	referred	to	provided	for	by	in
0	#	0	1	2	3	4	5	3	4	3	4	5	5
1	the	1	0	1	2	3	4	2	3	2	3	4	4
2	period	2	1	0	1	2	3	1	2	1	2	3	3
3	referred	3	2	1	1	2	3	0	1	1	2	3	2
4	to	4	3	2	2	**2**	3	1	**0**	2	2	3	1
5	in	5	4	3	3	3	**2**	2	1	3	3	**3**	0

Table 2: Edit-Distance Calculation using Algorithm 2

when Type 3 and Type 4 paraphrases are not available (e.g. edit-distance calculation of the second token "period"). *Lines 28 to 63* account for the case when Type 3 and Type 4 paraphrases are available. *Line 32* calculates the edit-distance of the corresponding longest source phrase and stores it in DS matrix as shown in Algorithm 2 (e.g. calculation of the

[9]These thresholds were determined empirically for our experiments.

Algorithm 2 Edit-Distance with paraphrasing procedure

```
1: procedure EDIT-DISTANCEPP(Input, TMS)
2:     $M \leftarrow length$(TMS)    ▷ number of tokens in TM segment
3:     $N \leftarrow length$(Input)    ▷ number of tokens in Input segment
4:     D$[i, 0] \leftarrow i$ for $0 \leq i \leq N$    ▷ initialise two dimensional matrix D
5:     D$[0, j] \leftarrow j$ for $0 \leq j \leq (M + p')$ where $p'$ accounts for increase in TM segment length because of paraphrasing
6:     $dp \leftarrow 0$ , $j \leftarrow 1$
7:     $scost \leftarrow 1$, $dcost \leftarrow 1$, $icost \leftarrow 1$    ▷ initialisation of substitution, deletion and insertion cost
8:     while $j \leq M$ do
9:         $t \leftarrow$ TMS$_j$    ▷ getting current TM token to process, e.g. $3^{rd}$ token "laid"
10:        if $t$ has no paraphrases of type 3 and type 4 or $dp \geq N$ then
11:            $dp \leftarrow dp + 1$, $j \leftarrow j + 1$
12:            for $i \leftarrow 1 \ldots N$ do
13:                InputToken $\leftarrow$ Input$_i$
14:                if InputToken $=$ $t$ then
15:                    $scost \leftarrow 0$
16:                else
17:                    $scost \leftarrow 1$
18:                end if
19:                if $scost = 1$ then
20:                    OneWordPP $\leftarrow$ getOneWordPP($t$)    ▷ get one word paraphrases associated with TM token $t$
21:                    if InputToken $\in$ OneWordPP then    ▷ applying type 1 and type 2 paraphrasing
22:                        $scost \leftarrow 0$
23:                    end if
24:                end if
25:                D$[i, dp] \leftarrow$ min D$[i, dp - 1] + dcost$, D$[i - 1, dp] + icost$, D$[i - 1, dp - 1] + scost$
26:            end for
27:        else
28:            TP $\leftarrow$ get paraphrases stored at $t$    ▷ e.g. TP for Token "laid" in Table 1
29:            LS $\leftarrow$ get corresponding source lengths stored at $t$    ▷ e.g. $LS$ for Token "laid" in Table 1
30:            $lsmax \leftarrow$ length of longest source phrase
31:            DS$[0, l - 1] \leftarrow D[0, dp + l]$ for $1 \leq l \leq lsmax$    ▷ initialise two dimensional matrix DS to calculate edit-distance of longest source phrase
32:            DS $\leftarrow$ calculate edit-distance of longest source phrase with Input using D    ▷ uses D for first word, consider Type 1 and Type 2 paraphrases
33:            $P \leftarrow$ number of paraphrases of type 3 and type 4    ▷ E.g. 2 for "laid"
34:            $index \leftarrow 0$, $paraphraselen \leftarrow 0$, $isppwin \leftarrow false$, $curDistance \leftarrow \infty$
35:            $prevDistance \leftarrow$ D$[dp, dp]$
36:            DTP$[k, 0, l - 1] \leftarrow$ D$[0, dp + l]$ for $0 \leq k \leq P - 1$ for $1 \leq l \leq length(TP[k])$▷ initialise three dimensional matrix DTP to calculate edit-distances of paraphrases
37:            for $k \leftarrow 0 \ldots P - 1$ do
38:                dps$[k] \leftarrow dp +$ LS$[k]$
39:                $ltp \leftarrow length$(TP$[k]$)    ▷ get paraphrase length e.g. 2 for "referred to"
40:                dpt$[k] \leftarrow dp + ltp$
41:                DTP$[k] \leftarrow$ calculate edit-distance of TP$[k]$ with Input using D    ▷ uses $D$ for first word of $TP[k]$
42:                if DTP$[k, ltp - 1$, dpt$[k]] <$ DS[LS$[k] - 1$, dps$[k]]$ and DTP$[k, ltp - 1$, dpt$[k]] < curDistance$ then
43:                    $ppwin \leftarrow true$
44:                    $curDistance \leftarrow$ DTP$[k, ltp - 1$, dpt$[k]]$
45:                    $index \leftarrow k$
46:                    $paraphraselen \leftarrow ltp$
47:                else if DS[LS$[k] - 1$, dps$[k]] < curDistance$ then
48:                    $ppwin \leftarrow false$
49:                    $curDistance \leftarrow$ DS[LS$[k] - 1$, dps$[k]]$
50:                    $index \leftarrow k$
51:                end if
52:            end for
53:            if $ppwin = true$ then    ▷ $true$ if paraphrase is better
54:                $j \leftarrow j +$ LS$[index]$
55:                $dp \leftarrow dp + paraphraselen$
56:                update D using DTP$[index]$
57:            else if $curDistance = prevDistance$ then    ▷ $true$ if source phrase is better and exactly matching
58:                $j \leftarrow j +$ LS$[index]$
59:                $dp \leftarrow dp +$ LS$[index]$
60:                update D using DS
61:            else
62:                $j \leftarrow j + 1$, $dp \leftarrow dp + 1$
63:                update D using DS
64:            end if
65:        end if
66:    end whileReturn D$[N, dp]$
67: end procedure
```

edit-distance of “laid down in” in Table 2). *Lines 37 to 50* account for the edit-distance calculation of each paraphrase (e.g. calculation of “referred to” and “provided for by” in Table 2). The edit-distance of each paraphrase is stored in DTP matrix as shown in Algorithm 2. *Lines 42 to 50* account for the selection of the minimum edit-distance paraphrase or source phrase. At *line 42*, the algorithm compares the edit-distance of paraphrase DTP (e.g. “referred to”) with the edit-distance of the corresponding source phrase (e.g. “laid down”) as well as with the current minimum distance. *Lines 54, 58 and 62* account for updating the value of j to reflect the current position for further calculation of edit-distance (e.g. $j = 5$ after selecting “referred to”) and *lines 56, 60 and 63* update the matrix D as shown in Algorithm 2.

As we can see in Table 2, starting from the third token of the TM, “laid”, three separate edit-distances are calculated, two for the two paraphrases “referred to” and “provided for by” and one for the corresponding longest source phrase “laid down in” and the paraphrase “referred to” is selected as it gives a minimum edit-distance of 0. The last column of Table 2 ($j = 5$) shows the edit-distance calculation of the next token “in” after selecting “referred to”.

3.1.6 Computational Considerations

The time complexity of the basic edit-distance procedure is $O(mn)$ where m and n are lengths of source and target segments, respectively. After employing paraphrasing of Type 1 and Type 2 the complexity of calculating the substitution cost increases from $O(1)$ to $O(log(p))$ (as searching the p words takes $O(log(p))$ time) where p is the number of paraphrases of Type 1 and Type 2 per token of TM source segment, which increases the edit-distance complexity to $O(mnlog(p))$. Employing paraphrasing of Type 3 and Type 4 further increases the edit-distance complexity to $O(lmn(log(p) + q))$, where q is the number of Type 3 and Type 4 paraphrases stored per token and l is the average length of paraphrase. Assuming the source and target segment are of same length n and each token of the segment stores paraphrases of length l, the complexity will be $O((q + log(p))n^2l)$. By limiting the number of paraphrases stored per token of the TM segment we can replace $(q+log(p))$ by a constant c. In this case complexity will be $c \times O(n^2l)$. However, in practice it will take less time as not all tokens in the TM segment will have p and q paraphrases and the paraphrases are also stored in the reduced form.

3.1.7 Other Approaches

There are other approaches to implement the edit-distance. Other most popular approach is using finite state automata. One need to create two automata for two set of strings and a composition transducer. The single source shortest path algorithm is used to find the minimum edit distance in this FST. The complexity of this approach is $O(|A_1||A_2|)$ where A_1 and A_2 are the sizes (the sum of the number of states and transitions) of input automata A_1 and A_2 (Mohri, 2003).

Representing each TM segment as a separate FST does not save time. The algorithm has same complexity as implementing using array based dynamic programming approach with an additional overhead of creating FST. Although time can be saved if the whole TM is represented as an FST. In this case, implementing paraphrasing is difficult and filtering steps cannot be applied when whole TM is represented as a single FST. However, similarity threshold implementation of our system (where selection of segments based on basic edit distance similarity required) can benefit using FST.

3.2 Automatic Evaluation using MT Metrics

For automatic evaluation, we have used English-French pairs of the 2013 release of the DGT-TM corpus (Steinberger, Eisele, Klocek, Pilos, & Schlüter, 2012). In our case English was the source language and French was the target language. From this corpus we have filtered out segments of fewer than five words and remaining pairs were used to create the TM and Test dataset. Tokenization of the English data was done using Berkeley Tokenizer (Petrov, Barrett, Thibaux, & Klein, 2006). The thresholds for filtering were kept as follows: length threshold 49%, similarity threshold 49%, N-best threshold 100 and beam threshold 35%. Table 3 shows our corpus statistics. In our case, average number of phrases per TM segment for which paraphrases are present in PPDB is 37 (AvgPhrases) and average number of paraphrases per TM segment present in PPDB is 146 (AvgPP) as shown in the Table 3.

	TM	**Test**
Segments	319709	25000
Source words	8200796	640265
Target words	7807577	609165
Average source length	25.65	25.61
Average target length	24.42	24.36
AvgPhrases	37	
AvgPP	146	

Table 3: Corpus Statistics

When we use paraphrasing in the matching and retrieval process, the fuzzy match score of a paraphrased segment is improved, which results in the retrieval of more segments at a particular threshold. This improvement in retrieval can be classified in two types: without changing the top rank; and by changing the top rank. For example, for a particular input segment, we have two segments A and B in the TM. Using simple edit-distance, A has 65% fuzzy score and B has 60% fuzzy score; the fuzzy score of A is better than that of B. As a result of using paraphrasing we notice two types of score changes:

1. the score of A is still better than or equal to that of B, for example, A has 85% fuzzy score and B has 70% fuzzy score;

2. the score of A is less than that of B, for example, A has 75% fuzzy score and B has 80% fuzzy score.

In the first case, paraphrasing does not supersede the existing model and just facilitate it by improving the fuzzy score so that the top segment ranked using edit distance get retrieved. However, in the second case paraphrasing changes the ranking and now the top ranked segment is different. In the second case, paraphrasing model supersedes the existing simple edit distance model. This second case also gives a reference to compare with. We take the top segment retrieved using simple edit distance as a reference against the top segment retrieved using paraphrasing and compare to see which is better.

Our evaluation has two objectives: first to see how much impact paraphrasing has in terms of retrieval and second to see the translation quality of those segments which changed their ranking and brought them up to the top because of the paraphrasing. The results of our evaluations are given in Tables 4, 5, 6, 7 and 8 where each table shows the similarity threshold for TM (TH), the total number of segments retrieved using the baseline approach (EDR), the additional segments retrieved using paraphrasing (+PPR), the percentage improvement in retrieval obtained over the baseline (Imp), the number of segments which changed their ranking and come up to the top because of paraphrasing (RC), the BLEU score (Papineni et al., 2002) on target side over translations retrieved by our approach for segments which changed their ranking and come up to the top because of paraphrasing (BPP), the BLEU score on target side over corresponding translations retrieved (irrespective of similarity score) by baseline approach for these segments (BED), the METEOR score on baseline approach (MED) and METEOR score using paraphrasing (MPP). Table 8 shows the results on threshold intervals whereas other tables show the results over a cutoff thresholds.

TH	**100**	**95**	**90**	**85**	**80**
EDR	6352	7062	8369	9829	10730
+PPR	92	110	107	109	123
Imp	1.45	1.56	1.28	1.11	1.15
RC	13	20	43	68	88
BED	65.89	70.29	60.70	63.29	61.31
BPP	**74.31**	**73.16**	**65.01**	63.29	60.84
MED	86.9	87.2	78.5	78.6	76.8
MPP	**91.3**	**90.5**	**84.7**	**82.2**	**80.2**

Table 4: Results on surface form: Using all four types of paraphrases

TH	**100**	**95**	**90**	**85**	**80**
EDR	6352	7062	8369	9829	10730
+PPR	69	80	81	86	90
Imp	1.09	1.13	0.97	0.87	0.84
RC	8	13	27	45	55
BED	60.86	71.43	61.96	65.10	63.28
BPP	**73.18**	**73.98**	**63.08**	64.37	**63.37**
MED	83.0	88.2	77.2	78.7	77.4
MPP	**89.3**	**92.0**	**83.5**	**82.5**	**81.6**

Table 5: Results on surface form: Using paraphrases of Types 1 and 2 only

TH	100	95	90	85	80
EDR	8179	8675	9603	10456	11308
+PPR	115	127	132	141	154
IMP	1.41	1.46	1.37	1.35	1.36
RC	21	30	43	73	108
BED	59.89	67.88	66.32	63.57	61.92
BPP	**68.61**	**78.04**	**75.40**	**69.06**	**63.93**
MED	78.3	79.6	76.9	75.0	73.4
MPP	**82.1**	**86.4**	**85.2**	**80.8**	**77.0**

Table 6: Results with placeholders: Using all four types of paraphrases

TH	100	95	90	85	80
EDR	8179	8675	9603	10456	11308
+PPR	98	102	103	112	114
IMP	1.2	1.18	1.07	1.07	1.01
RC	19	24	30	49	73
BED	52.00	54.81	60.09	62.13	57.42
BPP	**58.28**	**67.95**	**71.03**	**68.03**	**61.02**
MED	74.5	70.4	72.1	73.6	69.5
MPP	**77.3**	**80.5**	**82.3**	**81.1**	**74.6**

Table 7: Results with placeholders: Using paraphrases of Types 1 and 2 only

As we can see in Table 4, on surface form for a threshold of 90% we got a 1.28% improvement over baseline in terms of retrieval, i.e. we have retrieved 107 more segments. We can observe an increase of more than four BLEU points for the 90% threshold and an increase of more than eight BLEU points for the 100% threshold for the segments which change their rank. There are 13 segments for threshold 100% which change their rank and 43 segments for threshold 90% which change their rank. Table 5 shows improvements we have obtained using paraphrases of Types 1 and 2 only.

To get more matches in TM, we have removed punctuation and replaced numbers and dates with placeholders. For this experiment we observed significant improvement for a threshold of 80% and above as shown in Tables 6 & 7. We can observe that after removing punctuation and replacing numbers and dates with placeholders we obtained more than five BLEU points improvement over the baseline for a threshold of 85% and above for the segments which changes their rank.

Table 7 shows the improvements we have obtained using paraphrases of Type 1 and 2 only with placeholders. As we can see, improvements in retrieval is less compared to Table 6 which uses all paraphrases but the BLEU score is still improving significantly. We can observe an increase of more than 10 BLEU points over the baseline for thresholds of 95% and 90%.

Table 8 shows the results based on the intervals (on surface form). We can observe that for exact matches we get the improved BLEU and METEOR. For interval [70, 85) and [55, 70), METEOR is improving (MPP in Table 8) but BLEU is decreased whereas for [85,

TH	100	[85, 100)	[70, 85)	[55, 70)
EDR	6352	3477	2425	2128
+PPR	92	102	144	144
%Imp	1.44	2.93	5.93	6.76
RC	13	27	41	33
BED	65.89	51.11	39.05	26.16
BPP	**74.31**	42.22	36.70	19.67
MED	86.99	69.60	59.00	44.84
MPP	**91.37**	68.20	**59.15**	**45.36**

Table 8: Results with Threshold intervals: All four types of paraphrases

100) both scores are decreased.

The automatic evaluation suggest that paraphrasing bring improvement in retrieval. However, automatic measures do not substantiate how much benefit a translator get from these improvements. In the next section, we present the human evaluation involving translators.

3.3 Human Evaluation

In TM, the performance of retrieval can be measured by counting the number of segments or words retrieved. However, NLP techniques are not 100% accurate and most of the time, there is a tradeoff between the precision and recall of this retrieval process. This is also one of the reasons that TM developers shy away from using semantic matching. One cannot measure the gain unless this retrieval benefits the translator. We considered here two aspects from the translators perspective: less editing needed; saving post editing time. Measuring these two aspect is a difficult process. This is also the reason why previous proposed approaches (Planas & Furuse, 1999; Hodász & Pohl, 2005; Pekar & Mitkov, 2007; Mitkov, 2008; Utiyama et al., 2011) lack proper human evaluations.

Often quality assessment of TM or MT is needed for different purposes including evaluating different MT systems as well as comparing segments retrieved by TM against MT output. Apart from automatic evaluations using MT metrics, keystroke, post-editing time analysis and subjective evaluations by translators are often used for this purpose. Keystroke analysis has been used to judge translators' productivity (Langlais & Lapalme, 2002; Whyman & Somers, 1999). Koponen, Aziz, Ramos, and Specia (2012) suggested that post-editing time reflects the cognitive effort in post-editing the MT output. Sousa, Aziz, and Specia (2011) evaluated the different MT system performances against translating from scratch. Their study also concluded that subjective evaluations of MT system output also correlate with the post-editing time needed. In our research we have carried out both automatic as well as human evaluations. We measure post-editing time, keystrokes along with subjective evaluations. To the best of our knowledge this is the first work on assessing the quality of any type of semantically informed TM fuzzy matches based on post-editing time or keystrokes.

3.3.1 Corpus, Tool and Translators

For TM and the input set, we have used English-German pairs of the Europarl V7.0 (Koehn, 2005) corpus with English as the source language and German as the target language. This is different corpus from the corpus that we have used in our experiments in the automatic evaluation (Section 3.2). Furthermore, we used English-German instead of English-French pairs earlier used. The reason for changing to this corpus was that the translation students participated in this test were native speaker of German. The translation students were not familiar with the domain and terminology used in DGT-TM. DGT-TM is of legal domain with difficult terminology that is not easier to translate for a common translation student. Furthermore, it can introduce a bias of unfamiliarity or familiarity because of the terms used in legal domain. Europarl is a corpus containing text of European parliament proceedings. Therefore, the text is of spoken genre in a formal settings. The text has been compiled by Koehn (2005) and he used it for the SMT experiments. We have also done automatic evaluation on this dataset for the sake of completeness.

From Europarl corpus we have filtered out segments of fewer than seven words and greater than 40 words, with the remaining pairs used to create the TM and Test datasets. Tokenization of the English data was done using the Berkeley Tokenizer (Petrov et al., 2006). In these experiments, we have not paraphrased any capitalised words. This is to avoid paraphrasing any named entities. The thresholds for filtering were kept as follows, length threshold 39%, similarity threshold 39%, N-best threshold 100 and beam threshold 35%. Table 9 shows our corpus statistics.

	TM	**Input**
Segments	1565194	9981
Source words	37824634	240916
Target words	36267909	230620

Table 9: Corpus Statistics

We have carried out four different experiments. We have measured post-editing time, keystrokes, two subjective evaluations, HTER and HMETEOR to substantiate our research. Section 3.3.2.1 describes the settings and measures used for post-editing evaluation, and Sections 3.3.2.2 and 3.3.2.3 describe the settings for the subjective evaluations. The translators involved in the experiments were third year bachelor or masters students who were native speakers of German with English language level C1 or above. Translators were in the age group of 21 to 40 years old with a majority of female students.

We have used the PET tool (Aziz, Castilho, & Specia, 2012) for all our human experiments. However, settings were changed depending on the experiment. To familiarise translators with the PET tool we have done a pilot experiment before the actual experiment with the Europarl corpus. This experiment was done on a corpus (Vela, Neumann, & Hansen-Schirra, 2007) different from Europarl. 18 segments are used in this experiment. While the findings are not included in this document, they informed the design of our main experiments.

3.3.2 Experimental settings

We have performed evaluations in four different settings given below:

3.3.2.1 Post-editing Time (PET) and Keystrokes (KS)

In this evaluation, the translators were presented with fuzzy matches and the task was to post-edit the segment in order to have a correct translation. The translators were presented with an input English segment, German segment retrieved from TM for post-editing and an English segment used for matching in TM. [10]

In this task, we have recorded post-editing time (PET) and keystrokes (KS). The post-editing time taken for the whole file is calculated by summing up the time taken on each segment. Only one segment is visible on the whole screen. The segment is only visible after clicking and the time is recorded from when the segment becomes visible until the translator finishes post-editing and goes to the next screen. The next screen is a blank screen so that the translator can have a rest after post-editing a segment. The figure 1 shows the blank screen and the figure 2 shows the screen when a translator is editing.

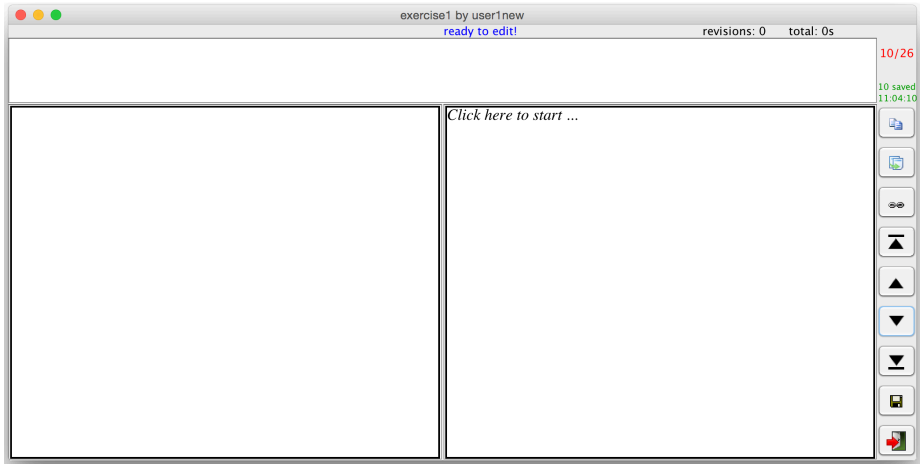
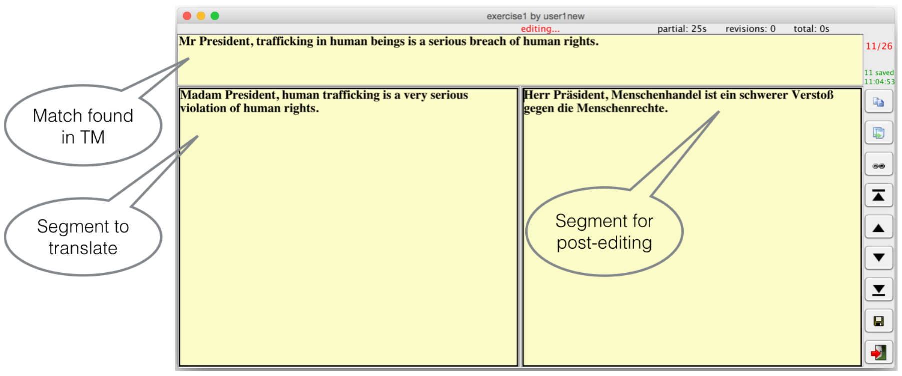


Figure 1: Blank screen

The translators were aware that the time is being recorded. Each translator post-edited half of the segments retrieved using simple edit distance (ED) and half of the segments retrieved using paraphrasing (PP). The ED and PP matches were presented one after the other. However, the same translator did not post-edit the match retrieved using PP and ED for the same segment. Five different translators post-edited the segment retrieved using PP and another five different translators post-edited the match retrieved using ED.

Post-editing time (PET) for each segment is the mean of the normalised time (N) taken by all the translators on this segment. Furthermore, normalised time (N) is calculated by multiplying the actual time taken by a translator on a segment by the average time taken on that file by all translators, divided by the time taken by the translator on this file. This normalisation is done to account for both slow and fast translators.

[10] To simulate the TM working procedure we have also presented the English segment used in the matching process to retrieve the German segment.



Mr President, trafficking in human beings is a serious breach of human rights.

Madam President, human trafficking is a very serious violation of human rights.

Herr Präsident, Menschenhandel ist ein schwerer Verstoß gegen die Menschenrechte.

Figure 2: Editing is in progress

$$\text{PET}_j = \frac{\sum_{i=1}^{n} N_{ij}}{n} \tag{6}$$

$$N_{ij} = T_{ij} \times \frac{\text{Avg time on this file}}{\sum_{j=1}^{m} T_{ij}} \tag{7}$$

In the equations 6 and 7 above, PET_j is the post editing time for each segment j, n is the number of translators, N_{ij} is the normalised time of translator i on segment j, m is the number of segments in the file, and T_{ij} is the actual time taken by a translator i on a segment j.

Along with the post-editing time, we also recorded all printable keystrokes, whitespace and erase keys pressed. For our analysis, we considered average keystrokes pressed by all translators for each segment.

3.3.2.2 Subjective Evaluation with Two Options (SE2) In this evaluation, we have done subjective evaluation with two options (SE2). We presented fuzzy matches retrieved using both paraphrasing (PP) and simple edit distance (ED) to the translators. The translators were unaware of the source (ED or PP) of fuzzy matches. To neutralise any bias, half of the ED matches were tagged as A and the other half as B, with the same applied to PP matches. The translator has to choose between two options: A is better; or B is better. 17 translators participated in this test. Finally, the decision of whether 'ED is better' or 'PP is better' is made on the basis of how many translators choose one over the other.

3.3.2.3 Subjective Evaluation with Three Options (SE3) This evaluation is similar to Evaluation 2 except that we have given one more option to translators. Translators can choose among three options: A is better; B is better; or both are equal. 7 translators participated in this test.

3.3.2.4 Subjective Evaluation on Exact Matches (SEM) In this evaluation, our objective is to check whether exact match after paraphrasing is really an exact match. We have presented only exact matches retrieved using paraphrasing, which are not exact matches using simple edit-distance. 14 segments were presented to ten or more translators. The translators have to correct the segment and select an option from two options presented: Can not be accepted as it is (Post-edting was required) ; Correct Translation (No Post-editing was required).

3.3.3 Results and Analysis of PET, KS, SE2 and SE3

The retrieval results are given in Table 10 and Table 11. Table 10 presents the results in the format as opted in the automatic evaluation section.

TH	100	95	90	85	80	75	70
EDR	117	127	163	215	257	337	440
PPR	133	143	185	248	306	416	542
Imp	13.68	12.6	13.5	15.35	19.07	23.44	23.18
RC	9	10	16	25	36	65	97
BED	31.88	32.37	27.70	21.71	19.32	14.98	12.25
BPP	**52.00**	**47.92**	**43.90**	**31.76**	**25.24**	**19.75**	**15.28**
MED	45.48	46.48	45.59	39.24	37.32	34.02	31.10
MPP	**68.08**	**67.03**	**61.09**	**50.07**	**44.16**	**38.35**	**33.19**

Table 10: Results on Europarl dataset: Automatic Evaluation, using all four types of paraphrases

Table 11 shows the interval wise results. We have chosen the threshold intervals so as to select the segments from each range for the human evaluations. Table 11 shows similarity threshold for TM (TH), the total number of segments retrieved using the baseline approach (EDR) in the respective intervals, the additional number of segments retrieved using the paraphrasing approach (+PPR) in the respective intervals, the percentage improvement in retrieval obtained over the baseline (Imp), the number of segments that changed their ranking and rose to the top because of paraphrasing (RC), and the number of unique paraphrases used to retrieve +PPR (NP) and RC (NPRC).

We can see in Table 11 and 10 that for the Europarl dataset also we get improvements. Table 11 shows that when using paraphrasing we obtain around 13.67% improvement in retrieval for exact matches and more than 30% and 43% improvement in the intervals [85, 100) and [70, 85), respectively. This clearly shows that paraphrasing significantly improves the retrieval results. We have also observed that there are different paraphrases used to bring this improvement. As given in Table 11, in the interval [70, 85), 169 different paraphrases are used to retrieve 98 more segments.

The sets' distribution for human evaluation is given in the Table 12. The sets contain randomly selected segments from the additionally retrieved segments using paraphrasing who changed their top ranking.

Results for human evaluations (PET, KS, SE2 and SE3) on both sets (Set1 and Set2) are given in Table 13. Here 'Seg #' represents the segment number, 'ED' represents the match retrieved using simple edit distance and 'PP' represents the match retrieved after incorporating paraphrasing. 'EDB', 'PPB' and 'BEQ' in Subjective Evaluations represent the number

TH	100	[85, 100)	[70, 85)	[55, 70)
EDR	117	98	225	703
+PPR	16	30	98	311
%Imp	13.67	30.61	43.55	44.23
RC	9	14	55	202
BED	31.88	13.18	6.85	5.32
BPP	52.00	17.10	8.37	5.60
MED	45.48	34.37	25.76	20.05
MPP	68.08	40.00	25.82	21.69
NP	24	49	169	535
NPRC	14	24	92	356

Table 11: Results of Retrieval

TH	100	[85, 100)	[70, 85)	Total
Set1	2	6	6	14
Set2	5	4	7	16
Total	7	10	13	30

Table 12: Test Sets for Experiments PET, KS, SE2 and SE3

of translators prefer the 'ED is better', 'PP is better' and 'Both are equal' options respectively.

3.3.3.1 Results: Post-editing Time (PET) and Keystrokes (KS) As we can see in Table 13, improvements were obtained for both sets. $\uparrow$ demonstrates cases in which PP performed better than ED and $\downarrow$ shows where ED performed better than PP. Entries in bold for PET, KS and SE2 indicate where the results are statistically significant [11].

For Set1, Translators made 356.20 keystrokes compared to 532.60 keystrokes in editing PP and ED matches respectively. Translators took 466.44 seconds for PP as opposed to 520.02 seconds for ED matches. So, by using PP matches, translators edit 33.12% less (49.52% more using ED), which saves time by 10.3%.

For Set2, Translators made 468.59 keystrokes compared to 570.6 keystrokes in editing PP and ED matches respectively. Translators took 603.17 seconds for PP as opposed to 657.75 seconds for ED matches. This means that by using PP matches, translators edit 17.87% less (21.76% more using ED), which saves time by 8.29%.

In total, combining both the sets, Translators made 824.79 keystrokes compared to 1103.2 keystrokes in editing PP and ED matches respectively. Translators took 1069.61 seconds for PP as opposed to 1177.77 seconds for ED matches. So, by using PP matches, translators edit 25.23% less, which saves the time by 9.18%. In other words, ED matches require 33.75% more keystrokes and 10.11% more time. We can observe that the percent-

[11] $p < 0.05$, Welch single tailed t-test is used for PET and KS, and χ^2 test for SE2. Because of the small sample size for SE3, no significance test was done on individual segment basis.

	Post-editing				**Subjective Evaluations**				
	PET		**KS**		**SE2 (2 Options)**		**SE3 (3 options)**		
Seg #	**ED**	**PP**	**ED**	**PP**	**EDB**	**PPB**	**EDB**	**PPB**	**BEQ**
1	42.98	41.30↑	42.4	**0.4**↑	**1**	**16**↑	0	7↑	0
2!+	13.72	10.65↑	2.8	2.4↑	10	7↓	2	2	3
3*!	13.88	12.62↑	2.0	3.6↓	12	5↓	4	1↓	2
4	37.97	**17.64**↑	26.2	**6.2**↑	**1**	**16**↑	0	6↑	1
5!+	21.52	17.69↑	22.4	13.2↑	**13**	**4**↓	2	3↑	2
6!+	41.14	42.74↓	13.2	34.4↓	**4**	**13**↑	2	0	5
7!+	33.69	31.59↑	34.0	33.4↑	10	7↓	1	0	6
8	47.14	**23.41**↑	61.6	**6.4**↑	**0**	**17**↑	0	7↑	0
9	22.89	**14.20**↑	37.2	**2.2**↑	**0**	**17**↑	0	6↑	1
10	46.89	38.20↑	77.6	65.6↑	**1**	**16**↑	0	1	6
11	58.25	53.65↑	82.8	58.8↑	**0**	**17**↑	0	3	4
12!+	34.04	45.03↓	36.8	39.6↓	**2**	**15**↑	0	6↑	1
13	30.34	**21.12**↑	54.8	39.2↑	7	10↑	1	1	5
14!+	75.50	96.54↓	38.8	50.8↓	5	12↑	0	3	4
Set1-subtotal	520.02	466.44	532.60	356.20	66	172	12	46	40
15	24.14	**9.18**↑	24.0	**0.0**↑	5	12↑	1	5↑	1
16*+	28.30	29.20↓	23.4	15.4↑	11	6↓	2	2	3
17*!	65.64	53.49↑	6.2	22.4↓	10	7↓	2	3↑	2
18	41.91	**20.98**↑	28.0	**2.0**↑	**1**	**16**↑	0	6↑	1
19	29.81	19.71↑	23.8	**6.8**↑	7	10↑	2	3↑	2
20	41.25	**15.42**↑	39.0	**3.8**↑	**0**	**17**↑	1	5↑	1
21*!	**42.04**	65.44↓	39.4	36.0↑	7	10↑	1	2	4
22	29.28	35.87↓	17.0	33.4↓	12	5↓	5	0↓	2
23	**32.64**	49.49↓	**11.4**	50.8↓	11	6↓	2	2	3
24!+	59.35	54.54↑	79.6	79.2↑	**17**	**0**↓	5	0↓	2
25	62.51	61.30↑	71.0	54.0↑	**2**	**15**↑	0	3	4
26*!	36.82	41.06↓	55.0	**23.4**↑	**1**	**16**↑	0	6↑	1
27!+	**27.21**	44.02↓	**24.4**	48.8↓	**4**	**13**↑	1	5↑	1
28	40.99	**33.08**↑	39.6	**24.6**↑	5	12↑	3	4↑	0
29	52.01	**31.55**↑	50.6	**23.4**↑	**2**	**15**↑	0	6↑	1
30*!	43.76	38.76↑	38.2	44.6↓	**15**	**2**↓	1	1	5
Set2-subtotal	657.75	603.17	570.6	468.59	110	162	26	53	33
Total	1177.77	1069.61	1103.2	824.79	176	334	38	99	73

Table 13: Results of Human Evaluation on Set1 (1-14) and Set2 (15-30)

age improvement obtained by keystroke analysis is smaller compared to the improvement obtained by post-editing time. This is because the translator spends a fair amount of time reading a segment before starting editing.

3.3.3.2 Results: Using post-edited references We have also calculated the human-targeted translation error rate (HTER) (Snover et al., 2006) and human-targeted METEOR (HMETEOR)(Denkowski & Lavie, 2014). HTER and HMETEOR was calculated between ED and PP match presented for post-editing and references generated by editing corresponding ED and PP match. Table 14 lists HTER5 and HMETEOR5, which use five corresponding ED or PP references only and HTER10 and HMETEOR10, which use all ten references generated using ED and PP.

Table 14 shows improvement in both the HTER5 and HMETEOR5 scores. For Set-1, HMETEOR5 improved from 59.82 to 81.44 and HTER5 improved from 39.72 to 17.63. For Set-2, HMETEOR5 improved from 69.81 to 80.60 and HTER5 improved from 27.81 to 18.71. We can also observe that while ED scores of Set1 and Set2 differ substantially (59.82 vs 69.81 and 39.72 vs 27.81), PP scores are nearly the same (81.44 vs 80.60 and 17.63 vs 18.71). This suggest that paraphrasing not only brings improvement but also consistency.

	Set-1		Set-2	
	ED	**PP**	**ED**	**PP**
HMETEOR5	59.82	81.44	69.81	80.60
HTER5	39.72	17.63	27.81	18.71
HMETEOR10	59.82	81.44	69.81	80.61
HTER10	36.93	18.46	27.26	18.40

Table 14: Results using human targeted references

3.3.3.3 Results: Subjective evaluations The subjective evaluations also show significant improvements.

In subjective evaluation with two options (SE2) as given in Table 13, in total 510 (30$\times$17) replies for 30 segments from both sets by 17 translators, 334 replies tagged 'PP is better' and 176 replies tagged 'ED is better' (statistically significant, χ^2 test, $p < 0.001$).

In subjective evaluation with three options (SE3), in total 210 (30$\times$7) replies for 30 segments from both sets by 7 translators, 99 replies tagged 'PP is better', 73 replies tagged 'both are equal' and 38 replies tagged 'ED is better' (statistically significant, χ^2 test, $p < 0.001$).

3.3.3.4 Results: Segment wise analysis An individual segment-by-segment analysis of 30 segments from both sets shows that 21 segments extracted using PP were found to be better by PET evaluation and 20 segments using PP were found to be better by KS evaluation. In subjective evaluations, 20 segments extracted using PP were found to be better by SE2 evaluation whereas 27 segments extracted using PP were found to be better or equally good by SE3 evaluation (15 segments were found to be better and 12 segments were found to be equally good).

We have also observed that not all evaluations perfectly correlate with each other. '!', '+' and '*' on each segment number in Table 13 denotes the contradictions with evaluations. '!' denotes where PET and SE2 contradict each other, '+' denotes where KS and SE2 contradict each other and '*' denotes where PET and KS contradict each other.

In twelve segments where KS evaluation or PET evaluation show PP as statistically significant better, except for two cases all the evaluations also shows them better and most of the segments are significantly better. For Seg #13 SE3 shows 'Both are equal' and for Seg #26, PET is better for ED, however for these two sentences also all the other evaluations show PP as better.

In three segments (Seg #'s 21, 23, 27) where KS evaluation or PET evaluation show ED as statistically significant better, but none of the segment are tagged better by all the evaluations. In Seg #21, only PET is better and all other evaluations[12] shows PP as better. In Seg #23, SE3 shows 'both are equal'. Seg #23 is given as follows:

Input: The next item is the Commission declaration on Belarus .

ED: The next item is the Commission Statement on AIDS .//Als nächster Punkt folgt die Erklärung der Kommission zu AIDS.

[12]In the Section 3.3.3 all evaluations refers to PET, KS, SE2 and SE3.

PP: The next item is the Commission statement on Haiti .//Nach der Tagesordnung folgt die Erklärung der Kommission zu Haiti.

In Seg #23, apart from "AIDS" and "Haiti" the source side does not differ but the German side differs. The reason for PP match retrieval was that "statement on" in lower case was paraphrased as "declaration on" while in the other segment "Statement" was capitalised and hence was not paraphrased. If we look at the German side of both ED and PP, "Nach der Tagesordnung" requires a broader context to accept it as a translation of "The next item" whereas "Als nächster Punkt" does not require much context.

In Seg #27, we get the contradictions between post-editing evaluations and subjective evaluations. Seg #27 is given below (EDPE and PPPE are post-edited translations of ED and PP match respectively):

Input: That would be an incredibly important signal for the whole region .

ED: That could be an important signal for the future .//Dies könnte ein wichtiges Signal für die Zukunft sein.

PP: That really would be extremely important for the whole region .//Und das wäre wirklich für die ganze Region extrem wichtig.

EDPE: Dies könnte ein unglaublich wichtiges Signal für die gesamte Region sein.

PPPE: Das wäre ein unglaublich wichtiges Signal für die ganze Region.

In subjective evaluations, translators tagged PP as better than ED, which is also the case. But, post-editing suggests that it takes more time and keystrokes to post-edit the PP compare to ED.

There is one segment, Seg #22, in which all the evaluations shows ED is better. Seg #22 is given below:

Input: I would just like to comment on one point.

ED: I would just like to emphasise one point.//Ich möchte nur eine Sache betonen.

PP: I would just like to concentrate on one issue.//Ich möchte mich nur auf einen Punkt konzentrieren.

In segment 22, the ED match is clearly closer to the input than the PP match. Paraphrasing "on one point" as "on one issue" does not improve the result. Also, "konzentrieren" being a long word takes more time and keystrokes in post-editing.

3.3.4 Results: Subjective Evaluation on Exact Matches only (SEM)

The results of subjective evaluation on exact matches (SEM) are given in Table 15. On 10 segments out of 14 segments, seven or more (two third) of the translators agree that the segment is not required any post-editing. In rest of the cases, for two segments (Seg #13 and Seg #15) the judgements was contradictory with half of the translators agree and half disagree whether the segment needs post-editing. In other two cases (Seg #7 and Seg #9) most of the translators chosen to post-edit the segments. The Seg #7 is given below (PPPE represents the most preferred post-edited translation):

Seg #	Yes	No	No Post-editing
1	11	0	Yes
2	10	1	Yes
3	10	1	Yes
4	9	2	Yes
5	8	3	Yes
6	9	2	Yes
7	2	9	**No**
8	10	1	Yes
9[13]	1	9	**No**
10	11	0	Yes
11	11	0	Yes
12	6	5	indecisive
13	7	4	Yes
14	5	6	indecisive
Total	110	43	-

Table 15: Results of Human Evaluation on Exact Matches

EN The vote will take place immediately following the ongoing debates.

PP The vote will take place immediately after the ongoing debates. // Die Abstimmung findet unverzüglich im Anschluss an die laufenden Aussprachen statt.

PPPE Die Abstimmung findet unverzüglich im Anschluss an die laufenden Debatten statt.

We can see that the source segment match is accurate. Most of the translators edited 'Aussprachen' to 'Debatten'.

The Seg #9 is given below:

EN (The sitting was suspended at 11.25 p.m.)

PP (The sitting was closed at 11.25 p.m.) // (Die Sitzung wird um 23.25 geschlossen)

PPPE (Die Sitzung wurde um 23:25 geschlossen)

In segment 9, 'closed' and 'suspended' bit differ but this does not impact the target side. Translators changed auxiliary verb 'wird' to 'wurde'.

Because in most of the segments translators agree to accept as it is. This suggests that paraphrasing match can be presented as an exact match.

3.4 Conclusion

In this section, we have presented an approach to implement paraphrasing in matching and retrieval. The method use existing paraphrase database in an efficient manner. Our both automatic as well as human evaluations suggest that paraphrasing improves matching and retrieval in TM. We conclude that paraphrasing significantly improves retrieval. We observe more than 30% and 43% improvement for the threshold intervals [85, 100) and [70, 85), respectively for Europarl corpus. The quality of the retrieved segment is also significantly better, which is evident from all our human translation evaluations. On average on both sets used for human evaluation, compared to paraphrasing simple edit distance takes 33.75% more keystrokes and 10.11% more time when evaluating the segments who changed their top rank and come up in the threshold intervals because of paraphrasing. The approach as it is can be used for any language pair where the source language is English. A paraphrasing corpus (and word segmentation for the languages where words are not separated by a space e.g. Chinese) of the source language will be required to extend this approach for the language pair where source language is not English.

In the next section, we describe the experiments performed for TM matching and retrieval by the system implementing advanced semantic similarity computation using various NLP techniques.

4 Advanced semantic matching for TM

In this section, we present experiments performed using a semantic similarity measure based on available NLP technology. The implicit assumption is that using these techniques we can get more semantically similar matches which edit distance with paraphrasing proposed previously may miss. We use a supervised machine learning approach. The approach use SVM regression model trained on features extracted using NLP technology from a human annotated dataset. Our approach relies on features inspired by deep semantics (such as parsing and paraphrasing), machine translation evaluation and Corpus Pattern Analysis (CPA[14]).

4.1 Our Approach

We used a Support Vector Machine (SVM) regression model with RBF kernel for the semantic similarity calculation between two segments. For the actual implementation we used LibSVM[15] (Chang & Lin, 2011). The SVM used an RBF kernel with $C = 8$ and $\gamma = 0.125$. The values of C and γ have been optimised through a grid-search which uses a 5-fold cross-validation method. This model estimates a continuous score between 1 and 5 for each sentence. We considered a continuous score between 1 to 5 because of the available training dataset is tagged with scores 1 to 5.

Data Set: The training dataset for the SVM is a set of 4934 parallel sentences of the SICK dataset (Marelli et al., 2014). This dataset is annotated with similarity scores by humans. The dataset consists of simple sentences extracted mostly from image captions. The dataset has on average 9.6 words per sentence.

The system described in (Gupta et al., 2014) calculates the similarity and entailment between a pair of sentences. The system performed well in the SemEval-2014 task with 0.71 Pearson correlation on human annotated dataset. This system was adapted to measure the similarity between two TM segments. Given the amount of calculation involved

[14]http://pdev.org.uk

[15]http://www.csie.ntu.edu.tw/ cjlin/libsvm/

in the task, we kept only those features, which can be quickly calculated and proved the most useful for the original system. After removing these features we observed a slightly low (0.69) Pearson correlation.

4.1.1 Features

Our system uses 12 features for training the system. The feature descriptions are given below:

4.1.1.1 Language Technology Features We used existing language processing tools to extract features. Stanford CoreNLP[16] toolkit provides lemma, parts of speech (POS), named entities, dependencies relations of words in each sentence.

We calculated Jaccard similarity on surface form, lemma, dependencies relations, POS and named entities to get the feature values. The Jaccard similarity computes sentence similarity by dividing the overlap of words on the total number of words of both sentences.

$$Sim(s1, s2) = \frac{|s1 \cap s2|}{|s1 \cup s2|} \tag{8}$$

where in equation (8), $Sim(s1, s2)$ is the Jaccard similarity between sets of words $s1$ and $s2$.

We used the same toolkit to identify coreference relations and determine clusters of coreferential entities. The coreference feature value was calculated using clusters of coreferential entities. The intuition is that sentences containing coreferential entities should have some semantic relatedness. In order to extract clusters of coreferential entities, the pair of sentences was treated as a document. The coreference feature value using these clusters was calculated as follows:

$$Value_{coref} = \frac{CC}{TC} \tag{9}$$

where CC is the number of clusters formed by the participation of entities (at least one entity from each sentence of the pair) in both sentences and TC is the total number of clusters.

We calculated two separate feature values for dependency relations: the first feature concatenated the words involved in a dependency relation and the second used grammatical relation tags. For example, for the sentence pair “the kids are playing outdoors” and “the students are playing outdoors” the Jaccard similarity is calculated based on concatenated words “kids::the, playing::kids, playing::are, ROOT::playing, playing::outdoors” and “students::the, playing::students, playing::are, ROOT::playing, playing::outdoors” to get the value for the first feature and “det, nsubj, aux, root, dobj” and “det, nsubj, aux, root, dobj” to get the value for the second feature.

These language technology features try to capture the token based similarity and grammatical similarity between a pair of sentences.

[16]http://nlp.stanford.edu/software/corenlp.shtml

4.1.1.2 Paraphrasing Features We used the PPDB paraphrase database (Ganitkevitch et al., 2013) to get the paraphrases. We used lexical and phrasal paraphrases of "L" size. For each sentence of the pair, we created two sets of bags of n-grams ($1 \leq n \leq$ length of the sentence). We extended each set with paraphrases for each n-gram available from paraphrase database. We then calculated the Jaccard similarity (see Section 4.1.1.1) between these extended bag of n-grams to get the feature value. This feature capture the cases where one sentence is a paraphrase of the other.

4.1.1.3 Negation Feature Our system does not attempt to model similarity with negation, but since negation is an important feature for contradiction, we designed a non-similarity feature. The system checks for the presence of a negation word such as 'no', 'never' and 'not' in the pair of sentences and returns "1" ("0" otherwise) if both or none of the sentences contain any of these words.

4.1.1.4 Machine Translation Evaluation Features Additionally, we used BLEU (Papineni et al., 2002), a very popular machine translation evaluation metric, as a feature. BLEU is based on n-gram counts. It is meant to capture the similarity between translated text and references for machine translation evaluation. The BLEU score over surface, lemma and POS was calculated to get three feature values. In a pair of sentences, one side was treated as a translation and another as a reference. We applied it at the sentence level to capture the similarity between two sentences.

4.1.1.5 Corpus Pattern Analysis Feature Corpus Pattern Analysis (CPA) (Hanks, 2013) is a procedure in corpus linguistics that associates word meaning with word use by means of semantic patterns. CPA is a new technique for mapping meaning onto words in text. It is currently being used to build a "Pattern Dictionary of English Verbs"(PDEV[17]). It is based on the Theory of Norms and Exploitations (Hanks, 2013).

There is one feature extracted from PDEV. The feature make use of a derived resource called the CPA network (Bradbury & El Maarouf, 2013). The CPA network links verbs according to similar semantic patterns (e.g. both 'pour' and 'trickle' share an intransitive use where the subject is "liquid").

The feature value is calculated as follows. It compares the main verbs in both sentences. When both verbs are same or both verbs occur in a same pattern, the system returns a value of "1". When verb in one segment occur in a pattern but verb in the other segment does not share the same pattern, the system returns 0. In all other cases, system return 0.5. We do not detect a pattern but check whether both verbs belong to the same pattern irrespective of the pattern. One of the example of a CPA pattern is given below with the possible set of verbs following this pattern:

[Human] [Verb] [Rule]

Verb: follow abrogate abolish activate disregard simplify obey break

[17] http://pdev.org.uk

4.2 Experiments and Results

We carried out evaluations on two different sets. Because the system implemented is slow, we used the smaller subsets of the dataset used in Section 3 (DGT-TM corpora (Steinberger et al., 2012)). The test sets were generated by a random selection of segments. We use same measure as baseline as used in Section 3 (the word based edit-distance measure). The statistics for our test sets is given in the Table 16 below:

	Test-1 (# segments)	Test-2 (# segments)
Input	500	2500
TM	5000	10000

Table 16: Test sets statistics

We performed both a manual and an automatic evaluation. For our automatic evaluation, we used the machine translation evaluation metrics METEOR (Denkowski & Lavie, 2014) and BLEU (Papineni et al., 2002). For each input segment, we retrieved the most similar sentence (and their proposed translation into French) as indicated by the baseline and our similarity metric. Table 17 presents the results of automatic evaluation when having a threshold of 70% over the edit-distance. BLEU-ED-70 represents BLEU score using edit distance, BLEU-SS-70 represents BLEU score using our approach, METEOR-ED-70 represents METEOR score using edit distance, and METEOR-SS-70 represents METEOR score using our approach. The proposed method yields better results for Test-1 but not for Test-2.

	Test-1	Test-2
BLEU-ED-70	77.32	81.34
BLEU-SS-70	81.61	77.14
METEOR-ED-70	91.5	87.35
METEOR-SS-70	92.6	84.55

Table 17: Results automatic evaluation

To gain a deeper understanding of our system's performance, we also performed a manual evaluation on Test-2. We considered the source side (English) of the segments for this evaluation. A native speaker of English performed the manual evaluation. Three different options were given to the evaluator: Semantic similarity is better; Edit-distance is better; or both are similar. When keeping the 70% threshold and ignoring exact matches, we retrieved 266 different fuzzy matched segments. In these 266 segments, 258 segments were tagged as similar, for 6 segments, edit-distance retrieved better and for 2, our semantic similarity approach retrieved better. Some of the examples from Test-2 are given in Table 18.

Example 1 shows our approach (SS) performed better, while examples 2 and 3 show edit-distance (ED) performed better.

The initial results, as stated earlier, show comparable results. Although, the approach does not perform better overall, there are several factors, which should be taken into consideration. The genre of the training set and test set were very different. The SICK dataset

1	Input	For the purposes of this Regulation :
	ED	For the purpose of this demonstration :
	SS	For the purposes of this Regulation the following definitions shall apply :
2	Input	This Decision shall enter into force on the date of its publication in the Official Journal of the European Union .
	ED	This Decision shall enter into force on the day of its publication in the Official Journal of the European Union .
	SS	This Decision shall enter into force on the date of its adoption .
3	Input	The Commission sought and verified all information deemed necessary for the determination of dumping .
	ED	The Commission sought and verified all the information deemed necessary for the purposes of the review .
	SS	The Commission sought and verified all the information provided by interested parties and deemed necessary for the determination of dumping , resulting injury and Union interest .

Table 18: Examples from Test-2

consists of simple sentences extracted mostly from image captions while DGT-TM corpus has much larger and complex sentences from mainly legal domain. The average words per segment for TM is 27.9 and for input is 32.54 for test set, whereas for SICK training dataset average words per sentence is only 9.63.

4.3 Conclusion

In this section, we suggested an approach to employ a semantic similarity system in a TM framework. Our initial experiment shows some positive indication in this direction with comparable results to the basic edit distance. The approach as it is can be used for any language pair where the source language is English. For the other language pairs, the approach can be extended if there are certain language dependent resources (like parser, parts of speech tagger, lemmatiser and CPA dictionary) used in our approach are available for the source language of the language pair.

5 Conclusion

In this document, we have presented our work on improving translation memory matching and retrieval. We have presented related work on sentence level similarity computation used in various NLP tasks like semantic textual similarity, textual entailment, machine translation evaluation and translation memory. We have presented two novel approaches to incorporate semantic information in TM matching and retrieval. Our work present a feasible approach to implement paraphrasing TM matching and retrieval. Our work suggests that paraphrasing is very useful in matching and retrieval. It is evident from the experiments that paraphrasing not only improves the retrieval but also the quality of retrieved segments. Another approach implementing semantic textual similarity shows some positive indication. The approach obtained comparable results to the basic edit distance despite several limitations. In future, we would like to explore techniques inspired from deep learning for TM matching and retrieval.

List of Related Publications

Gupta, R., Béchara, H., Maarouf, I. E., & Orăsan, C. (2014). UoW: NLP techniques developed at the University of Wolverhampton for Semantic Similarity and Textual Entailment. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval-2014).*

Gupta, R., Bechara, H., & Orăsan, C. (2014). Intelligent Translation Memory Matching and Retrieval Metric Exploiting Linguistic Technology. In *Proceedings of the Translating and Computer 36* (pp. 86–89).

Gupta, R., & Orăsan, C. (2014). Incorporating Paraphrasing in Translation Memory Matching and Retrieval. In *Proceedings of the European Association of Machine Translation (EAMT-2014).*

Gupta, R., Orăsan, C., Zampieri, M., Vela, M., & Genabith, J. v. (2015). Can translation memories afford not to use paraphrasing? In *Proceedings of the European Association of Machine Translation (EAMT-2015).*

References

Agirre, E., Diab, M., Cer, D., & Gonzalez-Agirre, A. (2012). Semeval-2012 task 6: A pilot on semantic textual similarity. In *Proceedings of the sixth international workshop on semantic evaluation, in conjunction with the first joint conference on lexical and computational semantics* (pp. 385–393).

Alabbas, M., & Ramsay, A. (2013). Natural language inference for Arabic using extended tree edit distance with subtrees. *Journal of Artificial Intelligence Research*, *48*, 1–22.

Arthern, P. J. (1978). Machine Translation and Computerized Terminology Systems, A Translator's viewpoint. In *Translating and the computer: Proceedings of a seminar* (pp. 77–108).

Aziz, W., Castilho, S., & Specia, L. (2012). PET: a Tool for Post-editing and Assessing Machine Translation. *Language Resources and Evaluation*.

Bannard, C., & Callison-Burch, C. (2005). Paraphrasing with bilingual parallel corpora. In *Proceedings of the 43rd annual meeting on association for computational linguistics* (pp. 597–604).

Bär, D., Biemann, C., Gurevych, I., & Zesch, T. (2012). Ukp: Computing semantic textual similarity by combining multiple content similarity measures. In *First joint conference on lexical and computational semantics, association for computational linguistics* (pp. 435–440).

Bertoldi, N., Cettolo, M., & Federico, M. (2013). Cache-based Online Adaptation for Machine Translation Enhanced Computer Assisted Translation. In *Proceedings of the xiv machine translation summit* (pp. 35–42).

Bille, P. (2005, June). A survey on tree edit distance and related problems. *Theoretical Computer Science*, *337*(1-3), 217–239. doi: 10.1016/j.tcs.2004.12.030

Bojar, O., Buck, C., Federmann, C., Haddow, B., Koehn, P., Leveling, J., … Tamchyna, A. (2014, June). Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the ninth workshop on statistical machine translation* (pp. 12–58). Baltimore, Maryland, USA: Association for Computational Linguistics. Retrieved from `http://www.aclweb.org/anthology/W/W14/W14-3302`

Bradbury, J., & El Maarouf, I. (2013). An empirical classification of verbs based on Semantic Types: the case of the 'poison' verbs. In *Proceedings of the joint symposium on semantic processing. textual inference and structures in corpora* (p. 70-74).

Carl, M., & Hansen, S. (1999). Linking translation memories with example-based machine translation. In *Proceedings of machine translation summit vii* (pp. 617–624).

Chang, C.-C., & Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, *2*, 27:1–27:27.

Dandapat, S. (2012). *Mitigating the Problems of SMT using EBMT Sandipan Dandapat*. Unpublished doctoral dissertation, Dublin City University.

Denkowski, M., & Lavie, A. (2014). Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the eacl 2014 workshop on statistical machine translation.*

Doddington, G. (2002). Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the second international conference on human language technology research* (pp. 138–145).

Dolan, B., Quirk, C., & Brockett, C. (2004). Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the 20th international conference on computational linguistics* (p. 350).

Ganitkevitch, J., Benjamin, V. D., & Callison-Burch, C. (2013). Ppdb: The paraphrase database. In *Proceedings of naacl-hlt* (pp. 758–764). Atlanta, Georgia.

Gupta, R., Béchara, H., Maarouf, I. E., & Orăsan, C. (2014). UoW: NLP techniques developed at the University of Wolverhampton for Semantic Similarity and Textual Entailment. In *Proceedings of the 8th international workshop on semantic evaluation (semeval-2014).*

Hanks, P. (2013). *Lexical analysis: Norms and exploitations*. Mit Press.

Heilman, M., & Smith, N. A. (2010). Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *The 2010 annual conference of the north american chapter of the acl* (pp. 1011–1019).

Hodász, G., & Pohl, G. (2005). MetaMorpho TM: a linguistically enriched translation memory. In *In international workshop, modern approaches in translation technologies.*

Islam, A., & Inkpen, D. (2008, July). Semantic text similarity using corpus-based word similarity and string similarity. *ACM Transactions on Knowledge Discovery from Data*, *2*(2), 1–25. doi: 10.1145/1376815.1376819

Koehn, P. (2005). Europarl: A parallel corpus for statistical machine translation. In *Mt summit* (Vol. 5, pp. 79–86).

Koehn, P., & Senellart, J. (2010). Convergence of translation memory and statistical machine translation. In *Proceedings of amta workshop on mt research and the translation industry* (pp. 21–31).

Koponen, M., Aziz, W., Ramos, L., & Specia, L. (2012). Post-editing time as a measure of cognitive effort. In *Workshop on post-editing technology and practice in amta 2012* (pp. 11–20).

Lagoudaki, E. (2006). Translation Memories Survey 2006: Users' perceptions around TM use. In *Proceedings of translating and the computer 28* (pp. 1–29). London: Aslib.

Langlais, P., & Lapalme, G. (2002). Trans type: Development-evaluation cycles to boost translator's productivity. *Machine Translation*, *17*(2), 77–98.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady* (Vol. 10, pp. 707–710).

Li, Y., McLean, D., Bandar, Z. A., O'shea, J. D., & Crockett, K. (2006). Sentence similarity based on semantic nets and corpus statistics. *Knowledge and Data Engineering, IEEE Transactions on*, *18*(8), 1138–1150.

Marelli, M., Menini, S., Baroni, M., Bentivogli, L., Bernardi, R., & Zamparelli, R. (2014). A sick cure for the evaluation of compositional distributional semantic models. In *Proceedings of lrec 2014.*

Marsi, E., Moen, H., Bungum, L., Sizov, G., Gambäck, B., & Lynum, A. (2013). Ntnu-core: Combining strong features for semantic similarity. In *Second joint conference on lexical and computational semantics (*sem)* (pp. 66–73).

Mitkov, R. (2008). Improving Third Generation Translation Memory systems through identification of rhetorical predicates. In *Proceedings of langtech'2008.*

Mohri, M. (2003). Edit-Distance of Weighted Automata: General Definitions and Algorithms. *International Journal of Foundations of Computer Science*, *14*(6), 957–982.

doi: 10.1142/S0129054103002114

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the acl* (pp. 311–318).

Pekar, V., & Mitkov, R. (2007). New Generation Translation Memory: Content-Sensivite Matching. In *Proceedings of the 40th anniversary congress of the swiss association of translators, terminologists and interpreters.*

Petrov, S., Barrett, L., Thibaux, R., & Klein, D. (2006). Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the coling/acl* (pp. 433–440).

Planas, E., & Furuse, O. (1999). Formalizing Translation Memories. In *Proceedings of the 7th machine translation summit* (pp. 331–339).

Planas, E., & Furuse, O. (2000). Multi-level similar segment matching algorithm for translation memories and example-based machine translation. In *Proceedings of the 18th conference on computational linguistics* (Vol. 2, pp. 621–627).

Porter, M. F. (1980). An algorithm for suffix stripping. *Program*, *14*(3), 130–137.

Roukos, S., Graff, D., & Melamed, D. (1995). Hansard French/English. *Linguistic Data Consortium*. Retrieved from `https://catalog.ldc.upenn.edu/LDC95T20`

Simard, M., & Fujita, A. (2012). A Poor Man ’ s Translation Memory Using Machine Translation Evaluation Metrics . In *Proceedings of the tenth conference of the association for machine translation in the americas.*

Simard, M., & Isabelle, P. (2009). Phrase-based machine translation in a computer-assisted translation environment. In *Proceedings of the twelfth machine translation summit* (pp. 120–127).

Snover, M., Dorr, B., Schwartz, R., Micciulla, L., & Makhoul, J. (2006). A study of translation edit rate with targeted human annotation. In *Proceedings of association for machine translation in the americas* (pp. 223–231).

Snover, M., Madnani, N., Dorr, B., & Schwartz, R. (2008). Terp system description. In *Metricsmatr workshop at amta.*

Sousa, S. C. d., Aziz, W., & Specia, L. (2011). Assessing the post-editing effort for automatic and semi-automatic translations of dvd subtitles. In *Proceedings of recent advances in natural language processing* (pp. 97–103).

Steinberger, R., Eisele, A., Klocek, S., Pilos, S., & Schlüter, P. (2012). DGT-TM: A freely available Translation Memory in 22 languages. *LREC*, 454–459. Retrieved from `http://www.mt-archive.info/LREC-2012-Steinberger.pdf`

Tezcan, A., & Vandeghinste, V. (2011). SMT-CAT integration in a Technical Domain : Handling XML Markup Using Pre & Post-processing Methods. In *Proceedings of european association of machine translation* (pp. 55–62).

Tiedemann, J. (2009). News from OPUS - A collection of multilingual parallel corpora with tools and interfaces. In N. Nicolov, K. Bontcheva, G. Angelova, & R. Mitkov (Eds.), *Recent advances in natural language processing* (Vol. V, pp. 237–248). Borovets, Bulgaria: John Benjamins, Amsterdam/Philadelphia.

Utiyama, M., Neubig, G., Onishi, T., & Sumita, E. (2011). Searching Translation Memories for Paraphrases. In *Machine translation summit xiii* (pp. 325–331).

Vela, M., Neumann, S., & Hansen-Schirra, S. (2007). Querying multi-layer annotation and alignment in translation corpora. In *Proceedings of the corpus linguistics conference cl.*

Wang, K., Zong, C., & Su, K.-Y. (2014, August). Dynamically integrating cross-domain translation memory into phrase-based machine translation during decoding. In *Proceedings of coling 2014, the 25th international conference on computational linguistics: Technical papers* (pp. 398–408). Dublin, Ireland: Dublin City University and Association for Computational Linguistics. Retrieved from `http://www.aclweb.org/anthology/C14-1039`

Wang, M., & Cer, D. (2012). Stanford: probabilistic edit distance metrics for STS. In *Proceedings of the first joint conference on lexical and computational semantics* (pp. 648–654). Retrieved from `http://dl.acm.org/citation.cfm?id=2387746`

Whyman, E. K., & Somers, H. L. (1999). Evaluation metrics for a translation memory system. *Software-Practice and Experience*, *29*(14), 1265–84.

Zhang, K., & Shasha, D. (1989). Simple Fast Algorithms for the Editing Distance between Trees and Related Problems. *SIAM Journal on Computing*, *18*(6), 1245–1262. doi: 10.1137/0218082