Improving Translation Memory Matching through Clause Splitting

Katerina Timonera

Research Group in Computational Linguistics

University of Wolverhampton

krtimonera@gmail.com

Ruslan Mitkov

Research Group in Computational Linguistics

University of Wolverhampton

r.mitkov@wlv.ac.uk

Abstract

We propose the integration of clause splitting as a pre-processing step for match retrieval in Translation Memory (TM) systems to increase the number of relevant sub-segment matches. Through a series of experiments, we investigate the impact of clause splitting in instances where the input does not match an entire segment in the TM, but only a clause from a segment. Our results show that there is a statistically significant increase in the number of retrieved matches when both the input segments and the segments in the TM are first processed with a clause splitter.

1 Rationale

Translation memory tools have had a great impact on the translation industry as they provide considerable assistance to translators. They allow translators to easily re-use previous translations, providing them with valuable productivity gains in an industry where there is a great demand for quality translation delivered in the shortest possible time. However, existing tools have some shortcomings. The majority of existing tools rely on Levenshtein distance, and seek to identify matches only at the sentence level. Semantically similar segments are therefore difficult to retrieve if the string similarity is not high enough, as are sub-segment matches because if only part of a sentence matches the input, even if this part is an entire clause, it is unlikely that this sentence would be retrieved (Pekar and Mitkov, 2007). As a result, TMs are especially useful only for highly repetitive text types such as updated versions of technical manuals.

In this study, we aim to address the problem of retrieving sub-segment matches by performing clause splitting on the source segment as a pre-processing step for TM match retrieval. While matches for entire sentences or almost entire sentences are the most useful type of matches, it is also less likely for such matches to be found in most text types, and even less so for complex sentences. Retrieving clauses is desirable because there is a higher chance for a match to be found for a clause than for a complex sentence, and at the same time, clauses are similar to sentences in that they both contain a subject and a verb, hence a "complete thought", therefore clause matches are more likely to be in context and to actually be used by the translator than phrase matches, for example.

We perform experiments comparing the match retrieval performance of TM tools when they are used as is, and when the input file and the TM segments are first processed with a clause splitter before being fed into the TM tool. The paper is organised as follows: In section 2 we discuss related work on TM matching. In section 3 we discuss how clause splitting can be beneficial to TM matching and how clause splitting was implemented for this study. We then describe our experiments in section 4, and discuss the results in section 5. Finally, we present our conclusion and future work in section 6.

2 Related Work

Attempts to address the shortcomings of existing tools include the integration of language processing to break down a sentence into smaller segments. The so-called 'second generation' TM system Similis (Planas, 2005) performs chunking to split sentences into syntagmas to allow sub-sentence matching.

However, Reinke (2013) observes that for certain language pairs like English-German, only rather short phrases like simple NPs are identified, and larger syntactic units cannot be retrieved This can be regarded as disadvantage as the processing of larger units would be desirable for the support of professional computer-assisted human translation (Kriele (2006) and Macken (2009), cited in Reinke, 2013).

MetaMorphoTM (Hodász and Pohl, 2005) also divides sentences into smaller chunks. Moreover, it uses a multi-level linguistic similarity technique (surface form, lemma, word class) to determine similarity between two source-language segments.

Other attempts involve deeper linguistic processing techniques. In Pekar and Mitkov (2007) we propose the 'third-generation translation memory' which introduces the concept of semantic matching. We employ syntactic and semantic analysis of segments stored in a TM to produce a generalised representation of segments which reduces equivalent lexical, syntactic and lexico-syntactic constructions into a single representation. Then, a retrieval mechanism operating on these generalised representations is used to search for useful previous translations in the TM.

This study is part of the third-generation translation memory project, of which the ultimate goal is to produce more intelligent TM systems using NLP techniques. To the best of our knowledge, clause splitting has not previously been investigated as a possible method for increasing the number of relevant retrieved matches.

3 Clause splitting for TM matching

Macklovitch and Russel (2000) note that when a sufficiently close match cannot be found for a new input sentence, current TM systems are unable to retrieve sentences that contain the same clauses or other major phrases.

For example, (a) below is a new input sentence composed of twenty five-character words. The TM contains the sentence (b), which shares an identical substring with sentence (a). However, as this substring only makes up only 25% of the sentence's total number of characters, it is unlikely that current TM tools would be able to retrieve it as a fuzzy match.

(a) w1 w2 w3 w4 w5, w6 . . . w20.

(b) w1 w2 w3 w4 w5, w21 . . . w35.

If clause splitting were employed, clauses would be treated as separate segments, thus increasing the likelihood that clauses which are subparts of larger units, could have a match score sufficiently high to be retrieved by the TM system.

In this paper, we compare the effect of performing clause splitting before retrieving matches in a TM. In each experiment, we identify the difference in matching performance when a TM tool is used as is, and when the input file and the translation memory are first run through a clause splitter. The clause splitter we use in this study is a modified version of the one described in Puscasu (2004). The original version employs both machine learning and linguistic rules to identify finite clauses for both English and Romanian, but in this version only the rule-based module is used. Puscasu (2004) developed a clause splitting method for both English and Romanian, and to maintain consistency between the two languages, her definition of a clause is the one prescribed by the Romanian Academy of Grammar, which is that a clause is

group of words containing a finite verb. Non/finite and verbless clauses are therefore not considered. The reported F-measure for identifying complete clauses in English is 81.39% (Marsic, 2011).

In this study, the clause splitter is used on both the segments in the input file and the translation memory database. After processing the input and the TM segments with the clause splitter, these were then imported into existing TM tools to examine how well these tools will perform if clause splitting is used in pre-processing.

4 Experiments

Experiments were performed to study the impact of clause splitting when used in pre-processing for the retrieval of segments in a TM. Our hypothesis is that when a clause splitter is used, the number of relevant retrieved matches will increase.

The effect of clause splitting is examined by comparing the number of matches retrieved when TM tools are used as is and when both the input segments and the segments in the translation memory are first processed with a clause splitter before being imported into the TM tools. The tools used are Wordfast Professional 3 and Trados Studio 2009, which are among the most widely used TM tools (Lagoudaki, 2006).

Segments used as the input were selected from the Edinburgh paraphrase corpus (Cohn, Callison-Burch and Lapata, 2008) (in Macken, 2009). We use a paraphrase corpus because we wish to investigate the effect of using a clause splitter in pre-processing to retrieve both segments that contain the entire input clause and segments that do not contain the exact input clause but may still be relevant as they contain a clause that shares a considerable degree of similarity with the input.

We examine the segments retrieved using both the default fuzzy match threshold (75% for Wordfast and 70% for SDL Trados) and the minimum threshold (40% for Wordfast and 30% for Trados). It is not normally recommended for translators to set a low fuzzy match threshold, as this might result in the retrieval of too many irrelevant segments if the translation memory is large. However, in this study, we argue it would be beneficial to examine matches retrieved with the minimum threshold as well. Given that translation memory match scores are mainly calculated using Levenshtein distance, if only one clause in a segment in the TM matches the input, there is a greater chance of the segment being retrieved with a lower threshold. We therefore wish to examine whether the employment of clause splitting will still result in a considerable improvement from using the Levenshtein distance-based matching algorithm in most TM tools if the match threshold setting is already optimised for the retrieval of sub-segment clauses.

It must also be noted that for this study, we are working with the source segments only. Therefore, in the TM files used, both the source and target segments are in English.

We conducted two main sets of experiments referred to as Set A and Set B which are outlined below. In Set A we selected sentences from the Edinburgh corpus that contained more than one clause. We use one clause, or part of it, as the input segment, and we store the entire sentence in the TM. In the experiments where no clause splitting is done, the sentence is stored as is. In the experiments with clause splitting, the original input segments are split into clauses (if there are more than one) and the segments in the TM are the component clauses of the original sentence. For the experiments done without clause splitting, there are 150 input segments and for each one, we test whether the longer corresponding segment in the TM can be retrieved. For the experiments where clause splitting is used, the 150 input segments are split into 180 segments as some of these segments contain more than one clause. We then test whether the corresponding clause from the original longer sentence can be retrieved. An example is presented in Table 1.

The underlined segments are the corresponding segments that should be retrieved.

In Set B there are also 150 input segments for the experiments where no clause splitting is used, and the corresponding segment in the TM is a longer sentence containing a paraphrase of the input segment. For the experiments with clause splitting, there are 185 input segments (as in set A, some of the original 150 have more than one clause) and in the TM, the component clauses of the original longer segment are stored, and we test whether the clause that is a paraphrase of the input can be retrieved. Below is an example.

Without clause splitting	
Input	**Segment in TM**
the ministry of defense once indicated that about 20,000 soldiers were missing in the korean war	the ministry of defense once indicated that about 20,000 soldiers were missing in the korean war and that the ministry of defense believes there may still be some survivors .
With clause splitting	
Input	**Segments in TM**
- the ministry of defense once indicated - that about 20,000 soldiers were missing in the korean war	- the ministry of defense once indicated - that about 20,000 soldiers were missing in the korean war - and that the ministry of defense believes - there may still be some survivors .

Table 1. Set A Example

Without clause splitting	
Input	**Segment in TM**
a member of the chart-topping collective so solid crew dumped a loaded pistol in an alleyway	a member of the rap group so solid crew threw away a loaded gun during a police chase, southwark crown court was told yesterday .
With clause splitting	
Input	**Segments in TM**
a member of the chart-topping collective so solid crew dumped a loaded pistol in an alleyway	- a member of the rap group so solid crew threw away a loaded gun during a police chase , - southwark crown court was told yesterday .

Table 2. Set B Example

5 Results

WORDFAST	**W/o clause splitting**	**W/ clause splitting**
% Retrieved (Default threshold)	23.33%	90.00%
% Retrieved (Minimum threshold)	38.00%	92.22%
TRADOS	**W/o clause splitting**	**W/ clause splitting**
% Retrieved (Default threshold)	14.00%	88.89%
% Retrieved (Minimum threshold)	14.00%	96.67%

Table 3. Percentage of correctly retrieved segments in Set A

Table 3 shows the results of the experiments in set A. It is clear that clause splitting considerably increases the number of matches in instances where the input segment can be found in a longer segment stored in the TM. When the corresponding segments that could not be retrieved even with the minimum threshold were analysed, we found that in set A, all instances were due to errors in clause splitting, more specifically the fact that the clause splitter failed to split a sentence containing more than one clause.

Table 4 summarises the percentage of correctly retrieved segments in set B. In this set, it was observed that although the percentage of retrieved matches is generally lower than the percentages in set A, there is

still a noticeable increase in the percentage of matches retrieved.

WORDFAST	W/o clause splitting	W/ clause splitting
% Retrieved (Default threshold)	2.67%	17.84%
% Retrieved (Minimum threshold)	10.00%	41.08%
TRADOS	**W/o clause splitting**	**W/ clause splitting**
% Retrieved (Default threshold)	3.33%	25.95%
% Retrieved (Minimum threshold)	36.00%	70.67%

Table 4. Percentage of correctly retrieved segments in Set B

Upon examination of the segments that could not be retrieved even with the default threshold, we found that in both Wordfast and Trados, around 24% had clause splitting errors, such as when a segment is not split at all when it has more than one clause, or when the segment is incorrectly split. As for the rest of the unretrieved segments, we presume that they are so heavily paraphrased that even when clause splitting is performed correctly, the TM tools are still unable to retrieve them.

For each experiment, we conduct a paired t-test using the match scores produced by the TM tools when retrieving each segment (Table 5). When there are no matches or the correct match is not retrieved, the match score is 0. In instances where the original input segment has more than one clause and is thus split by the clause splitter, we take the average match score of the clauses and take this as one case in order to make the results comparable. In all experiments, the difference is significant at the 0.0001 level when computed with SPSS. We can therefore reject the null hypothesis and conclude that there is a statistically significant difference between the results.

SET A			
	Mean		
WORDFAST	**Without clause splitting**	**With clause splitting**	**p-value**
Default threshold	19.23	85.30888891	0.000
Minimum threshold	29.28	86.48888891	0.000
TRADOS	**Without clause splitting**	**With clause splitting**	**p-value**
Default threshold	11.78	83.87777781	0.000
Minimum threshold	11.78	88.07111113	0.000
SET B			
	Mean		
WORDFAST	**Without clause splitting**	**With clause splitting**	**p-value**
Default threshold	2.23	17.21111111	0.000
Minimum threshold	6.49	30.62111112	0.000
TRADOS	**Without clause splitting**	**With clause splitting**	**p-value**
Default threshold	2.71	23.083	0.000
Minimum threshold	16.83	46.36777779	0.000

Table 5. Paired t-test on all experiments

6 Conclusion

Our results show that introducing clause splitting as a pre-processing step in TM match retrieval can significantly increase matching

performance in instances where the TM contains segments of which one of the clauses corresponds to the input segment or is a paraphrase of the input segment.

It is worth mentioning that the data used in these experiments are not data imported from the translation memories of practicing translators as they are not easily available. We nevertheless believe that the results of this study provide significant support to the proof-of-concept of third-generation TM systems where NLP processing is expected to improve performance of operational TM systems.

In future work, we wish to incorporate alignment so that on the target side, what is retrieved is not the original target segment but the corresponding clause, as in its current state, our method would only be able to retrieve the original target segment, given that we perform clause splitting only on the source side. It would also be desirable to implement a working TM tool that incorporates clause splitting and examine to what extent these help a translator working on an actual translation project, as the final test of the usefulness of the methods employed is how they actually increase the productivity of translators in terms of time saved.

Acknowledgements

We would like to acknowledge the members of the Research Group in Computational Linguistics at University of Wolverhampton for their support, especially Dr Georgiana Marsic, Dr Constantin Orasan and Dr Sanja Štajner. Part of the research reported in this paper is supported by the People Programme (Marie Curie Actions) of the European Unions Framework Programme (FP7/2007-2013) under REA grant agreement no. 317471.

References

Cohn, T., Callison-Burch, C. and Lapata, M. (2008) Constructing Corpora for the Development and Evaluation of paraphrase systems. *Computational Linguistics*, *34*(4), 597-614.

Hodász, G. and Pohl, G. (2005) MetaMorpho TM: A Linguistically Enriched Translation Memory. In *Proceedings of the International Conference on n Recent Advances in Natural Language Processing (RANLP-05).* Borovets, Bulgaria.

Kriele, C. (2006) Vergleich der Beiden Translation-Memory-Systeme TRADOS und SIMILIS. Diploma thesis. Saarbrücken: Saarland University [unpublished].

Lagoudaki, E. (2006) Translation Memories Survey 2006: Users' Perceptions around TM Use. In *proceedings of the ASLIB International Conference Translating & the Computer* (Vol. 28, No. 1, pp. 1-29).

Macken, L. (2009) In Search of the Recurrent Units of Translation. In *Evaluation of Translation Technology*, ed. by Daelemans, Walter and Véronique Hoste, 195-212. Brussels: Academic and Scientific Publishers.

Macklovitch, E. and Russell, G. (2000) What's been Forgotten in Translation Memory. In *Envisioning machine translation in the information future* (pp. 137-146). Springer Berlin Heidelberg.

Marsic, G. (2011) *Temporal Processing of News: Annotation of Temporal Expressions, Verbal Events and Temporal Relations* [PhD thesis, University of Wolverhampton].

Pekar, V. and 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*, 29-30 September 2006, Bern.

Planas, E. (2005) SIMILIS - Second generation TM software. In *Proceedings of the 27th International Conference on Translating and the Computer (TC27)*. London, UK.

Puscasu, G. (2004) A Multilingual Method for Clause Splitting. In *Proceedings of the 7th Annual Colloquium for the UK Special Interest Group for Computational Linguistics*.

Reinke, U. (2013) State of the Art in Translation Memory Technology. *Translation: Computation, Corpora, Cognition*, *3*(1).

.