Translation Memory Retrieval Using Lucene

Kwang-hyok Kim[1], Myong-ho Cho[1], Chol-ho Ryang[1], Ju-song Im[1], Song-yong Cho[1], Yong-jun Han[2]

[1]Faculty of Information Science **Kim Il Sung** University Pyongyang, DPRK
{kh.kim0107,ch.ryang0415,sy.jo1020}@ryongnamsan.edu.kp
[2]Faculty of Foregin Language and Literature, **Kim Il Sung** University Pyongyang, DPRK
flit1@ryongnamsan.edu.kp

Abstract

Translation Memory (TM) system, a major component of computer-assisted translation (CAT), is widely used to improve human translators' productivity by making effective use of previously translated resource. We propose a method to achieve high-speed retrieval from a large translation memory by means of similarity evaluation based on vector model, and present the experimental result. Through our experiment using Lucene, an open source information retrieval search engine, we conclude that it is possible to achieve real-time retrieval speed of about tens of microseconds even for a large translation memory with 5 million segment pairs.

1 Introduction

Translation memory technique is a key functionality being widely used in the field of CAT. Translation Memories (TMs) are "structured archives of past translations" which store pairs of corresponding text segments in source and target languages known as "translation units" (Simard, 2020). The size of translation memories and the quality of their contents are major impact factors crucial to the effectivity of the translation memory system which uses them. Due to the importance of translation memories, there has been done lots of research work for building large TMs on world-wide scale, not just in individual countries (Steinberger et al. 2012).

What plays an important role for TM system is also the similarity evaluation between the input sentence to be translated and the source segment in the TM. The main task of TM system is to get a translation unit whose source segment is the most similar to the input sentence out of TM. There are two possible solutions in performing the task: One solution is to adopt an intelligent TM matching mechanism which is able to correctly calculate the similarity between the input sentence and the source segment in the translation memory. The other solution is to increase the size of TM by collecting translated resources as much as possible. No matter how intelligent the TM matching mechanism is, small-size TM cannot afford rich performance. Of course, the choice of TM matching method is important for improving the effectivity of TM system. But what is no less important than any TM matching method is to use a reasonable size TM. The larger the TM, the higher the possibility to get a translation unit whose source segment is very similar to the input sentence out of the TM. In general, the main value of a TM consists in the number of segments - its size. However, large TMs automatically lead to slow response times. A slow TM might actually slow down a translator, so that fast response time is an essential characteristic of any TM. Many research works have been reported to improve TM matching and retrieval, but the majority of those approaches were just evaluated on relatively small TMs. To our best knowledge, the largest TM tested so far in previous research works is the first five parts of the 2013 DGT-TM (which consisted of 305,324 segment pairs) used in (Weitz 2017) and the 2018 Volume 1 of the DGT-TM (which had 230,000 segment pairs) used in (Ranasinghe et al. 2020).

The main problem we are going to solve in this paper is to provide a TM retrieval mechanism to ensure real-time performance on very large TMs, e.g. with millions of segment pairs. We propose a TM retrieval method based on Vector Model (VM), which is widely used in information retrieval (IR), and implement our proposal using Lucene, an open source IR search engine. The rest of the paper is organized as follows: Section 2 briefly reviews previous research works related to TM matching and retrieval. Section 3 describes TM retrieval

Proceedings of Recent Advances in Natural Language Processing, pages 684–691
Sep 1–3, 2021.
https://doi.org/10.26615/978-954-452-072-4_078

method based on VM, and Section 4 presents experiment result. Finally, Section 5 discusses the result and draws a conclusion.

2 Previous Work on Translation Memory Matching and Retrieval

2.1 Research Work to Improve Translation Memory Matching

The mission of TM matching is to evaluate how similar the source segment in the TM is to the input sentence to be translated. Hence most of research work on TM matching focuses on how to calculate the similarity between the input sentence and the source segment in the TM.

(Planas and Furuse 2000) introduces edit distance based similarity vector whose coordinates refer to the levels of analysis of the segments. Their Multi-level Similar Segment Matching (MSSM) algorithm uses 3 different levels of data (surface words, lemmas, parts of speech (POS)) in a combined and uniform way.

There are studies for improving TM matching by segmenting source sentences. It is less likely for exact matches to be found in most text types, and even less so for complex sentences. MetaMorpho TM (Hodász and Pohl 2005) also divides sentences into smaller chunks. Moreover, it uses a multi-level linguistic features (surface form, lemma, and word class) to determine similarity between two source-language segments, especially for morphologically rich languages like Hungarian. The so-called 'second generation' TM system SIMILIS (Planas 2005) performs chunking to split sentences into syntagmas to allow sub-sentence matching. (Timonera and Mitkov 2015) suggests improving translation memory matching by performing clause splitting on the source segment as a pre-processing step for TM match retrieval, since clauses both contain a subject and a verb, hence a "complete thought", and therefore clause matches are more likely to be in context and to be actually used by the translator.

(Vanallemeersch and Vandeghinste, 2014) also proposes a method which performs matching at level of syntactic trees. The authors notice that tree matching method is "prohibitively slow".

(Pekar and Mitkov 2007) proposes the 'third-generation translation memory' which introduces the concept of semantic matching. They employ syntactic and semantic analysis of segments stored in a TM to produce a generalized representation of segments which reduces equivalent lexical, syntactic and lexicosyntactic constructions into a single representation. Then, a retrieval mechanism operating on these generalized representations is used to search for useful previous translations in the TM.

(Chatzitheodorou 2015) presents an approach to match sentences having different words but the same meaning. They use NooJ to create paraphrases of Support Verb Constructions (SVC) of all source translation units to expand the fuzzy matching capabilities when searching in the translation memory.

(Ranasinghe et al. 2020) introduces a TM matching and retrieval method based on Universal Sentence Encoder. They argue that their method is an effective and efficient solution to replace edit distance based algorithms.

2.2 Research Work to Improve Translation Memory Matching

The mission of translation unit retrieval is to filter translation units out of TM which are to be matched against the input sentence. In general, the time consumed for translation unit retrieval is linear to the size of TM. Levenshtein distance, which is widely being used and one of the simplest means for TM matching, can be computed with dynamic programming in O(mn) time, where m is the length of the input sentence, and n the length of the source segment of a translation unit in the translation memory. However, in case of a large TM with more than tens of millions of segment pairs, computing edit distance against the whole TM is too slow for real-time use. This is why the preliminary retrieval is necessary.

(Dandapat et al. 2012) uses an open-source IR engine SMART to retrieve a potential set of candidate sentences (likely to contain the closest match sentence) from the example base. Unigrams extracted from the sentences of the example-base are indexed using the language model and complete sentences are considered as retrievable units. They reported that finding a set of candidate sentences took only 0.3 seconds and 116 seconds, respectively, for 414 and 10,000 example input sentences given 20k and 250k sentence example base on a 3GHz Core 2 Duo machine with 4GB RAM. In order to find the closest matching sentences efficiently, (Dandapat et al. 2012) also proposes a heuristic-based grouping method which divides the example-base into bins based on

sentence length and considers only the segments which has comparable length to the length of the input sentence.

(Wäschle and Riezler 2015) uses MinHash signatures, an efficient way to estimate the similarity of two documents, to efficiently approximate the n-gram overlap of the input sentence and the source segment by representing each sentence as a set of n-grams in that n-gram overlap is a good predictor of TM match quality.

In order to reduce the search space size for Korean-Chinese TM retrieval, (Ryang 2018) builds a structured index using as features the sentence length and the sequence of Korean particles which is included in the sentence.

3 Vector Model-based Similarity Evaluation for Translation Memory Retrieval

3.1 Primary and Secondary Retrieval of Translation Memory

When retrieving from a large TM, it is common and reasonable to use the two-stage approach in which the TM system, firstly, filters candidates likely to be related to the input sentence for TM matching and then finds the most similar segments by fine-grained matching. The filtering is referred to the primary retrieval and the fine-grained finding is referred to the secondary retrieval. (Figure 1)

The primary retrieval is intended to filter translation units whose source segment is likely to be close matched with the input sentence. The secondary retrieval returns as reference translation the target segments of the translation units whose source segment is best matched with the input sentence. The main difference between the primary and secondary retrieval lies in the fact that the secondary retrieval uses a certain similarity threshold, μ, and the count of the secondary retrieval output should be much smaller than the primary one, because the secondary retrieval output is for human. The primary and secondary retrieval can be formulated respectively as follows:

$$TM(S_0, K_1) = \underset{\substack{tm \subset TM \\ |tm| = K_1}}{\operatorname{argmax}} \sum_{(S_i,T_i) \in tm} FMS_1(S_0, S_i)$$

$$TM_\mu(S_0, K_2) = \underset{\substack{tm \subset TM(S_0,K_1) \\ |tm| \le K_2}}{\operatorname{argmax}} \sum_{\substack{(S_i,T_i) \in tm \\ FMS_2(S_0,S_i) \ge \mu}} FMS_2(S_0, S_i)$$

where

$FMS_1(S_0, S_i)$: similarity score of the input sentence S_0 and the source segment S_i, used in the primary retrieval

$FMS_2(S_0, S_i)$: similarity score of the input sentence S_0 and the source segment S_i, used in the secondary retrieval

$TM = \{(S_i, T_i) | i = \overline{1, N}\}$: Translation Memory

(S_i, T_i): Translation Unit,
S_i: Source Segment, T_i:Target Segment

N: the number of translation units in the translation memory

K_1, K_2: the limit count of the primary/secondary retrieval output

One of the essential requirements which the similarity measure should meet for the primary retrieval of TM is to allow partial match. A useful solution to this requirement is to use vector model

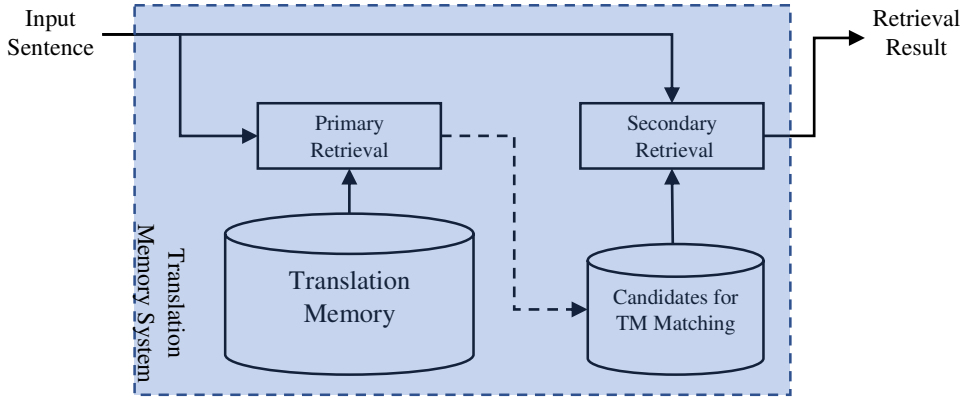

Figure 1: Two-stage Translation Memory Retrieval

by representing as vectors the input sentence and the source segments in the translation memory. We adopt vector model based similarity evaluation for the primary retrieval of TM.

3.2 Primary and Secondary Retrieval of Translation Memory

For the vector representation of the input sentence and the source segments in the TM, we use word-sentence relation matrix which is widely used in IR. Let W be the set of words occurring in the source segments.

$W = \{w_i | i = \overline{1,T}\}$, T: The total number of words occurring in the source segments

Let V be the word-sentence relation matrix. Then V is a $N \times T$ dimensional matrix:

$$V = \begin{pmatrix} v_{11}\ v_{12} \dots v_{1T} \\ v_{21}\ v_{22} \dots v_{2T} \\ \dots \dots \dots \dots \dots \dots \\ v_{N1}\ v_{N2} \dots v_{NT} \end{pmatrix}$$

The i-th row of V is a vector representing the source segment, S_i:

$$V_{S_i} = (v_{i1}, v_{i2}, \dots, v_{iT})$$

v_{ij}: The weight indicating how important the word $w_j \in W$ is for the source segment S_i

Let U_{S_0} be the vector of the input sentence S_0:

$$U_{S_0} = (u_1, u_2, \dots, u_T)$$

u_j: The weight indicating how important the word $w_j \in W$ is for the input sentence S_0

Given two vectors, V_{S_i} and U_{S_0}, the similarity score of the input sentence S_0 and the source segment S_i used for the primary retrieval can be defined as follows:

$$FMS_1(S_0, S_i) = \frac{U_{S_0} \cdot V_{S_i}}{\|U_{S_0}\| \, \|V_{S_i}\|}$$

$U_{S_0} \cdot V_{S_i}$: Dot product of two vectors, U_{S_0} and V_{S_i}

$\|U_{S_0}\|$: Euclidean norm of the vector U_{S_0}

$\|V_{S_i}\|$: Euclidean norm of the vector V_{S_i}

We suggest using TF-IDF weight of the words, which is commonly used feature for IR. But there is one problem in using TF-IDF weight for TM retrieval.

In IR, the length of a query is very short than documents. However, in case of TM retrieval, the lengths of the input sentence and the source segment, two objects to be compared, don't make such contrastive difference as in the relationship between the query and document in IR. It can rather be assumed that the length of the input sentence is similar to the source segment in the TM. Even when the length of the input sentence is short than the source segment, as in IR, if a word occurs only once in the input sentence, it is not true that a source segment, in which that word occurs two or three times, is more similar to the input sentence than any other source segment in which that word occurs once. When a word occurs twice in the input sentence, it can be assumed that a source segment, in which that word occurs twice, is more similar than any other source segment in which that word occurs once. Based on this consideration, we define the elements of the vectors U_{S_0} and V_{S_i} as:

$$v_{ij} = \min\{tf(w_j, S_0), tf(w_j, S_i)\}$$

$$u_j = \begin{cases} idf(w_j), & w_j \in S_0 \\ 0\ , & w_j \notin S_0 \end{cases}$$

Consequently, the similarity score of the input sentence S_0 and the source segment $S_i$becomes:

$$FMS_1(S_0, S_i) = \\ = \frac{\sum_{w_j \in S_0} \min\{tf(w_j, S_0), tf(w_j, S_i)\} \times idf(w_j)}{\|U_{S_0}\| \, \|V_{S_i}\|}$$

In the calculation of the above similarity score function, the elimination of the term $\|U_{S_0}\|$ from the denominator doesn't affect the final result. So the practical similarity score function can be written as:

$$FMS_1(S_0, S_i) = \\ = \frac{\sum_{w_j \in S_0} \min\{tf(w_j, S_0), tf(w_j, S_i)\} \times idf(w_j)}{\|V_{S_i}\|}$$

3.3 Semantic Similarity for the Primary Retrieval of TM

There are many previous research works taking into account semantic similarity for TM matching. For example, given two sentences, "What is the actual aim of this practice?" and "What is the real goal of this mission?", it is possible to judge that these two sentences are very similar, based on the linguistic analysis that the words "actual" and "goal" are similar to the words "real" and "aim," respectively. When implementing two-stage TM retrieval which consists of primary and secondary retrieval for a large TM, it is very important to ensure that the output of the primary retrieval might contain the segments likely to be semantically similar to the input sentence for any

semantic similarity measure to be applied at the secondary retrieval stage. We are going to use linguistic knowledge like synonym for accommodating semantic similarity evaluation in the primary retrieval of TM.

Our solution to evaluate semantic similarity taking into account the synonym knowledge in the primary retrieval of TM, is to change the input sentence S_0 into a pseudo sentence S_0' which includes all the words of S_0 and also their synonym words, and then calculate the similarity score of the pseudo sentence S_0' and the source segments of TM. The pseudo expansion of the input sentence and the similarity score calculation is trivial since the vector representation is based on TF-IDF weights. The similarity score of the pseudo sentence S_0' and the source segment S_i is:

$$FMS_1(S_0',\ S_i) = \frac{\sum_{w_j \in S_0'} \min(tf(w_j, S_0'), tf(w_j, S_i)) \times idf(w_j)}{\|V_{S_i}\|}$$

where

$$tf(w_j, S_0') = \begin{cases} tf(w_j, S_0), & w_j \in S_0 \\ \alpha, & w_j \notin S_0 \wedge \exists k, w_k \in S_0 \wedge w_j \in SYN(w_k) \\ 0, & w_j \notin S_0 \wedge w_j \notin SYN(S_0) \end{cases}$$

$SYN(S_0) = \cup_{w_i \in S_0} SYN(w_i)$, $SYN(w_i)$: The set of synonym words of w_i

In the above expression, α is a real number between 0 and 1, which is introduced as a weight of the synonym word added into the pseudo input sentence S_0'.

The knowledge database for synonym are not always available for every language, and even if available, they are qualitatively and quantitatively different from each other. For English, WordNet developed by Princeton University is a useful knowledge database for finding synonym.

As a matter of fact, it is not quite easy to find correctly the synonym of any word in the input sentence. To speed up the primary retrieval on a large TM while avoiding complex linguistic analysis, we establish the following principle for building synonym dictionary which will be used in the similarity evaluation for TM retrieval.

First, for any word w_i which has only one part-of-speech (POS), its all synonym words will be included in the synonym dictionary $SYN(W)$.

Second, when the word w_i has several POSes, only if w_i doesn't have verb POS, its synonym words will be included in the synonym dictionary $SYN(W)$.

Third, if the word w_i has both general synonym and special synonym, only the general synonym words with more high frequency will be included in the synonym dictionary $SYN(W)$.

Our principles are based on the linguistic consideration that the synonym of any word can be discussed only when its POS is determined, that there exist two categories of synonym, absolute synonym and relative synonym, and that there are also general synonym and special synonym in terms of use frequency.

We don't use synonym of multi-POS words with verb POS, because a verb is the core of the statement unit which can determine the meaning of a sentence from a linguistic point of view and linguistic analysis like POS tagging is not applied in the primary retrieval of TM.

According to our analysis on WordNet 3.0[1], it has a total of 117,659 senses with 147,306 words related to each other. Among those words, there are 49,754 words which does not have any synonym at all. Using above-mentioned principles for synonym selection, we selected 36,185 senses with 90,258 words related to each other to build an English synonym dictionary for TM retrieval.

4 Experimental Result

We use Lucene, an open source IR engine in Java, to implement the TM retrieval system using the vector model based similarity evaluation we proposed. As the data structure of a translation unit of TM, the *Document* class of Lucene is used which has two fields corresponding to the source and target segment of TM, respectively. The version number of Lucene used is 8.5.1. Levenshtein Distance based similarity score is applied for the secondary retrieval of TM. The TM used for the evaluation of the proposed TM retrieval system is an English-to-Korean TM which is made of about 5,000,000 English segments and their automatic Korean translation by English-to-Korean machine translator "Ryongnamsan" 2.0. In the experiment, we use parameter settings for the primary search of TM such that $K_1 = 100$, and $\alpha = 0.5$. All measurement was carried out on a desktop PC with

[1] (http://wordnetcode.princeton.edu/3.0/WNprolog-3.0.tar.gz)

Intel® Core™ i3-3240 CPU @ 3.40GHz and 2GB of RAM. The operating system installed is Windows 7, 64bit.

4.1 Evaluation Method of Retrieval Performance

First of all, the retrieval performance on the English-to-Korean TM using Lucene can be evaluated with the retrieval time on varying size of TM. We randomly selected 1,000 sentences which are not included in the English-to-Korean TM, and then measured the total time consumed for retrieving all those sentences on different size of TM. The time consumed for retrieving was measured 5 times, and the fastest, slowest and averaged time were all recorded. Next, the relevance of retrieval result was automatically tested. Finally, we compare the retrieving performance of our proposal with the retrieving performance when using MongoDB's full text search API.

4.2 Result

– Relation between the size of TM and the retrieving time

.

Figure 2 shows the relation between the size of TM and the retrieving time.

As the size of TM increases, so does the retrieving time on the TM.

– Relation between the length of the input sentence and the retrieving time

We also investigate the influence of the length of the input sentence on the retrieving time on TM.

Size of TM (×10,000)	Retrieving time (milliseconds) (Number of Measurement)				
	1	2	3	4	5
50	14,289	6,284	6,349	6,379	6,284
100	25,506	8,703	8,659	8,551	8,767
150	33,215	10,581	10,481	10,578	10,470
200	42,523	12,526	12,433	12,306	12,447
250	47,191	14,914	14,899	14,851	14,743
300	59,027	17,375	17,379	17,473	17,459
350	72,304	18,998	19,077	19,063	18,940
400	78,106	21,199	21,408	21,231	21,214
450	85,019	23,479	23,446	23,211	23,290
500	96,876	27,112	25,414	25,448	25,411

Table 1: Retrieving time according to the size of TM

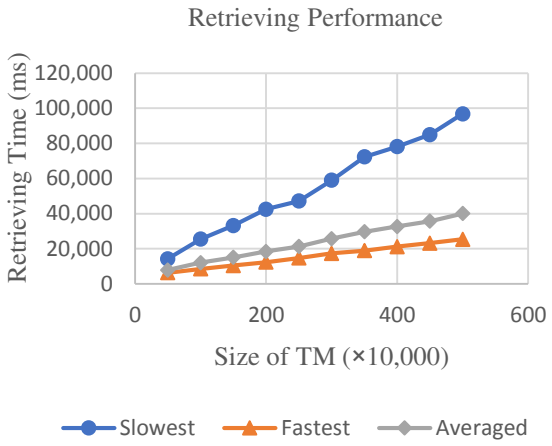

Figure 2: Retrieving time according to the size of TM

For 1,000 input sentences being tested, the retrieving time for each sentence on the largest TM with 5,000,000 segments was measured and averaged according to the length of those sentences. Figure 3 and Figure 4 show the sentence length-frequency distribution on the test sentences and the average retrieving time according to the sentence length, respectively.

The result shows that the longer the input sentence, the longer its retrieving time of TM.

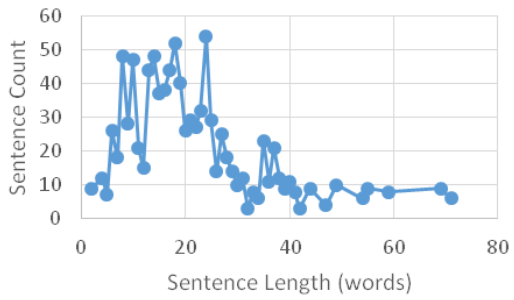

Figure 3: Sentence frequency according to its length

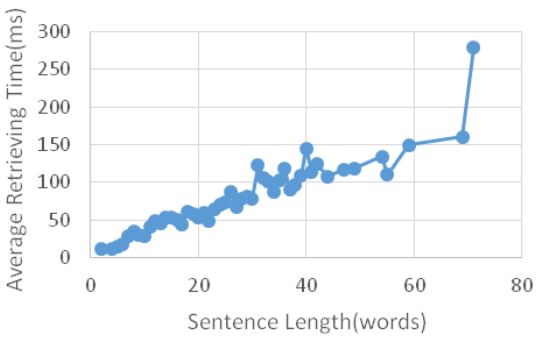

Figure 4: Average retrieving time according to the sentence length

– Relevance of the primary retrieval result of TM

For evaluating the relevance of the primary retrieval result of TM, we checked the ranking result of the primary retrieval when retrieving 1,000 English sentences randomly selected from the largest TM of 5,000,000 segments. According to an automatic checking of the ranking result, the translation unit whose source segment is the input sentence ranked at the first place all the time. This implies that the proposed primary retrieval of TM is relevant for exact match of TM.

The relevance of the primary retrieval result for the sentences which is not included in the TM is impossible to automatically evaluate, and is also related to the secondary retrieval of TM. We did a small manual test but the result was not fully reliable, so we didn't present the result here.

– Comparison with the retrieving performance of TM using MongoDB

MongoDB, a NoSQL database management system, supports textual data indexing and searching which allows partial matching. For comparison with our proposal, we implemented a TM retrieval module using the full text search API of MongoDB, and evaluated its performance on a desktop PC with Intel® Core™ i7-7700 CPU @ 3.6 GHz and 16 GB of RAM. The version of MongoDB used is 4.4. It took about 18 minutes to insert into the MongoDB collection the English-to-Korean TM of 5 million segment pairs. It also took about 5 minutes to index the source language field and about 1 hour and 38 minutes to retrieve a set of candidate sentences for the same 1,000 English sentences as in the previous experiment. The size of the set of candidate sentences was limited to 10. By automatically checking the relevance of the retrieval result, the translation unit whose source segment is the input sentence ranked at the first place all the time, too. Obviously, the retrieving speed when using Lucene is incomparably superior to when using MongoDB.

5 Conclusion and Future Work

Through a series of experiments on the primary retrieval of English-to-Korean TM using vector model based similarity evaluation, we conclude that:

- The time and space complexity of indexing the TM increases linear to the size of the TM. The indexing time consumed for a large-scale English-to-Korean TM with about 5,000,000 segments is about 5 minutes, and the indexed data size is 1.84 GB with an increase of about 29 % compared to the text file size of the TM.

- The time complexity of the primary retrieval of TM increases linear to the size of the TM and the length of the input length. The fact that the retrieving time of TM is in linear proportion to the size of the TM and the length of the input sentence fully accords with Lucene's inverted indexing principle and the ranking process of our vector model based similarity evaluation.

- When there is a translation unit whose source segment is the same as the input sentence, the translation unit ranks at the first place in the primary retrieval result of TM. The automatic checking result of the source segments included in the TM shows that Lucene is an effective means for exact match, as well as fuzzy matching.

The effect of the vector model based similarity evaluation for the primary retrieval of TM wholly depends on the correctness of the morphological analysis and the richness of the synonym knowledge. Since the difficulty of the morphological analysis and the availability of the synonym knowledge like WordNet is all different for each language, we plan to do more research work on these aspects. Furthermore, we also plan to evaluate more comprehensively the relevance of the primary retrieval of TM.

References

Emmanuel Planas and Osamu Furuse. (2000) *Multi-level Similar Segment Matching Algorithm for Translation Memories and Example-Based Machine Translation*. In: COLING 2000, proceedings of the 18th international conference on computational linguistics, Saarbrücken, Germany, pages 621–627

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

Gábor Hodász and Gábor Pohl. (2005) MetaMorpho *TM: A Linguistically Enriched Translation Memory*.

In Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP-05). Borovets, Bulgaria.

Katerina Timonera and Ruslan Mitkov. (2015) *Improving Translation Memory Matching through Clause Splitting*. In Proceedings of the Workshop on Natural Language Processing for Translation Memories (NLP4TM), pages 17–23, Hissar, Bulgaria

Katharina Wäschle and Stefan Riezler. (2015) *Integrating a Large, Monolingual Corpus as Translation Memory into Statistical Machine Translation*, In Proceedings of the 18th Annual Conference of the European Association for Machine Translation, pages 169-176

Konstantinos Chatzitheodorou. (2015) *Improving translation memory fuzzy matching by paraphrasing*, In Proceedings of the Workshop on Natural Language Processing for Translation Memories (NLP4TM), pages 24–30, Hissar, Bulgaria

Kum-Chol Ryang. (2018) *Improving fuzzy search on translation memory*, (In Korean) Technology Innovation, Volume 2018(5), page 39

Melanie Weitz. (2017) *Improving retrieval performance of translation memories using morphosyntactic analyses and generalized suffix arrays*, Machine Translation (2017) 31: 117-146

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

Sandipan Dandapat, Sara Morrissey, Andy Way, and Joseph van Genabith. (2012) *Combining EBMT, SMT, TM and IR Technologies for Quality and Scale*, In Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics, pages 48–58, Avignon, France

Simard Michel. 2020. Building and using parallel text for translation. In O'Hagan, Minako, editor, The Routledge Handbook of Translation and Technology, chapter 5, pages 78-90. Routledge.

Tharindu Ranasinghe, Constantin Orasan, and Ruslan Mitkov. (2020) Intelligent Translation Memory Matching and Retrieval with Sentence Encoders. ArXiv, abs/2004.12894.

Tom Vanallemeersch and Vincent Vandeghinste. (2014) *Improving fuzzy matching through syntactic knowledge*. In Translating and the Computer 36, volume 36, pages 90 – 99, London, UK.

Viktor Pekar and Ruslan Mitkov. (2007) *New Generation Translation Memory: ContentSensitive Matching*. In Proceedings of the 40th Anniversary Congress of the Swiss Association of Translators, Terminologists and Interpreters, 29-30 September 2006, Bern.