Incorporating Paraphrasing in Translation Memory Matching and Retrieval

Rohit Gupta and **Constantin Orăsan**
RGCL, Research Institute of Information and Language Processing,
University of Wolverhampton, Stafford Street,
Wolverhampton WV11LY, UK
{R.Gupta, C.Orasan}@wlv.ac.uk

Abstract

Current Translation Memory (TM) systems work at the surface level and lack semantic knowledge while matching. This paper presents an approach to incorporating semantic knowledge in the form of paraphrasing in matching and retrieval. Most of the TMs use Levenshtein edit-distance or some variation of it. Generating additional segments based on the paraphrases available in a segment results in exponential time complexity while matching. The reason is that a particular phrase can be paraphrased in several ways and there can be several possible phrases in a segment which can be paraphrased. We propose an efficient approach to incorporating paraphrasing with edit-distance. The approach is based on greedy approximation and dynamic programming. We have obtained significant improvement in both retrieval and translation of retrieved segments for TM thresholds of 100%, 95% and 90%.

1 Introduction

Translation Memories (TMs) are tools commonly used by professional translators to speed up the translation process. 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 system helps translators by retrieving previously translated segments to extract the relevant match for reuse. TMs also help them in maintaining the consistency with previous work and use of appropriate terminology. 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.

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. So far, most of the research in TM has been carried out mostly in industry with more focus on improving user interface and user experience in general rather than employing language technology to improve matching and retrieval. Recent research (Koehn and Senellart, 2010; Zhechev and Genabith, 2010) on TM focus more on improving machine translation using TMs.

The TMs currently used by translators find matches for a given segment on the basis of surface form. This means that even if a paraphrased segment is available in the TM, the TM systems have no way to retrieve such segments. In this paper we try to mitigate this problem by using existing paraphrase databases. To achieve this, we have incorporated paraphrasing in the TM matching process. A trivial approach to incorporating paraphrasing would be to generate all the possible segments based on paraphrases available. However in this approach the number of segments increases exponentially and hence can not be applied in our task. This paper proposes a greedy approximation and dynamic programming technique to incorporate paraphrasing in the matching algorithm.

2 Paraphrasing for TM

2.1 Existing Work

The idea of incorporating paraphrasing or semantic features at the conceptual level is not new. Work done by (Pekar and Mitkov, 2007) and (Mitkov, 2008) explores the issues in TM systems. Although these works present good insight into TM systems and their limitations, there is no feasible practical implementation proposed to improve them. Another work (Utiyama et al., 2011) incorporates paraphrasing into TM. This approach uses a statistical framework to integrate paraphrasing which requires corpora from the same domain with an abundance of similar segments. 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. In addition, the approach was used to get exact matches only. In SMT, Onishi et al. (2010) and Du et al. (2010) use paraphrasing lattice to improving MT by gaining more coverage.

2.2 Need for Paraphrasing

Current TM systems work on the surface level with no 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 edit-distance as implemented in OmegaT[1]. 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 beauty of edit-distance which has been trusted by translators, translation service providers and TM developers over the years.

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

2.3 PPDB:The Paraphrase Database

The PPDB 1.0 paraphrases database (Ganitkevitch et al., 2013) contains lexical, phrasal and syntactic paraphrases automatically extracted using a large collection of parallel corpora. This 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. We have used lexical and phrasal paraphrases of "L" size for our approach. The reason for choosing L size was to retain the quality of segments retrieved using paraphrasing and at the same time gain some coverage.

2.4 Classification of Paraphrases

We have classified paraphrases obtained from PPDB 1.0 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" ⇒"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 as well as 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 on the source and target sides differ, e.g. "a reasonable period of time to" ⇒ "a reasonable period to"

As we have already pointed out, 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. To handle this problem,

each class of paraphrases is processed in a different manner. In our classification, Type 1 are one-word paraphrases and Type 2 can be reduced to one-word paraphrases after considering the context when storing in the TM. For Type 1 and Type 2, we get the same accuracy as the trivial method in polynomial time complexity (see Section 3 for details). Paraphrases of Type 3 and Type 4 require additional attention because they still remain multiword paraphrases after reduction and greedy approximation is needed to implement them in polynomial time.

3 Our Approach

A general approach for TM matching and retrieval is as follows:

1. Read the Translation Memories available
2. Read the file that needs to be translated
3. Preprocess the input file, apply filter for different file formats and identify the segments
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
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.

For our implementation we used the open source TM tool OmegaT, which uses word-based edit-distance with cost 1 for insertion, deletion and substitution. We have employed OmegaT edit-distance as a baseline and adapted this to incorporate paraphrasing so that at a later stage we can add this feature in OmegaT without compromising the confidence users have in OmegaT fuzzy matches.

Our approach can be briefly described as the following steps:

1. Read the Translation Memories available
2. Collect all the paraphrases from the paraphrase database and classify them according to the classes presented in Section 2.4
3. Store all the paraphrases for each segment in the TM in their reduced forms according to the process presented in Section 3.1
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 3.2 and search for the most similar segment based on approach described in Section 3.3 and retrieve the most similar segment if above a predefined threshold

3.1 Storing Paraphrases

The paraphrases are stored in the TM in their reduced forms as after capturing paraphrases for a particular segment we have already considered the context and there is no need for it to be considered again while calculating edit-distance. We store only the longest uncommon substring instead of the whole paraphrase. This 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".

3.2 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

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

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. Our filtering steps for getting potential candidates for paraphrasing are as follows:

- We first filter out the segments based on length because if segments differ considerably in length, the edit-distance will also differ. In our case, the threshold for length was 49%. So, the TM segments which are shorter than 49% of the input are filtered.
- 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. In our case, the threshold was 49%.
- Next, after filtering the candidates with the above two steps we sort the remaining segments in decreasing order of similarity and pick the top 100 segments.
- 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% will be discarded[2].

3.3 Matching and Retrieval

For matching, similarity is calculated with the potential segments for paraphrasing extracted as per Section 3.2. Type 1 and Type 2 paraphrases after reduction (as per Section 3.1) are single-word paraphrases and Type 3 and Type 4 paraphrases have multiple words. For Type 1 and Type 2 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.

For paraphrases of Types 3 and 4 the algorithm takes the decision locally at the point where all paraphrases finish. The basic edit-distance calculation procedure is given in Algorithm 1. The algorithm elaborating our decision-making process is given in Algorithm 2. 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 22* executes when Type 3 and Type 4 paraphrases are not available (e.g. edit-distance calculation of the second token "period"). *Lines 24 to 57* account for the case when Type 3 and Type 4 paraphrases are available. *Line 28* 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 edit-distance of "laid down in" in Table 2). *Lines 33 to 46* 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 38 to 46* account for the selection of the minimum edit-distance paraphrase or source phrase. At *line 38*, 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 48, 52 and 56* account for updating the value of j to reflect the current position for further calculation of edit-

[2]these thresholds were determined empirically

Algorithm 1 Basic Edit-Distance Procedure

1: **procedure** EDIT-DISTANCE($Input$,TMS)
2: $M \leftarrow$ length of TMS ▷ Initialise M with length of TM segment
3: $N \leftarrow$ length of $Input$ ▷ 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: $D[i, j] \leftarrow minimum(D[i-1, j] + insertionCost, D[i, j-1] + deletionCost, D[i-1, j-1] + substitutionCost)$ ▷ store minimum of insertion, substitution and deletion
15: Return $D[N, M]$ ▷ Return minimum edit-distance
16: **end procedure**

	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	**3**	2	1	3	3	**3**	0

Table 2: Edit-Distance Calculation using Algorithm 2

distance (e.g. $j = 5$ after selecting "referred to") and *lines 50, 54 and 57* 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".

3.4 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.

4 Experiments and Results

For our experiments we have used English-French pairs of the 2013 release of the DGT-TM corpus (Steinberger et al., 2012). The corpus was se-

Algorithm 2 Edit-Distance with paraphrasing procedure

procedure EDIT-DISTANCEPP(Input,TMS)
  $M \leftarrow length(TMS)$ ▷ number of tokens in TM segment
  $N \leftarrow length(Input)$ ▷ number of tokens in Input segment
  $D[i, 0] \leftarrow i$ for $0 \leq i \leq N$ ▷ initialise two dimensional matrix D
  $D[0, j] \leftarrow j$ for $0 \leq j \leq (M + p')$ where p' accounts for increase in TM segment length because of paraphrasing
  $decisionPoint \leftarrow 0$, $j \leftarrow 1$
  $scost \leftarrow 1$, $dcost \leftarrow 1$, $icost \leftarrow 1$ ▷ initialisation of substitution, deletion and insertion cost
  while $j \leq M$ **do**
    $t \leftarrow TMS_j$ ▷ getting current TM token to process, e.g. 3^{rd} token "laid"
    if t has no paraphrases of type 3 and type 4 **or** $decisionPoint \geq N$ **then**
      $decisionPoint \leftarrow decisionPoint + 1$, $j \leftarrow j + 1$
      for $i \leftarrow 1...N$ **do**
        $InputToken \leftarrow Input_i$
        if $InputToken = t$ **then**
          $scost \leftarrow 0$
        else
          $scost \leftarrow 1$
        if $scost = 1$ **then**
          $OneWordPP \leftarrow getOneWordPP(t)$ ▷ get one word paraphrases associated with TM token t
          if $InputToken \in OneWordPP$ **then** ▷ applying type 1 and type 2 paraphrasing
            $scost \leftarrow 0$
        $D[i, decisionPoint] \leftarrow minimum(D[i, decisionPoint - 1] + dcost, D[i - 1, decisionPoint] + icost, D[i - 1, decisionPoint - 1] + scost)$
    else
      $tp \leftarrow$ get paraphrases stored at t ▷ e.g. tp for Token "laid" in Table 1
      $ls \leftarrow$ get corresponding source lengths stored at t ▷ e.g. ls for Token "laid" in Table 1
      $lsmax \leftarrow$ length of longest source phrase
      $DS[0, l - 1] \leftarrow D[0, decisionPoint + l]$ for $1 \leq l \leq lsmax$ ▷ initialise two dimensional matrix DS to calculate edit-distance of longest source phrase
      $DS \leftarrow$ calculate edit-distance of longest source phrase with $Input$ using D ▷ uses D for first word, consider Type 1 and Type 2 paraphrases
      $P \leftarrow$ number of paraphrases of type 3 and type 4 ▷ E.g. 2 for "laid"
      $index \leftarrow 0$, $paraphraselen \leftarrow 0$, $isppwin \leftarrow false$, $curDistance \leftarrow \infty$
      $prevDistance \leftarrow D[decisionPoint, decisionPoint]$
      $DTP[k, 0, l - 1] \leftarrow D[0, decisionPoint + l]$ 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
      for $k \leftarrow 0...P - 1$ **do**
        $dps[k] \leftarrow decisionPoint + ls[k]$
        $ltp \leftarrow length(tp[k])$ ▷ get paraphrase length e.g. 2 for "referred to"
        $dpt[k] \leftarrow decisionPoint + ltp$
        $DTP[k] \leftarrow$ calculate edit-distance of $tp[k]$ with $Input$ using D ▷ uses D for first word of $tp[k]$
        if $DTP[k, ltp - 1, dpt[k]] < DS[ls[k] - 1, dps[k]]$ and $DTP[k, ltp - 1, dpt[k]] < curDistance$ **then**
          $ppwin \leftarrow true$
          $curDistance \leftarrow DTP[k, ltp - 1, dpt[k]]$
          $index \leftarrow k$
          $paraphraselen \leftarrow ltp$
        else if $DS[ls[k] - 1, dps[k]] < curDistance$ **then**
          $ppwin \leftarrow false$
          $curDistance \leftarrow DS[ls[k] - 1, dps[k]]$
          $index \leftarrow k$
      if $ppwin = true$ **then** ▷ $true$ if paraphrase is better
        $j \leftarrow j + ls[index]$
        $decisionPoint \leftarrow decisionPoint + paraphraselen$
        update D using $DTP[index]$
      else if $curDistance = prevDistance$ **then** ▷ $true$ if source phrase is better and exactly matching
        $j \leftarrow j + ls[index]$
        $decisionPoint \leftarrow decisionPoint + ls[index]$
        update D using DS
      else
        $j \leftarrow j + 1$, $decisionPoint \leftarrow decisionPoint + 1$
        update D using DS
  Return $D[N, decisionPoint]$
end procedure

lected in such a way that it was not used to produce PPDB. For this reason, its language may be slightly different from the one used to produce PPDB, which may be a reason for the relatively modest results obtained in this paper. 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 et al., 2006). 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

TH	**100**	**95**	**90**	**85**	**80**
EDR	6352	7062	8369	9829	10730
PPR	6444	7172	8476	9938	10853
Imp	1.45	1.56	1.28	1.11	1.15
RC	13	20	43	68	88
BPP	**74.31**	**73.16**	**65.01**	63.29	60.84
BED	65.89	70.29	60.70	63.29	61.31

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	6421	7142	8450	9915	10820
Imp	1.09	1.13	0.97	0.87	0.84
RC	8	13	27	45	55
BPP	**73.18**	**73.98**	**63.08**	64.37	**63.37**
BED	60.86	71.43	61.96	65.10	63.28

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	8294	8802	9735	10597	11462
IMP	1.41	1.46	1.37	1.35	1.36
RC	21	30	43	73	108
BPP	**68.61**	**78.04**	**75.40**	**69.06**	**63.93**
BED	59.89	67.88	66.32	63.57	61.92

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

TH	**100**	**95**	**90**	**85**	**80**
EDR	8179	8675	9603	10456	11308
PPR	8277	8777	9706	10568	11422
IMP	1.2	1.18	1.07	1.07	1.01
RC	19	24	30	49	73
BPP	**58.28**	**67.95**	**71.03**	**68.03**	**61.02**
BED	52.00	54.81	60.09	62.13	57.42

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

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, and 7 where each table shows the similarity threshold for TM (TH), the total number of segments retrieved using the baseline approach (EDR), the total number of segments retrieved using our approach (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) and the BLEU score on target side over corresponding translations retrieved (irrespective of similarity score) by baseline approach for these segments (BED).

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, which is usually the case for real 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% .

5 Conclusion and Future work

We have presented an efficient approach to incorporating paraphrasing in TM. The approach is simple and fast enough to implement in practice. We have also shown that incorporating paraphrasing significantly improves TM matching and retrieval. Apart from TM, the approach can also be used for other natural language processing tasks (e.g. to incorporate paraphrasing in sentence semantic similarity measures exploiting edit-distance).

In future, we would like to consider the syntactic structure of the paraphrases when performing matching and retrieval, and also to take into account the context in which the paraphrases are used in order to have better accuracy. Alternative ways to implement using Finite State Transducers (FST) can also be considered and compared.

Acknowledgement

The research leading to these results has received funding from the People Programme (Marie Curie Actions) of the European Union's Seventh Framework Programme FP7/2007-2013/ under REA grant agreement no. 317471.

References

Arthern, Peter J. 1978. Machine Translation and Computerized Terminology Systems, A Translator's viewpoint. In *Translating and the Computer: Proceedings of a Seminar*, pages 77–108.

Du, Jinhua, Jie Jiang, and Andy Way. 2010. Facilitating Translation Using Source Language Paraphrase Lattices. In *Proceeding of EMNLP*, pages 420–429.

Ganitkevitch, Juri, Van Durme Benjamin, and Chris Callison-Burch. 2013. Ppdb: The paraphrase database. In *Proceedings of NAACL-HLT*, pages 758–764, Atlanta, Georgia. Association for Computational Linguistics.

Koehn, Philipp and Jean Senellart. 2010. Convergence of translation memory and statistical machine translation. In *Proceedings of AMTA Workshop on MT Research and the Translation Industry*, pages 21–31.

Lagoudaki, Elina. 2006. Translation Memories Survey 2006: Users' perceptions around TM use. In *Proceedings of Translating and the Computer 28*, pages 1–29, London. Aslib.

Mitkov, Ruslan. 2008. Improving Third Generation Translation Memory systems through identification of rhetorical predicates. In *Proceedings of LangTech2008*.

Onishi, Takashi, Masao Utiyama, and Eiichiro Sumita. 2010. Paraphrase Lattice for Statistical Machine Translation. In *Proceeding of the ACL*, pages 1–5.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the ACL*, pages 311–318.

Pekar, Viktor and Ruslan Mitkov. 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, Slav, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the COLING/ACL*, pages 433–440.

Steinberger, Ralf, Andreas Eisele, Szymon Klocek, Spyridon Pilos, and Patrick Schlüter. 2012. DGT-TM: A freely available Translation Memory in 22 languages. *LREC*, pages 454–459.

Utiyama, Masao, Graham Neubig, Takashi Onishi, and Eiichiro Sumita. 2011. Searching Translation Memories for Paraphrases. In *Machine Translation Summit XIII*, pages 325–331.

Zhechev, Ventsislav and Josef Van Genabith. 2010. Seeding statistical machine translation with translation memory output through tree-based structural alignment. In *Proceedings of ACL*, pages 43–51.