Noname manuscript No.
(will be inserted by the editor)

Improving Translation Memory Matching and Retrieval using Paraphrases

Rohit Gupta · Constantin Orăsan · Marcos Zampieri · Mihaela Vela · Josef van Genabith · Ruslan Mitkov

Received: date / Accepted: date

Abstract Most of the current Translation Memory (TM) systems work on string level (character or word level) and lack semantic knowledge while matching. They use simple edit-distance calculated on surface-form or some variation on it (stem, lemma), which does not take into consideration any semantic aspects in matching. This paper presents a novel and efficient approach to incorporating semantic information in the form of paraphrasing in the edit-distance metric. The approach computes edit-distance while efficiently considering paraphrases using dynamic programming and greedy approximation. In addition to using automatic evaluation metrics like BLEU and METEOR, we have carried out an extensive human evaluation in which we measured post-editing time, keystrokes, HTER, HMETEOR, and carried out three rounds of subjective evaluations. Our results show that paraphrasing substantially improves TM matching and retrieval, resulting in translation performance increases when translators use paraphrase-enhanced TMs.

Rohit Gupta
RGCL, RIILP, University of Wolverhampton, Stafford Street, Wolverhampton WV11LY, UK
E-mail: r.gupta@wlv.ac.uk

Constantin Orăsan
RGCL, RIILP, University of Wolverhampton, Stafford Street, Wolverhampton WV11LY, UK
E-mail: c.orasan@wlv.ac.uk

Marcos Zampieri
Saarland University and DFKI, Saarbrücken 66123, Germany
E-mail: marcos.zampieri@uni-saarland.de

Mihaela Vela
Saarland University, Saarbrücken 66123, Germany
E-mail: m.vela@mx.uni-saarland.de

Josef van Genabith
Saarland University and DFKI, Saarbrücken 66123, Germany
E-mail: josef.van_genabith@dfki.de

Ruslan Mitkov
RGCL, RIILP, University of Wolverhampton, Stafford Street, Wolverhampton WV11LY, UK
E-mail: r.mitkov@wlv.ac.uk

1 Introduction

Apart from retrieving exact matches, one of the core features of a Translation Memory (TM) system is the retrieval of previously translated similar segments for post-editing in order to avoid translation from scratch when an exact match is not available. However, this retrieval process is generally limited to edit-distance based measures operating on surface-form (or sometimes stem/lemma) matching. Most commercial systems use edit distance (Levenshtein, 1966) or some variation of it, e.g. the open-source TM OmegaT[1] employs word-based Levenshtein edit distance with some extra preprocessing. This preprocessing typically involves tokenisation, removing punctuation, removing stop words and stemming. Although these measures provide a strong baseline, they are not sufficient to capture semantic similarity between segments as judged by humans. This results in uneven post-editing time by translators for the same fuzzy match scored segments and non-retrieval of semantically similar segments. 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, second segment may not be retrieved given the first in current TM systems as they have only 57% similarity based on word based Levenshtein edit distance as implemented in OmegaT, even though one segment is a paraphrase of the other 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.

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

This paper presents a novel approach to improve matching and retrieval in TM using paraphrasing based on dynamic programming and greedy approximation techniques. Using this approach, the fuzzy match score between segments can be calculated in polynomial time despite the inclusion of paraphrases. For example, if the translation memory used has a segment "What is the actual aim of this practice?" and the paraphrase database has paraphrases "the actual" $\Rightarrow$ "the real" and "aim of this" $\Rightarrow$"goal of this", for the input sentence "What is the real goal of this mission ?", the approach will give a 89.89% fuzzy match score (only one word, "practice", needs to be replaced by "mission") rather than 66.66% using simple word-based edit-distance.

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

Furthermore, the approach classifies paraphrases into different types for efficient implementation based on matching words between the source and corresponding paraphrase. For example, the paraphrase “aim of this” ⇒“goal of this” has the “of this” string in common. We make use of this fact for efficient implementation.

Our full implementation is available at Github.[2] Our preliminary work related to this paper has been published in (Gupta and Orsan, 2014) and (Gupta et al, 2015).

The rest of the paper is structured as follows: Section 2 gives a brief overview of the related work. Our approach is described in Section 3 and the methodology used for evaluation in Section 4. In Section 5 we present the results of our experiments and we finish the paper with our conclusions.

2 Related Work

Several researchers have used semantic or syntactic information in TMs, but the approaches were too inefficient for a large TM. In addition, evaluations were rather small and generally limited to subjective evaluation carried out by the authors. This makes it hard to judge how much a semantically informed TM matching system can benefit a translator.

Planas and Furuse (1999), Macklovitch and Russell (2000), Somers (2003), Hodász and Pohl (2005), Pekar and Mitkov (2007), and Mitkov and Corpas (2008) pointed out the need for similarity calculations in TMs beyond surface-form comparisons. Macklovitch and Russell (2000) explained that using NLP techniques like named entity recognition and morphological processing can improve matching in TM. Somers (2003) highlighted the need for more sophisticated matching techniques that include linguistic knowledge like inflection paradigms, synonyms and grammatical alternations. Both Planas and Furuse (1999) and Hodász and Pohl (2005) proposed the use of lemmas and parts of speech along with surface-form comparison. Hodász and Pohl (2005) also extend the matching process to a sentence skeleton where noun phrases are either tagged by a translator or by a heuristic NP aligner developed for English-Hungarian translation. Planas and Furuse (1999) tested a prototype model on 50 sentences from the software domain and 75 sentences from a journal with TM sizes of 7,192 and 31,526 segments respectively. A fuzzy match retrieved was considered usable if less than half of the words required editing to match the input sentence. The authors concluded that the approach gives more usable results compared to Trados Workbench (an industry-standard commercial system) used as a baseline. Hodász and Pohl (2005) claimed that their approach stored simplified patterns and hence made finding a match in the TM more likely. Pekar and Mitkov (2007) presented an approach based on syntactic transformation rules. On evaluation of the prototype model using a query sentence, the authors found that the syntactic rules help in retrieving better segments.

Clark (2002) proposed using alignments between the source words, phrases or characters and the target words, phrases or characters of the TM. This alignment information can be used to improve matching and translation of other similar segments.

[2] https://github.com/rohitguptacs/TMAdvanced

Recently, work by Utiyama et al (2011) presented an approach that used paraphrasing in TM matching and retrieval. They proposed an approach using a finite state transducer. They evaluate the approach with one translator and find that paraphrasing is useful for TM both in terms of precision and recall of the retrieval process. However, their approach limits TM matching to exact matches only. In statistical machine translation, Onishi et al (2010) and Du et al (2010) use paraphrasing lattices to improve MT by gaining more coverage.

Simard and Fujita (2012) used different MT evaluation metrics for TM similarity calculation as well as to test the quality of retrieval. For most of the metrics, the authors found that the metric which was used in evaluation gave a better score to itself (e.g. BLEU gave highest score to matches retrieved using BLEU as similarity measure). Timonera and Mitkov (2015) showed that clause splitting as a preprocessing stage significantly improves matching and retrieval.

Keystroke and post-editing time analysis are not new for TM and MT. Keystroke analysis has been used to judge translators' productivity (Langlais and Lapalme, 2002; Whyman and Somers, 1999). Koponen et al (2012) suggested that post-editing time reflects the cognitive effort in post-editing the MT output. de Sousa et al (2011) evaluated different MT system performances against translation from scratch. Their study also concluded that subjective evaluations of MT system output correlates with the post-editing time needed. Zampieri and Vela (2014) used post-editing time to compare TM and MT translations.

3 Our Approach

In this section, we present our approach to include paraphrasing in the TM matching and retrieval process. Section 3.1 describes the paraphrase dataset used for experiments, Section 3.2 describes the classification of paraphrases which is one of the important steps in our approach, Section 3.3 describes the matching steps of our approach, Section 3.4 describes the construction of the paraphrasing lattice, Section 3.5 describes filtering steps used to speed up the matching and retrieval process, Section 3.6 presents our edit-distance-with-paraphrasing algorithm and Section 3.7 analyses the complexity of our algorithm.

3.1 Paraphrase Corpus

We used the PPDB 1.0 paraphrases database (Ganitkevitch et al, 2013) for our work. This database contains lexical, phrasal and syntactic paraphrases automatically extracted using a large collection of parallel corpora. The paraphrases in this database are constructed using a 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 aim at more coverage. The smallest package (S) contains 600 thousand lexical and phrasal paraphrases while the largest package (XXXL) contains 68 million. 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 the segments retrieved using paraphrasing and at the same time gain some coverage. L contains 3 million paraphrases. We removed paraphrases with punctuation, numbers or any special characters and retained the remaining 2 million paraphrases for our work.

3.2 Classification of Paraphrases

We classified paraphrases obtained from PPDB 1.0 into four types on the basis of the number of words in the source and target phrases:

1. Paraphrases having one word in both the source and target sides, e.g. "period" ⇔"duration"
2. Paraphrases having multiple words on both sides but differing in one word only, e.g. "in the period" ⇔ "during the period"
3. Paraphrases having multiple words, but the same number of words on both sides, e.g. "laid down in article" ⇔ "set forth in article"
4. Paraphrases in which the number of words in the source and target sides differ, e.g. "a reasonable period of time to" ⇔ "a reasonable period to"

In our classification, Type 1 paraphrases are one-word paraphrases and Type 2 paraphrases can be reduced to one-word paraphrases after considering the context when storing in the TM. For Type 1 and Type 2, we obtain the same accuracy as using the trivial method but in polynomial time (see Section 3.3 for details). Paraphrases of Type 3 and Type 4 require additional attention because they still remain multiword paraphrases after reduction (see Table 1). We use dynamic programming along with greedy approximation to implement matching with paraphrases in polynomial time.

Type	**Paraphrases**	**Reduced form**
Type 1	period ⇔ duration	period ⇔ duration
Type 2	in the period ⇔ during the period	in ⇔ during
Type 3	laid down in article ⇔ set forth in article	laid down ⇔ set forth
Type 4	a reasonable period of time to ⇔ a reasonable period to	of time to ⇔ to

Table 1: Example: Paraphrases in the reduced form

Type 1 paraphrases appear to be simple synonyms (as e.g. in WordNet) but they are better than simple synonyms. Paraphrases in the PPDB dataset are extracted using a statistical method. They are retrieved as paraphrases because there was enough confidence to retrieve them even if they are single words. In addition, in typical TM settings we are interested in a more than 70% fuzzy match, which up to a certain extent makes sure that, apart from the paraphrases themselves, some other context words appear in both sides of the matching segements when a match is found in the TM. Furthermore, we also use filtering steps (explained in Section 3.5) to restrict the amount of paraphrasing allowed per segment.

3.3 Matching Steps

There are two options for incorporating paraphrasing in a typical TM matching 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 on 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.

This being said, paraphrasing the input has its own advantage and may be a better option in some scenarios. In general, input files are much smaller than TMs. Therefore, paraphrasing the input instead of the TM can save space.

Our approach is based on the following steps:

1. Read the Translation Memories available
2. Classify paraphrases according to the four types presented in Section 3.2
3. Store all the paraphrases for each segment in the TM in their reduced forms according to the process presented in Section 3.4
4. Read the file that needs to be translated
5. For each segment in the input file get the potential segments for paraphrasing in the TM according to the filtering steps of Section 3.5, then search for the most similar segment based on the approach described in Section 3.6 and retrieve the most similar segment if above a predefined threshold

3.4 Storing Paraphrases

TM entries ae augmented with paraphrases in their reduced forms because when capturing paraphrases for a particular segment, the context has already been considered and there is therefore no need for it to be considered again while calculating edit-distance. We first obtain the applicable paraphrases for a segment from the paraphrase database and then reduce it. We store only the non-matching substring instead of the whole paraphrase. Suppose we have paraphrases as given in Table 2. The TM segment and the paraphrases stored in a reduced form are given in Figure 1 (TM segment (i), TM lattice after considering paraphrases (ii)).

the period laid down in	the period referred to in
laid down in article	provided for by article
the period	the time
the period	the duration
in article	under article

Table 2: Example: Paraphrases

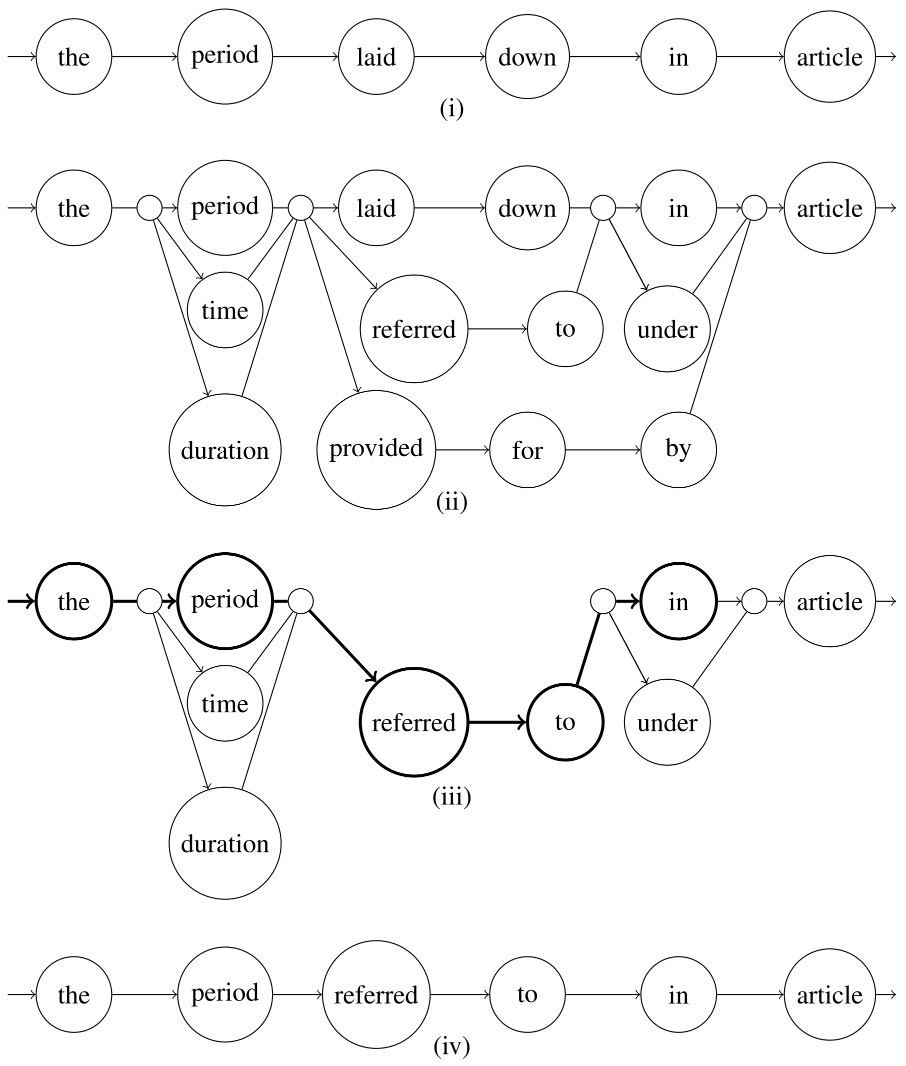
the period laid down in article
(i)
the period time duration laid referred to provided for down in under by article
(ii)
the period time duration referred to in under article
(iii)
the period referred to in article
(iv)

Fig. 1: (i) TM segment, (ii) TM lattice after considering paraphrases, (iii) TM lattice after edit-distance calculation of first five tokens, (iv) Input test segment

3.5 Filtering

Before matching a new string against the paraphrase augmented TM entries begins, several filtering steps are applied to each input segment. The purpose of this filtering process is to remove unnecessary candidates from participating in the paraphrasing process and speed up the process. Our filtering steps for obtaining potential candidates for paraphrasing are as follows:

1. LENFILTER: We first filter out the segments based on length because if segments differ considerably in length, the edit-distance will also differ correspondingly. In our case, the threshold for length was 39%. TM segments are discarded if the TM segments are shorter than 39% of the input or vice-versa.
2. SIMFILTER: Next, we filter the segments based on baseline edit-distance similarity. TM segments which have a similarity below a certain threshold are removed. In our case, the threshold was 39%.
3. MAXFILTER: Next, after filtering the candidates with the above two steps we sort the remaining segments in decreasing (non-increasing) order of baseline edit-distance similarity and pick the top 100 segments.
4. BEAMFILTER: Finally segments within a certain range of similarity with the most similar segment were selected for paraphrasing. In our case, the range is 35%. This means that if the most similar segment has 95% similarity, segments with a similarity below 60% are discarded.

The filtering thresholds were determined empirically by running the proposed method with various settings on the DGT-TM (Steinberger et al, 2012) English-French corpus. The detailed experiments are described in Gupta and Orsan (2014), where we used LENFILTER: 49%, SIMFILTER:49%, MAXFILTER:100 and BEAMFILTER: 35%. For the experiments on the Europarl corpus (Koehn, 2005) presented in this paper, we decided to lower the LENFILTER and SIMFILTER to 39% because the number of retrieved segments was low in comparison to DGT-TM. Furthermore, the Europarl corpus contains spoken data (often scripted) and is more likely to contain paraphrases than the DGT-TM corpus which contains text from the legal domain.

3.6 Edit-Distance with Paraphrasing

For our implementation we use a basic edit-distance algorithm (Levenshtein, 1966), which is a word-based edit-distance with cost 1 for insertion, deletion and substitution. Algorithm 1 describes the basic edit-distance procedure. The similarity is calculated by normalising edit-distance by the length of the larger segment.

We employed simple edit-distance as a baseline and adapted it to incorporate paraphrasing. 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 paraphrases of Type 1 are single word paraphrases and Type 2 have a unique property in that they can be reduced to single word paraphrases by removing matching words.

The algorithm implementing edit-distance-with-paraphrasing is given in Algorithm 4. In Algorithm 4, *InputSegment* is the segment that we want to translate and *TMLattice* is the TM lattice with paraphrases. The basic procedure as given in Algorithm 1 works by comparing, one by one, each token in the input segment with each token in the TM segment. This procedure makes use of previous edit-distance computations to optimise the edit-distance globally (for the whole sentence).

To implement the paraphrases of Types 1 and 2, we extend this procedure by searching (a successful search indicates a match) in a list of paraphrases (reduced

Algorithm 1 Basic Edit-Distance

```
1: procedure EDIT-DISTANCE(InputSegment, TMS)
2:     $M \leftarrow$ length of TMS                              ▷ Initialise $M$ with length of TM segment
3:     $N \leftarrow$ length of InputSegment                     ▷ 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:            $cost \leftarrow$ GETCOST(InputToken, TMToken)    ▷ GETCOST procedure is defined in 2
11:            $D[i,j] \leftarrow minimum(D[i-1,j]+1, D[i,j-1]+1, D[i-1,j-1]+cost)$    ▷ store minimum of insertion, substitution and deletion
12:        end for
13:    end for
14:    Return $D[N,M]$                                           ▷ Return minimum edit-distance
15: end procedure
```

Algorithm 2 Compute cost for basic edit-distance

```
1: procedure GETCOST(InputToken, TMToken)
2:     $cost \leftarrow 1$
3:     if $InputToken = TMToken$ then                ▷ match InputToken with TMToken
4:         $cost \leftarrow 0$                        ▷ Substitution cost if matches
5:     else
6:         $cost \leftarrow 1$                        ▷ Substitution cost if not matches
7:     end if
8:     Return $cost$                                 ▷ Return minimum edit-distance
9: end procedure
```

Algorithm 3 Compute Cost Using Type 1 and Type 2 Paraphrases

```
1: procedure GETCOSTPARAPHRASE12(InputToken,TMToken)
2:     $cost \leftarrow 1$
3:     $OneWordPP \leftarrow$ get Type 1 and Type 2 paraphrases associated with TMToken including TMToken itself
4:     if $InputToken \in OneWordPP$ then            ▷ applying Type 1 and Type 2 paraphrasing
5:         $cost \leftarrow 0$
6:     end if
7:     Return $cost$                                 ▷ Return minimum edit-distance
8: end procedure
```

single tokens) associated with the TM token in addition to comparing the TM token with the input token. For the example given in Figure 1 (Figure 1(iv) shows the input test segment and Figure 1(ii) shows the TM lattice), if a word from the input segment matches any of the words "period", "time" or "duration", the cost of substitution will be 0. The basic edit-distance procedure can be extended to incorporate Type 1 and Type 2 paraphrases by using a new GETCOSTPARAPHRASE12 procedure given in Algorithm 3 instead of the GETCOST procedure of Algorithm 2. In Algorithm 4, *lines 11 to 16* executes when Type 3 and Type 4 paraphrases are not available (e.g. edit-distance calculation of the second token "period"). Table 3 illustrates the edit-distance calculation of the first five tokens of the Input and TM segment with paraphrasing of the example given in Figure 1. The second column of the table represents the input segment and the second row represents the TM segment along with the paraphrases. In Table 3, if a word from the input segment matches any of the words "period", "time" or "duration", the cost of substitution will be 0.

	j	0	1	2				3	4				5
i		#	the	period duration time	laid	down	in under	referred	to	provided	for	by	in under
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 3: Edit-Distance Calculation

Algorithm 4 Edit-distance with Paraphrasing Procedure

```
1: procedure EDIT-DISTANCEPP(InputSegment, TMLattice)
2:     $M \leftarrow$ length of TM Segment                    ▷ number of tokens in the TM segment
3:     $N \leftarrow$ length of InputSegment                  ▷ number of tokens in the Input segment
4:     $D[i,0] \leftarrow i$ for $0 \le i \le N$               ▷ initialise two dimensional matrix $D$
5:     $D[0,j] \leftarrow j$ for $0 \le j \le (M + p')$ where $p'$ accounts for increase in the TM segment length because of paraphrasing
6:     $decisionPoint \leftarrow 0$ , $j \leftarrow 1$
7:     $cost \leftarrow 1$                                    ▷ initialisation of the substitution cost
8:     while $j \le M$ do
9:         $TMToken \leftarrow$ TMLattice$_j$        ▷ getting current TM token to process, e.g. $3^{rd}$ token "laid"
10:        if there are no paraphrases of type 3 and type 4 starting from $TMToken$ or $decisionPoint \ge N$ then
11:            $decisionPoint \leftarrow decisionPoint + 1, j \leftarrow j + 1$
12:            for $i \leftarrow 1...N$ do
13:                $InputToken \leftarrow Input_i$
14:                $cost \leftarrow$ GETCOSTPARAPHRASE12($InputToken, TMToken$)  ▷ GETCOSTPARAPHRASE12 procedure is defined in Algorithm 3
15:                $D[i, decisionPoint] \leftarrow minimum(D[i, decisionPoint - 1] + 1, D[i - 1, decisionPoint] + 1, D[i - 1, decisionPoint - 1] + cost)$
16:            end for
17:        else
18:            $prevDistance \leftarrow D[decisionPoint, decisionPoint]$
19:            $DP \leftarrow$ calculate edit-distance of each paraphrase and longest source phrase with $Input$ using $D$  ▷ uses $D$ for first word, consider Type 1 and Type 2 paraphrases for the source phrase
20:            selectedPhrase $\leftarrow$ select a minimum edit-distance paraphrase or a source phrase  ▷ source phrase is preferred in case of a tie between a paraphrase and the corresponding source
21:            $curDistance \leftarrow$ edit-distance of the selectedPhrase
22:            if selectedPhrase is a paraphrase then
23:                $j \leftarrow j +$ length of the source phrase corresponding to selectedPhrase
24:                $decisionPoint \leftarrow decisionPoint +$ length of selectedPhrase
25:                update $D$ using $DP$
26:            else if selectedPhrase is a source phrase and $curDistance = prevDistance$ then  ▷ $true$ if the source phrase is exactly matching
27:                $j \leftarrow j +$ length of selectedPhrase
28:                $decisionPoint \leftarrow decisionPoint +$ length of selectedPhrase
29:                update $D$ using $DP$
30:            else
31:                $j \leftarrow j + 1, decisionPoint \leftarrow decisionPoint + 1$
32:                update $D$ using $DP$
33:            end if
34:        end if
35:    end whileReturn $D[N, decisionPoint]$
36: end procedure
```

In Algorithm 4, *Lines 18 to 34* account for a case where Type 3 and Type 4 paraphrases are available. For paraphrases of Types 3 and 4 the algorithm takes the decision locally at the point where all paraphrases finish. Table 3 shows that, 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 (compared to "laid down" (2), "laid down in" (2), "provided for by" (3)). The last column of Table 3 ($j = 5$) shows the edit-distance calculation of the next token "in" after selecting "referred to". As the algorithm has selected "referred to" as a preferred paraphrase, the value for column "in" will be updated using only "to" as a previous column.

Lines 23, 27 and 31 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 25, 29 and 32* update the matrix D.

Figure 1(iii) shows the preferred path in bold after considering paraphrases. Figure 1(iii) also shows that we retain the Type 1 and Type 2 paraphrases for further edit-distance calculation.

3.7 Computational Considerations

The time complexity of the basic edit-distance procedure is $O(m\,n)$ 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 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 $O(m\,n\,log(p))$. Employing paraphrasing of Type 3 and Type 4 further increases the edit-distance complexity to $O(l\,m\,n(log(p)+q))$, where q is the number of Type 3 and Type 4 paraphrases stored per token and l is the average length of a Type 3 and Type 4 paraphrase. Assuming the source and target segment are of the same length n and each token of the segment stores paraphrases of length l, the complexity will be $O((q+log(p))n^2\,l)$. 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 $c \times O(n^2\,l)$. 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.

4 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. One cannot measure the gain unless retrieval benefits the translator.

When we use paraphrasing in the matching and retrieval process, the fuzzy match score of a paraphrased segment is increased, which results in the retrieval of more segments at a particular threshold. This increment in retrieval can be classified into 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 a 65% and B has a 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% and B has 70% fuzzy score;
2. the score of A is less than that of B, for example, A has 75% and B has 80% fuzzy score.

In the first case, paraphrasing does not supersede the existing model and just facilitates it by improving the fuzzy score so that the top segment ranked using edit-distance gets retrieved. However, in the second case, paraphrasing changes the ranking and now the top-ranked segment is different. In this case, the paraphrasing model supersedes the existing simple edit-distance model. This second case also gives a different reference with which to compare. In the experiments reported below, 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 for a human translator to work with. We measure post-editing time (PET), keystrokes (KS) and carried out subjective evaluations to find out to what extent paraphrasing helps. More details about these experiments are given in Section 4.1, Section 4.2 and Section 4.3.

We do not impose any penalty for paraphrasing; this means that if we obtain an exact match (100% fuzzy match) after considering paraphrasing we consider it an exact match. However, we prefer a match with simple edit-distance over paraphrasing in case of ties (because we do not want to retrieve a paraphrase match when we are already getting the same similarity match retrieved by matching at the surface-level using simple edit-distance). The question arises whether we need to impose any penalty for paraphrasing. Is an exact match retrieved using paraphrasing really an exact match? We carried out another human evaluation to assess whether exact matches retrieved after considering paraphrases are really exact matches. More details about this experiment is given in Section 4.4. The rest of this section presents and discusses the results of the experiments.

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

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

In this task, we 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 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 rest after post-editing a segment. These features were ensured by using the PET tool (Aziz et al, 2012). 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 (ED at odd positions and PP at even positions or vice versa). However, the same translator did not post-edit the match retrieved using PP and ED for the same segment; instead 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 translators on this segment. Normalisation is applied to account for slow and fast translators.

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

$$N_{ij} = T_{ij} \times \frac{\text{Avg time on this file by all translators}}{\sum\limits_{j=1}^{m} T_{ij}} \tag{2}$$

In the equations 1 and 2 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, white-space and erase keys pressed. For our analysis, we considered average keystrokes pressed by all translators for each segment.

4.2 Subjective Evaluation with Two Options (SE2)

In this evaluation, we carried out 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 did not know whether the fuzzy matches were obtained using paraphrasing or not. To avoid 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: whether A is better; or B is better. 17 translators participated in this experiment. Finally, the decision of whether 'ED is better' or 'PP is better' was made on the basis of how many translators choose one over the other.

4.3 Subjective Evaluation with Three Options (SE3)

This evaluation is similar to Evaluation SE2 except that we provided one more option to translators. Translators could choose between three options: whether A is better; B is better; or both are equal. Seven translators participated in this experiment.

4.4 Subjective Evaluation on Exact Matches only (SEM)

In this evaluation, our objective was to check whether an 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 11 translators. The translators had to correct the segment and select an option from two options presented: *can not be accepted as it is (post-editing was required)* and *correct translation (no post-editing was required)*.

5 Corpus, Tool and Translators expertise

As TM and test data, we used English-German pairs of the Europarl V7.0 Koehn (2005) corpus with English as the source language and German as the target language. From this corpus we have filtered out segments of fewer than seven words and more than 40 words to create the TM and test datasets. Tokenisation of the English data was done using the Berkeley Tokenizer (Petrov et al, 2006).

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

Table 4: Corpus Statistics

In our experiments, we did not paraphrase any capitalised words (but we change them to lowercase for both baseline and paraphrasing similarity calculation). This was to avoid paraphrasing any named entities. Table 4 shows our corpus statistics.

The translators involved in our experiments were third-year bachelor's or master's translation students who were native speakers of German with English language level C1, in the age group of 21 to 40 years with a majority of female students. Our translators were not experts in any specific technical or legal field. For this reason we did not use such a domain specific corpus. In this way we avoided any bias stemming from familiarity or unfamiliarity with domain-specific terms.

5.1 Familiarisation with the Tool

We used the PET tool (Aziz et al, 2012) for all our human experiments. However, settings were changed depending on the experiment. To familiarise translators with the PET tool we carried out a pilot experiment before the actual experiment with the Europarl corpus. This experiment was done on a corpus (Vela et al, 2007) different from Europarl. 18 segments were used in the pilot experiment. While the findings are not included in this paper, they informed the design of our main experiments.

5.2 Results and Analysis

The retrieval results are given in Table 5 and Table 6. Table 5 presents the results for different cutoff thresholds whereas Table 6 shows results for different intervals for the value of similarity scores. We have chosen the threshold intervals in order to select segments from each range for human evaluation.

Tables 5 and 6 show similarity thresholds (TH) for TM, the total number of segments retrieved using the baseline approach (EditRetrieved), the additional number of segments retrieved using the paraphrasing approach (+ParaRetrieved), the percentage improvement in retrieval obtained over the baseline (%Improve), and the number of segments that changed their ranking and rose to the top because of paraphrasing (RankCh). BLEU-ParaRankCh and METEOR-ParaRankCh represent the BLEU score (Papineni et al, 2002) and METEOR (Denkowski and Lavie, 2014) score over translations retrieved by our approach for segments which changed their ranking and come up to the top because of paraphrasing and BLEU-EditRankCh and METEOR-EditRankCh represent the BLEU score and METEOR score on corresponding translations retrieved by the baseline approach.

TH	**100**	**95**	**90**	**85**	**80**	**75**	**70**
EditRetrieved	117	127	163	215	257	337	440
+ParaRetrieved	17	16	22	33	50	80	101
%Improve	14.53	12.5	13.5	15.35	19.46	23.74	22.9
RankCh	10	11	16	25	38	68	100
BLEU-EditRankCh	26.35	26.14	27.70	21.71	22.37	17.43	13.85
BLEU-ParaRankCh	**51.56**	**47.81**	**43.90**	**31.76**	**26.50**	**20.67**	**16.05**
METEOR-EditRankCh	43.40	44.52	45.59	39.24	39.99	36.03	32.41
METEOR-ParaRankCh	**67.68**	**66.75**	**61.09**	**50.07**	**45.37**	**39.31**	**34.51**

Table 5: Results on Europarl dataset: Automatic Evaluation, Using All Four Types of Paraphrases

	TH	**100**	**[85, 100)**	**[70, 85)**	**[55, 70)**
	EditRetrieved	117	98	225	703
	+ParaRetrieved	17	29	97	311
	%Improve	14.52	29.59	42.92	44.17
	RankCh	10	13	54	202
	BLEU-EditRankCh	26.35	14.35	6.89	5.47
	BLEU-ParaRankCh	**51.56**	**15.46**	**8.45**	**5.83**
	METEOR-EditRankCh	43.40	35.13	25.96	19.99
	METEOR-ParaRankCh	**67.68**	**38.35**	**26.40**	**21.63**

Table 6: Results of Retrieval

Table 5 and 6 show that we obtain improvements on each threshold level and intervals. Table 6 shows that when using paraphrasing we obtain around 14% improvement in retrieval for exact matches and around 30% and 43% improvement in the intervals [85, 100) and [70, 85) compared to the baseline edit-distance, respectively. This clearly shows that paraphrasing significantly improves the retrieval results.

The test sets distribution for human evaluation is given in Table 7. The sets contain randomly selected segments from the additionally retrieved segments using paraphrasing which changed their top ranking. We have chosen the threshold intervals so as to select the segments from each range for the human evaluations. Two sets were created to facilitate the evaluation based on post-editing time and keystrokes (See Section 3.3.3.2). For this evaluation, each translator post-edited only one set.

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

Table 7: Test Sets for Experiments PET, KS, SE2 and SE3

Results for human evaluations (PET, KS, SE2 and SE3) on both sets (Set1 and Set2) are given in Table 8. 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 of translators who prefer the 'ED is better', 'PP is better' and 'Both are equal' options respectively.

5.3 Results: Post-editing Time (PET) and Keystrokes (KS)

Table 8 shows that 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. [3]

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

For Set2, translators made 468.59 keystrokes and 570.6 keystrokes when 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 8.29% time.

In total, combining both the sets, translators made 824.79 keystrokes and 1103.2 keystrokes when editing PP and ED matches, respectively. Translators took 1069.61 seconds for PP as opposed to 1177.77 seconds for ED matches. Therefore, by using PP matches, translators edit 25.23% less, which saves time by 9.18%. In other words, ED matches require 33.75% more keystrokes and 10.11% more time. We observe that the percentage improvement obtained by keystroke analysis is smaller compared to the improvement obtained by post-editing time. One of the reasons for this is that the translator spends a fair amount of time reading a segment before starting editing.

[3] $p<0.05$, one tailed Welch's t-test for PET and KS, χ^2 test for SE2 and SE3. Because of the small sample size for SE3, no significance test was performed on individual segment basis. Segments are different and each segment will take different post-editing time and keystrokes. Therefore, we can not apply t-test on all 30 segments as a whole because it represents 30 different tasks. However, we applied chi square test for subjective evaluations.

	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
Set-1-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
Set-2-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 8: Results of Human Evaluation on Set1 (1-14) and Set2 (15-30)

5.4 Results: Using post-edited references

We also calculated the human-targeted translation error rate (HTER) (Snover et al, 2006) and human-targeted METEOR (HMETEOR) (Denkowski and Lavie, 2014). HTER and HMETEOR was calculated between ED and PP matches presented for post-editing and references generated by editing the corresponding ED and PP match. Table 9 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.[4]

Table 9 shows improvements 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 also observe that while the 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

[4] For HMETEOR, higher is better and for HTER lower is better.

(81.44 vs 80.60 and 17.63 vs 18.71). This suggests that paraphrasing not only brings improvement but may also improve 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 9: Results Using Human Targeted References

5.5 Results: Subjective evaluations

The subjective evaluations also show significant improvements.

In subjective evaluation with two options (SE2) as given in Table 8, from a total of 510 (30×17) replies for 30 segments from both sets by 17 translators, 334 replies tagged ‘PP is better’ and 176 replies tagged ‘ED is better’. [5]

In subjective evaluation with three options (SE3), from a total of 210 (30×7) 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’. [6]

5.6 Results: Segment wise analysis

A segment wise analysis of 30 segments from both sets shows that 21 segments extracted using PP were found to be better according to PET evaluation and 20 segments using PP were found to be better according to KS evaluation. In subjective evaluations, 20 segments extracted using PP were found to be better according to SE2 evaluation whereas 27 segments extracted using PP were found to be better or equally good according to 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 correlate with each other on segment-by-segment basis. ‘!, ‘+ and ‘* next to each segment number in Table 8 indicate conflicting evaluations: ‘!’ denotes that PET and SE2 contradict each other, ‘+’ denotes that KS and SE2 contradict each other and ‘*’ denotes that PET and KS contradict each other. In twelve segments KS evaluation or PET evaluation show PP as statistically significantly better, except for two cases all the evaluations also show them better.[7] For Seg #13 SE3 shows ‘Both are equal’ and for Seg #26, PET is better

[5] statistically significant, χ^2 test, $p < 0.001$

[6] statistically significant, χ^2 test, $p < 0.001$

[7] In this section all evaluations refer to all four evaluations viz PET, KS, SE2 and SE3.

for ED, however for these two sentences also all the other evaluations show PP as better.

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

(1) **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.
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 segments are identical, but the German translations differ. 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 observe 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):

(2) **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. But, post-editing suggests that it takes more time and keystrokes to post-edit the PP compared with ED.

There is one segment, Seg #22, on which all the evaluations show that ED is better. Seg #22 is given below:

(3) **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.

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

The results of subjective evaluation on exact matches (SEM) are given in Table 10.[8] On 10 segments out of 14 segments, seven or more (two-thirds) of the translators

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	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 10: Results of Human Evaluation on Exact Matches

agreed that the segment did not require any post-editing. In the rest of the cases, for two segments (Seg #13 and Seg #15) the judgements were contradictory with half of the translators agreeing and half disagreeing whether the segment needed post-editing. In two other cases (Seg #7 and Seg #9) most of the translators chose to post-edit the segments. The Seg #7 is given below (PPPE represents the most preferred post-edited translation):

(4) **Input** 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:

(5) **Input** (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)

[8] The Seg # 9 was skipped by one of the translator. Therefore, we have 10 evaluators for this segment instead of 11 evaluators for other segments

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

Most of the segments translators agree to accept as it is. This suggests that for a majority of segments, a paraphrasing match can be presented as an exact match.

6 Conclusion

In this paper we have presented an efficient technique based on dynamic programming and greedy approximation to include paraphrasing in the simple edit-distance metric. We conducted both automatic and human evaluations to test our technique and the impact of paraphrasing on TM matching and retrieval. For automatic evaluation, we used automatic evaluation metrics BLEU and METEOR and for human evaluation, we measured post-editing time, keystrokes, HTER, HMETEOR, and carried out three rounds of subjective evaluations. We conclude that paraphrasing significantly improves TM matching and retrieval. We observe improvements of around 30% and 43% for the threshold intervals [85, 100) and [70, 85) respectively and around 23% improvement over the 70 or 75 cutoff threshold. The quality of the retrieved segments is also significantly better, which is evident from all our human translation evaluations. On average on both sets used for evaluation, compared to paraphrasing, simple edit distance takes 33.75% more keystrokes and 10.11% more time when evaluating the segments whose top rank was changed and came up in the threshold intervals because of paraphrasing.

In the future, we will investigate the usefulness of optimising globally using dynamic programming instead of taking the local greedy decision when considering Type 3 & Type 4 paraphrases. We also plan to use neural network techniques for similarity calculation in TM.

Acknowledgements

The research leading to these results has received funding from the People Programme (Marie Curie Actions) of the European Unions Seventh Framework Programme FP7/2007-2013/ under REA grant agreement no. 317471 and the EC-funded project QT21 under Horizon 2020, ICT 17, grant agreement no. 645452.

References

Aziz W, de Sousa SCM, Specia L (2012) PET: a Tool for Post-editing and Assessing Machine Translation. In: Proceedings of The Eight International Conference on Language Resources and Evaluation (LREC 2012), Istanbul, Turkey, pp 3982–3987

Clark JP (2002) System, method, and product for dynamically aligning translations in a translation-memory system. US Patent 6,345,244

Denkowski M, Lavie A (2014) Meteor Universal: Language Specific Translation Evaluation for Any Target Language. In: Proceedings of the Ninth Workshop on Statistical Machine Translation, Baltimore, Maryland, USA, pp 376–380

Du J, Jiang J, Way A (2010) Facilitating Translation Using Source Language Paraphrase Lattices. In: Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, MIT, Massachusetts, USA, pp 420–429

Ganitkevitch J, Van Durme B, Callison-Burch C (2013) PPDB: The Paraphrase Database. In: Proceedings of NAACL-HLT 2013, Atlanta, Georgia, pp 758–764

Gupta R, Orăsan C (2014) Incorporating Paraphrasing in Translation Memory Matching and Retrieval. In: Proceedings of the Seventeenth Annual Conference of the European Association for Machine Translation (EAMT2014), Dubrovnik, Croatia, pp 3–10

Gupta R, Orăsan C, Zampieri M, Vela M, Genabith JV (2015) Can Translation Memories afford not to use paraphrasing? In: Proceedings of the 18th Annual Conference of the European Association for Machine Translation (EAMT), Antalya, Turkey, pp 35 – 42

Hodász G, Pohl G (2005) MetaMorpho TM: a linguistically enriched translation memory. In: Workshop on Modern Approaches in Translation Technologies, Borovets, Bulgaria, pp 26 – 30

Koehn P (2005) Europarl: A parallel corpus for statistical machine translation. In: Proceedings of the 10th Machine Translation Summit, Phuket, Thailand, vol 5, pp 79–86

Koponen M, Aziz W, Ramos L, Specia L (2012) Post-editing time as a measure of cognitive effort. In: Proceedings of the AMTA 2012 Workshop on Post-editing Technology and Practice (WPTP 2012), San Diego, California, pp 11–20

Langlais P, Lapalme G (2002) Trans Type: Development-Evaluation Cycles to Boost Translator's Productivity. Machine Translation 17(2):77–98

Levenshtein VI (1966) Binary codes capable of correcting deletions, insertions, and reversals. In: Soviet physics doklady, vol 10, pp 707–710

Macklovitch E, Russell G (2000) What's been Forgotten in Translation Memory. In: Proceedings of the 4th Conference of the Association for Machine Translation in the Americas on Envisioning Machine Translation in the Information Future, London, UK, pp 137–146

Mitkov R, Corpas G (2008) Improving Third Generation Translation Memory systems through identification of rhetorical predicates. In: Proceedings of LangTech'2008, Rome, Italy

Onishi T, Utiyama M, Sumita E (2010) Paraphrase Lattice for Statistical Machine Translation. In: Proceedings of the ACL 2010 Conference Short Papers, Uppsala, Sweden, pp 1–5

Papineni K, Roukos S, Ward T, Zhu WJ (2002) BLEU: a method for automatic evaluation of machine translation. In: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL), Philadelphia, Pennsylvania, pp 311–318

Pekar V, Mitkov R (2007) New Generation Translation Memory: Content-Sensitive 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 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Sydney, Australia, pp 433–440

Planas E, Furuse O (1999) Formalizing Translation Memories. In: Proceedings of the 7th Machine Translation Summit, Singapore, pp 331–339

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, San Diego, California, USA

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, Cambridge, Massachusetts, USA, pp 223–231

Somers H (2003) Translation memory systems. Computers and Translation: A Translator's Guide 35:31–48

de Sousa SCM, 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, Hissar, Bulgaria, pp 97–103

Steinberger R, Eisele A, Klocek S, Pilos S, Schlüter P (2012) DGT-TM: A freely available Translation Memory in 22 languages. In: Proceedings of the 8th international conference on Language Resources and Evaluation (LREC'2012), Istanbul, Turkey, pp 454–459

Timonera K, Mitkov R (2015) Improving Translation Memory Matching through Clause Splitting. In: Proceedings of the Workshop on Natural Language Processing for Translation Memories (NLP4TM), Hissar, Bulgaria, pp 17–23

Utiyama M, Neubig G, Onishi T, Sumita E (2011) Searching Translation Memories for Paraphrases. In: Proceedings of the 13th Machine Translation Summit, Xiamen, China, 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, Birmingham, UK

Whyman EK, Somers HL (1999) Evaluation metrics for a translation memory system. Software-Practice and Experience 29(14):1265–1284

Zampieri M, Vela M (2014) Quantifying the Influence of MT Output in the Translators' Performance: A Case Study in Technical Translation. In: Workshop on Humans and Computer-Assisted Translation (HaCaT 2014), Gothenburg, Sweden, pp 93 – 98