Part II

Applications - Translation Memories

Chapter 6

Introduction to Translation Memories

Translation Memories (TMs) are "structured archives of past translations" which store pairs of corresponding text segments[1] in source and target languages known as "translation units" [156]. TMs are used during the translation process in order to reuse previously translated segments. The original idea of TMs was proposed more than forty years ago when Arthern [157] noticed that the translators working for the European Commission were wasting valuable time by re-translating (parts of) texts that had already been translated before. He proposed the creation of a computerised storage of source and target texts which could easily improve the performance of translators and that could be part of a computer-based terminology system. Based on this idea, many commercial TM systems appeared on the market in the early 1990s. Since then the use of this particular technology has kept growing and recent studies show that it is used on regular basis by a large proportion of translators [158].

TM systems help translators by continuously trying to provide them with so-called matches, which are translation proposals retrieved from its database. These matches are identified by comparing automatically the segment that has

[1] Segments are typically sentences, but there are implementations which consider longer or shorter units.

to be translated with all the segments stored in the database. There are three kinds of matches: exact, fuzzy and no matches. Exact matches are found if the segment to be translated is identical to one stored in the TM. Fuzzy matches are used in cases where it is possible to identify a segment which is similar enough to the one to be translated, and therefore, it is assumed that the translator will spend less time editing the translation retrieved from the database than translating the segment from scratch. No matches occur in cases where it is not possible to identify a fuzzy match (i.e. there is no segment similar enough to the one to be translated to be worth using its translation).

TMs distinguish between fuzzy matches and no matches by calculating the similarity between segments using a similarity measure and comparing it to a threshold. Most of the existing TM systems rely on a variant of the edit distance as the similarity measure and consider a fuzzy match when the edit distance score is between 70% and 95%.[2] The main justification for using this measure is the fact that edit distance can be easily calculated, is fast, and is largely language independent. However, edit distance is unable to capture the similarity between segments correctly when different wording and syntactic structures are used to express the same idea. As a result, even if the TM contains a semantically similar segment, the retrieval algorithm will not be able to identify it in most of the cases. To make this clearer, consider following three sentences.

[2]It is unclear the origin for these value, but they are widely used by translators. Most of the tools allow translators to customise the value of this threshold according to their needs. Translators use their experience to decide which value for the threshold is appropriate for a given text.

1. I like Madrid which is such an attractive and exciting place.

2. I dislike Madrid which is such an unattractive and unexciting place.

3. I love Madrid as the city is full of attractions and excitements.

Consider sentence 2 and 3 already had their translations in the TM database and now the sentence 1 was to be translated. Most of the commercial TM systems based on edit distance would return sentence 2 as a fuzzy match to the incoming sentence as the edit distance between sentence 1 and 2 are lower than sentences 1 and 3. However, sentences 1 and 3 are semantically very close and would not need a lot of edits in the post-editing process. This nature of the edit distance based TM systems of not proving semantically close matches can hinder the efficiency of the translators.

Researchers tried to address this shortcoming of the edit distance metric by employing similarity metrics that can identify semantically similar segments even when they are different at token level. Section 6.1 discusses some of the approaches proposed so far. Most of these approaches incorporate simple operations like paraphrasing to the TM matching process. As we noticed in Part I of the thesis, deep learning based architectures are the state-of-the-art in STS. Therefore, in Part II of the thesis we propose a novel TM matching and retrieval method based on deep learning which has the capability to capture semantically similar segments in TMs better than the methods based on edit distance. As we discussed in Part I of the thesis, in addition to providing state-of-the-art results, deep learning based STS methods can be adopted in other domains and languages

easily which would be beneficial to TMs as TMs are employed in wide range of domains and languages.

Utilising deep learning in TM matching methods would bring obvious challenges regarding efficiency and storage. In Chapter 7, we discuss those challenges and carefully pick STS methods that can be efficient in TM matching process. We evaluate those methods on a real-world TM comparing them with the edit distance. As far as we know, this is the first study done on employing a deep learning based STS metric in TM matching and retrieval. The main contributions of this part of the thesis are,

1. We perform a rigorous analysis on existing TM matching algorithms and identify the main shortcomings in them.

2. We propose a novel TM matching and retrieval algorithm based on deep learning and evaluate it on a real-word TM using English-Spanish pairs.

3. We identify key challenges in employing deep learning in TM matching and we progressively develop a complete solution for the TM matching process using deep learning.

The remainder of this chapter is structured as follows. Section 6.1 discusses the various TM matching algorithms and their short comings. In Section 6.2, we introduce the real-word TM we used for the experiments in this part of the thesis. Section 6.3 shows the evaluation metrics that we used to evaluate the experiments. Chapter concludes with the conclusions.

6.1 Related Work

As we discussed before, even though TM systems have revolutionised the translation industry, these tools are far from being perfect. A serious shortcoming has to do with the fact that the (fuzzy) matching algorithm of most commercial TM systems is based on edit distance and no language processing is employed. Among the first ones to discuss the shortcomings were Macklovitch and Russell [159] who maintained that Translation Memory technology was limited by the rudimentary techniques employed for approximate matching. They comment that unless a TM system can perform morphological analysis, it will have difficulty recognising similar segments in the matching process.

The above shortcomings paved the way to the development of second-generation TM tools which had some language processing capabilities such as grammatical pattern recognition and performed limited segmentation at sub-sentence level. However, there are only a few commercially available second-generation TM systems such as *Similis* [160], *Translation Intelligence* [161] and Meta Morpho TM system, *Morphologic* [162]. *Similis* [160] performs linguistic analysis in order to split sentences into syntactic chunks or syntagmas, making it easier for the system to retrieve matches. *Morphologic* uses lemmas and part-of-speech information in order to improve matching, especially for morphologically rich languages like Hungarian [162]. Even though the second-generation TM tools solved some of the issues in first-generation TM tools, Mitkov and Corpas [163] notice that they still can not provide strong matches in most of the cases. Mitkov

and Corpas [163] show that none of the second-generation TM systems would be capable of matching *Microsoft developed Windows XP* with *Windows XP was developed by Microsoft* or matching *The company bought shares* with *The company completed acquisition of shares*.

To overcome this shortcoming Pekar and Mitkov [164] developed the so-called third-generation TM tools which analyse the segments not only in terms of syntax but also in terms of semantics. Pekar and Mitkov [164] perform linguistic processing over tree graphs [165, 166] followed by lexicosyntactic normalisation. Then similarity between syntactic-semantic tree graphs is computed and matches at sub-sentence level are established, using a similarity filter and a node distance filter. While this promising work was the first example of matching algorithms for future third-generation TM systems, the described approach was not deemed suitable for practical applications due to its very long processing time (it could take days to compare matches). Another method which performs matching at level of syntactic trees is proposed by Vanallemeersch and Vandeghinste [167]. The results presented in their paper are preliminary and the authors notice that tree matching method is "prohibitively slow" in this research too.

Further work towards the development of third-generation TM systems included paraphrasing and clause splitting. Raisa Timonera and Mitkov [168] experimented with clause splitting and paraphrasing, seeking to establish whether these NLP tasks would improve the performance of TM systems in terms of matching. Furthermore in to this, Gupta et al. [169] experimented with paraphrasing the TM with a view to securing more matches. The authors sought to

embed information from PPDB[3], a database of paraphrases [170], in the edit distance metric by employing dynamic programming (DP) [169] as well as dynamic programming and greedy approximation (DPGA) [171]. In more recent work, Gupta et al. [32] developed a machine learning approach for semantic similarity and textual entailment based on features extracted using typed dependencies, paraphrasing, machine translation, evaluation metrics, quality estimation metrics and corpus pattern analysis. This similarity method was experimented with to retrieve the most similar segments from a translation memory but the evaluation results showed that the approach was too slow to be used in a real world scenario [172].

With this analysis, we identified two main limitations in current third-generation TM systems. First, most of them rely on external knowledge-bases like WordNet, PPDB which makes it difficult to use in many languages and domains. Secondly, most of these approaches are slow to be used in real-world applications. To address this we propose to use deep learning based STS metrics we experimented in Part I of the thesis in TM matching. As aforementioned, they are not dependent on external knowledge-bases and most of them have been optimised to use effectively in real-world scenarios. Therefore, in Chapter 7 we evaluate those STS metrics in TM matching and retrieval. To best of our knowledge, deep learning methods have not been used successfully in translation memories.

[3]PPDP is available on http://paraphrase.org/#/download

6.2 Dataset

For the experiments of this part of the Thesis we used the DGT-Translation Memory[4] which has been made publicly available by the European Commissions (EC) Directorate General for Translation (DGT) and the ECs Joint Research Centre. It consists of sentences and their professional translations covering twenty-two official European Union (EU) languages and their 231 language pair combinations [173]. DGT-TM contains official legal acts. The translations are produced by highly qualified human translators specialised in specific subject domains. It is typically used by translation professionals in combination with TM software to improve the speed and consistency of their translations. We should note that the DGT TM is a valuable resource for translation studies and for language technology applications, including statistical machine translation, terminology extraction, named entity recognition, multilingual classification and clustering, among others [174, 175].

While we chose English-Spanish sentence pairs for the experiments of this study , our approach is easily extendable to any language pair. In this particular study, 2018 Volume 1 was used as experimental translation memory and 2018 Volume 3 was used as input sentences. The translation memory we built from 2018 Volume 1 featured 230,000 sentence pairs whilst, 2018 Volume 3 had 66,500 sentence pairs which we used as input sentences.

[4]DGT-TM is available to download at https://ec.europa.eu/jrc/en/language-technologies/dgt-translation-memory.

6.3 Evaluation

TM systems are usually evaluated by measuring the quality of the retrieved segment by the algorithm. Quality is considered to be the correspondence between retrieved segment and the reference translation: *"the closer a retrieved segment is to a reference translation, the better it is"*. Scores are calculated for individual segments by comparing them with the relevant reference translations. Those scores are then averaged over the whole corpus to reach an estimate of the quality of the TM system. These quality evaluating techniques are typically referred as automatic metrics for machine translation (MT) evaluation.

Over the years, researches have produced many automatic metrics for MT evaluations. BLEU (bilingual evaluation understudy) [176] can be considered as the most popular and oldest automatic metric. BLEU was one of the first metrics to claim a high correlation with human judgements of quality, and remains one of the most inexpensive metrics. However, using BLEU has drawbacks. The main drawback in BLEU is it does not consider meaning and does not directly consider sentence structure [177]. Since the goal of this study is to provide TM matches that are closer in meaning, we did not consider BLEU as our evaluation metric.

METEOR is a more recent automatic metric for MT evaluation which was designed to explicitly address several observed weaknesses in BLEU [178]. Similar to BLEU, METEOR is also based on an explicit word-to-word matching. However, unlike BLEU it does not only supports matching between words that are identical in the two strings being compared, but can also match words that are

simple morphological variants of each other (i.e. they have an identical stem), and words that are synonyms of each other. Considering these advantages in using METEOR, we employed METEOR as our evaluation metric for the experiments in this part of the Thesis.

It should be noted that the automatic evaluation metrics are far from being perfect [177]. They have their own limitations which in turn can affect the evaluations of this study. Whatever the automatic evaluation metric we use, we would not be able to completely avoid these weaknesses. Therefore, in addition to the automatic evaluation, we also carried out a human evaluation. We asked three native Spanish speakers with a background in translation studies to compare the segments retrieved from our algorithm. In Chapter 7, we report these results too alongside the automatic evaluation metrics.

6.4 Conclusions

The Translation Memory (TM) tools revolutionised the work of professional translators and the last three decades have seen dramatic changes in the translation workflow. One of the most important functions of TM systems is its ability to match a sentence to be translated against the database. However, most of the current commercial TM systems rely on edit distance to provide TM matches. Despite being simple, edit distance is unable to capture the similarity between segments and as a result even if the TM contains a semantically similar segment, the retrieval algorithm will not be able to identify it. This can hinder the performance of translators who are using the TM.

As a solution to this second-generation and third-generation TM systems are proposed. However, they are far from being perfect. Most of them lack the efficiency which is required for TM systems. Furthermore they rely on language specific knowledge-bases which makes them less adoptable to other languages and domains. Therefore, to overcome these shortcomings, we propose a novel TM matching and retrieval algorithm based on STS methods we experimented in Part I of the thesis. In addition to providing state-of-the-art STS results these algorithms are fast and easily adoptable to other languages and domain which would be beneficial for TMs.

We will be using English-Spanish sentence pairs in DGT translation memory as the dataset for our experiments. Our evaluation will be based on METEOR; an automatic metric for MT evaluation. Furthermore, considering the limitations in automatic metrics, we would be incorporating a human evaluation too in our experiments. The proposed method, results and evaluation will be explained in-detail in Chapter 7.

Chapter 7

Sentence Encoders for Translation Memories

Matching and retrieving previously translated segments from a Translation Memory is the key functionality in Translation Memories (TM) systems. In most of the commercial TM systems this matching and retrieving process is still limited to algorithms based on edit distance. However, edit distance is unable to capture correctly the similarity between segments when different wording and syntactic structures are used to express the same idea Mitkov and Corpas [163]. As a result, even if the TM contains a semantically similar segment, the retrieval algorithm will not be able to identify it in most of the cases which we have identified as a major drawback in Chapter 6.

Researchers tried to address this shortcoming of the edit distance metric in so called "third-generation" TM tools by employing similarity metrics that can identify semantically similar segments even when they are different at token level [164]. As we stated in Part I of the thesis, deep learning based architectures are the state-of-the-art in calculating STS between texts. Furthermore as we have shown in multiple times they are very easy to be adopted in different languages and domains. Therefore, having a deep learning based STS metric would be beneficial for TMs in many ways. In this Chapter, we are going to continue the idea

of "third-generation" TM tools by employing deep learning based STS metrics in TM matching and retrieving algorithms.

Recalling from Chapter 5, Transformers have set a new state-of-the-art performance on semantic textual similarity. However, it requires that both sentences are fed into the network, which causes a massive computational overhead [150]. Finding the most similar sentence to the incoming sentence in a collection of 100,000 sentences requires 100,000 inference computations (1 hour) with Transformers. This would not be efficient enough for TMs. Therefore, we had to take a step back from Transformers and look for alternative solutions in STS.

As we discussed in Part I of the thesis, next STS method in line with regard to performance was Siamese architectures we explored in Chapter 4. The advantage of the Siamese architectures is that they can also be used as sentence encoders. Therefore, they don't require to have both sentences in the network at inference time. Sentence embeddings for the sentences in TM can be calculated in advance and stored in a database. Then, when a new sentence comes in for the TM system, the algorithm needs to get the embeddings for that sentence and perform a simple similarity measure over the sentence embeddings in TM to find a match. This process would require less time compared to transformers. Therefore, we utilised the best Siamese architecture we had in Part I of the thesis; Sentence-BERT [150] in the TM experiments we perform in this Chapter. In order to have a diverse set of algorithms, we also used best sentence encoders we had in Chapter 3; Infersent [41] and Universal Sentence Encoder [84]. As far as we know, this would be the first study to employ deep learning in TM systems.

We address two research questions in this chapter:

RQ1: Are the sentence encoders efficient enough for TM matching and retrieval task?

RQ2: How does the sentence encoders perform in TM retrieving task compared to other TM tools?

The main contributions of this chapter are as follows.

1. We evaluated three sentence encoders for TM retrieval task in English-Spanish segments using a real-world TM; DGT-TM. We compare the results against a popular TM system; Okapi[1]; which uses edit distance for the retrieval process.

2. Evaluations were carried out separately for different fuzzy match ranges and we show that in certain fuzzy match ranges sentence encoders outperform Okapi.

3. We further perform a detailed human evaluation on the matches retrieved from sentence encoders and Okapi, collaborating with three native Spanish speakers with a translation background. We show that sentence encoders generally provide better matches then Okapi.

4. The code with the experiments conducted are publicly available to the community[2].

[1]The Okapi Framework is a cross-platform and free open-source set of components and applications that offer extensive support for localising and translating documentation and software. It is available on https://okapiframework.org/. We specifically used the Rainbow application available in the framework which allows bulk matching and retrieval from a translation memory.

[2]The public GitHub repository is available on https://github.com/tharindudr/

The rest of this chapter is organised as follows. Section 7.1 describes motivation for the study comparing the performance of edit distance against sentence encoders in STS task. In section 7.2 we present the methodology we used to incorporate sentence encoders in to TM systems. Section 7.3 presents the results we got with sentence encoders for English-Spanish sentence pairs in DGT-TM. In section 7.4, we provide a detailed human evaluation done by three native Spanish speakers identifying the strengths and weaknesses of the proposed approach. The chapter finishes with conclusions and ideas for future research directions in TM matching and retrieving.

7.1 Motivation

We first evaluated the edit distance in two STS datasets introduced in Chapter 1; SICK and STS 2017. We compared those results to the results we got from sentence encoders in Chapter 3. Considering the accuracy of the STS task, we used Infersent2 from the pre-trained Infersent models, transformer encoder from the pre-trained Universal Sentence Encoder models and stsb-roberta-base-v2 from the pre-trained SBERT models which is based on RoBERTa [22].

With SICK dataset, edit distance achieve only 0.361 Pearson correlation while Infersent, Universal Sentence Encoder and SBERT achieve 0.769, 0.780 and 0.892 Pearson correlation respectively. Similarly with STS 2017 dataset too sentence encoders outperform edit distance by a large margin. This is a clear indication that sentence encoders can calculate the text similarity better than edit distance.

`intelligent-translation-memories`

Sentence 1	Sentence 2	GOLD	ED	Infersent	USE	SBERT
Israel expands subsidies to settlements	Israel widens settlement subsidies	1.0000	0.0214	0.8524	0.8431	0.8997
A man plays the guitar and sings.	A man is singing and playing a guitar.	1.0000	0.0124	0.7143	0.7006	0.8142
A man with no shirt is holding a football	A football is being held by a man with no shirt	1.0000	0.0037	0.9002	0.8852	0.9267
EU ministers were invited to the conference but canceled because the union is closing talks on agricultural reform, said Gerry Kiely, a EU agriculture representative in Washington.	Gerry Kiely, a EU agriculture representative in Washington, said EU ministers were invited but canceled because the union is closing talks on agricultural reform.	1.0000	0.1513	0.7589	0.7865	0.8190

Table 7.1: Examples sentence pairs where sentence encoders performed better than edit Distance in the STS task. GOLD column shows the score assigned by humans, normalised between 0 and 1. The ED column shows the similarity obtained regarding the edit distance. Infersent, USE and SBERT columns show the similarity obtained by Infersent, Universal Sentence Encoder and SBERT respectively.

To further confirm this, we analysed the results of individual sentence pairs. Table 7.1 shows some of the example sentence pairs from STS2017, where sentence encoders showed promising results against edit distance.

As can be seen in table 7.1 all the sentence encoders handle semantic textual similarity better than edit distance in many cases where the word order is changed in two sentences, but the meaning remains same. This detection of similarity even when the word order is changed will be important in segment matching and retrieval in TMs which is the motivation for this study.

7.2 Methodology

We conducted the following steps for all three sentence encoders we mentioned before; Infersent, Universal Sentence Encoder and SBERT. We used the same pre-trained models used in Section 7.1. As discussed in Chapter 6, for all the

experiments we used DGT-TM 2018 Volume 1 as the translation memory and 2018 Volume 3 - as the source for input sentences.

Step 1 : Calculated the sentence embeddings for each segment in the translation memory (230,000 segments) and stored the vectors in AquilaDB[3]. AquilaDB is a decentralized vector database to store feature vectors and perform k-nearest neighbours (KNN) retrieval. It is build on top of popular Apache CouchDB[4]. A record of the database has 3 fields: source segment, target segment and source segment vector.

Step 2 : Calculated the sentence embedding for one incoming segment.

Step 3 : Calculated the cosine similarity of that embedding with each of the embedding in the database using equation 3.1. We retrieve the embedding that had the highest cosine similarity with the input segment embedding and retrieve the corresponding target segment for the embedding as the translation memory match. We used *'getNearest'* functionality provided by AquilaDB for this step.

The efficiency of the TM matching and retrieval is a key-factor for translators who are using them. As we discussed in Chapter 6, most of the proposed third-generation TM systems were not efficient enough to be used in real-word scenarios. This was the reason why they are not popular in the community. We wanted to avoid doing the same mistake with our proposed approach in order to

[3]AquilaDB is available on https://github.com/Aquila-Network/AquilaDB
[4]CouchDB is available on https://github.com/apache/couchdb

make it more useful for the community. Therefore, as the first step, we calculated the efficiency of the proposed sentence encoders.

Table 7.2 discusses the efficiency of each sentence encoder. The experiments were done in a Intel(R) Core (TM) computer with i7-8700 CPU and 3.20GHz clock speed. While the performance of the sentence encoders would be more efficient in a GPU (Graphics Processing Unit), we carried our experiments in CPU (Central Processing Unit) since the translators using translation memory tools would not have access to GPU on daily basis.

Architecture	**Step 1**	**Step 2**	**Step 3**
USE	108s	1.23s	0.40s
Infersent	496s	0.022s	0.40s
SBERT	1102s	0.052s	0.52s

Table 7.2: Time for each step with experimented sentence encoders.

The translation memory was processed in batches of 256 sentences with a view to obtaining sentence embeddings. As seen in Table 7.2, the Universal Sentence Encoder(USE) was the most efficient encoder delivering sentence embeddings within 108 seconds for 230,000 sentences. At the other end was Sentence-BERT which took 1102 seconds to derive the sentence embeddings for the same number of sentences in the translation memory. Even though these times may appear very long, we should keep in mind that this process needs to be done only once as they are kept in a database, and do not need to be computed again.

The second column of the Table 7.2 reports the time needed for each sen-

tence encoder to embed a single sentence. Input sentences were not processed in batches as was done for the TM sentences. The rationale behind this decision was the fact that translators translate sentences one by one. An interesting observation is that while the Universal Sentence Encoder was the most efficient in generating sentence embeddings in batches, it was the least efficient encoder one to derive the embeddings for single sentences. It took 1.23 seconds to encode a single sentence. InferSent was the fastest sentence encoder for a single sentence.

The third column of the Table 7.2 reports the time needed to retrieve the best match from the translation memory. This includes the time taken to compute cosine similarity between the embeddings of TM sentences and the embeddings of the input sentence. Also, it includes the time taken to sort the similarities, get the index of the highest similarity and retrieve the TM sentence considered as perfect match for the input sentence. As shown in Table 1, all sentence encoders needed approximately 0.5 seconds only to perform this operation. As a whole, to identify the best match from the translation memory, InferSent and Sentence-BERT encoders did not take more than 1 second while Universal sentence encoder took 1.6 seconds which is considered good enough for an operational translation memory tool.

With these observations, we answer our **RQ1**: sentence encoders are efficient enough for TM matching and retrieval task. The numbers we calculated for each step provide evidence that the proposed methods are fast enough to be used in a real-world environment and bring huge improvement over the existing third-generation TM systems with regards to efficiency.

7.3 Results and Evaluation

In this section we report the results for the evaluation of the three selected sentence encoders. We ran automatic evaluation experiments by comparing the matches returned by Okapi which uses a simple variant of edit distance as the retrieving algorithm and the matches returned by each of the sentence encoders. With a view to measuring the quality of a retrieved segment, the METEOR score was computed between the translation of the incoming sentence as present in the DGT-TM 2018 and the translation of the match retrieved from the translation memory. This process was repeated for the segments retrieved by our deep learning methods and those retrieved using Okapi.

Fuzzy score	Okapi	USE	Infersent	SBERT	Amount
0.8-1.0	**0.931**	0.854	0.864	0.843	1624
0.6-0.8	0.693	0.702	**0.743**	0.698	4521
0.4-0.6	0.488	0.594	**0.630**	0.602	6712
0.2-0.4	0.225	0.318	**0.347**	0.316	13136
0-0.2	0.011	0.128	**0.134**	0.115	24612

Table 7.3: Result comparison between Okapi and the sentence encoders for each partition. Fuzzy score column represents the each partition. Okapi column shows the average METEOR score between the matches provided by the Okapi and the actual translations in that partition. USE, Infersent and SBERT columns show the average METEOR score between the matches provided by each of the sentence encoders and the actual translations in that partition. Amount column shows the number of sentences in each partition. Best result for each partition is shown in bold.

For a better comparison, we first removed the sentences where the matches provided by Okapi and the sentence encoders were same. Next, in order to analyse the results, we divided the results into 5 partitions according to the fuzzy

match score retrieved from Okapi: 0.8 - 1, 0.6 - 0.8, 0.4 - 0.6, 0.2 - 0.4, and 0 - 0.2. The ranges were selected to understand the behaviour of the sentence encoders in TM matching and retrieval task. The first partition contained the matches derived from Okapi that had a fuzzy match score between 0.8 and 1. We calculated the average METEOR score for the segments retrieved from Okapi and for the segments retrieved from each of the sentence encoders in this particular partition. We repeated this process for all the partitions. Table 7.3 lists the results for each sentence encoder and Okapi.

As can be seen from Table 7.3, for the fuzzy match score range 0.8-1.0, Okapi METEOR score mean is higher than any of the mean METEOR score of the sentence encoders which indicates that matches returned in that particular fuzzy match score range by Okapi are better than the matches returned by any of the sentence encoders. However, in rest of the fuzzy match score ranges the sentence encoders outperform Okapi which shows that for the fuzzy match score ranges below 0.8, the sentence encoders offer better matches than Okapi. From the sentence encoders InferSent performs better than both the Universal Sentence Encoder and SBERT. The results in Table 7.3 show that when there are close matches in the Translation Memory, edit distance delivers better matches than the sentence encoders. However, when the edit distance fails to find a proper match in the TM, the match offered by the sentence encoders will be better.

Usually, the TM matches with lower fuzzy match scores (< 0.8) are not used by professional translators, or when used, they lead to a decrease in translation productivity. But our method can provide better matches to sentences below

fuzzy match score 0.8, hence will be able to improve the translation productivity. According to the annotation guidelines of STS tasks which we explained in Chapter 1 a STS of 0.8 means *"The two sentences are mostly equivalent, but some unimportant details differ"* and semantic textual similarity score of 0.6 means *"The two sentences are roughly equivalent, but some important information differs/missing"*. If we further analyse the fuzzy match score range 0.6-0.8, as shown in table 7.3, the mean semantic textual similarity for the sentences provided by Infersent is 0.743. Therefore, we can assume that the matches retrieved from the Infersent in the fuzzy match score range 0.6-0.8 will help to improve the translation productivity.

With these observations we answer our **RQ2:** sentence encoders can improve TM matching and retrieval especially in the lower fuzzy match scenarios. The proposed process would upgrade the current third-generation TM tools as it provides good results and as it is considerably efficient.

7.4 Error Analysis

As we mentioned in Chapter 6, automatic machine translation evaluation metrics like METEOR are far from being perfect. Given the fact that METEOR relies largely on string overlap we assumed that it is unable to capture the fact that the segments retrieved using the sentence encoders are semantically equivalent. Therefore, a human evaluation is required for a study like this. In this section, we carried out a human evaluation in the form of an error analysis.

Three native Spanish speakers with backgrounds in translation went through

the matches provided by the sentence encoders and the matches provided by Okapi. The usual pattern they found was that the sentence encoders returned better results; however, there were a limited number of cases where Okapi performed better. The native speakers analysed more than one thousand segments and below is a brief analysis of the typical error cases they found.

In a number of cases InferSent performed better than Okapi because the latter proposed translations which contained information which did not appear in the English input segment. As an illustration of this typical case, for the input segment (1) for which the correct translation is (2) Okapi retrieved (3) whilst InferSent selected (4), which is more appropriate.

(1) The audit shall include.

(2) La evaluación incluirá.

(3) Los indicadores de rendimiento incluirán. (Key performer indicators shall include)

(4) El informe incluirá. (The report shall include)

In other cases, Okapi selects segments which capture only a part of the meaning of the input segment correctly but fails to provide its whole meaning. For example, for the input segment (5) Okapi selects (6). Due to its exclusive reliance on edit distance, Okapi selects a segment which has the correct temporal expression (16 June/16 de junio) but the rest of the retrieved translation does not have any connection with the original. In contrast, InferSent is able to retrieve a segment

which conveys the meaning correctly but has the incorrect date (7). From the point of view of the effort required to produce accurate translation, the segment selected by InferSent requires less effort (as the translator would have to correct the date only) than the one selected by Okapi.

(5) It shall apply in all Member States from 16 June 2020.

(6) A partir del 16 de junio de 2024, los Estados miembros utilizarán la función de registro centralizada. (Member States will use the centralised registration function from 16 June 2024)

(7) Los Estados miembros aplicarán dichas disposiciones a partir del 21 de diciembre de 2020. (These provisions shall apply in all Member States from 21 December 2020)

The advantage of sentence encoders can also be observed when comparing the performance of Okapi with the Universal Sentence Encoder. Okapi often recognises a part of the English sentences only, so the match that is suggested is only partially correct. As an illustration, the short sentence (8) is retrieved from Okapi as (9). The word brief does not appear in the retrieved text and, additionally, Okapi adds de las mercancías. The translation retrieved by the Universal Sentence Encoder is correct (10). This pattern can also be seen when comparing Okapi with SBERT. By way of example, while proposed match for (11) by SBERT is correct (12), Okapi only recognises one word of the segment, as the retrieved translation is (13).

(8) Brief description

(9) Descripción de las mercancías (Goods description)

(10) Breve descripción

(11) Test equipment

(12) Los equipos de ensayo (The test equipment)

(13) Equipo informático (IT equipment)

In general, and on a number of occasions, Okapi omits some of the information that the sentence encoders identify. The equivalent of the sentence (14) is retrieved by Okapi as (15) with Edible offal missing in Okapis proposal. The sentence retrieved from InferSent however, conveys this information (16).

(14) Edible offal of bovine animals, frozen

(15) De la especial bovina, congelados (Bovine animals, frozen)

(16) Carne de animales de la especie bovina, congelada.

Okapi often fails not only with whole sentences but also with segments that only contain one word. When retrieving the translation of the word (17) the sentence encoder InferSent suggest (18), whereas Okapi also adds the word Lugar (19). This also happens with (20), which InferSent returns as (21), whereas Okapi retrieves (22); the word elección (choice) does not appear in the English sentence. In addition, Okapi often fails with multiword expressions. The translation of the

multiword expression (23) is retrieved by Okapi as (24) and in this case the proposed match features redundant information. The segment retrieved by SBERT represents the best solution (25).

(17) Date

(18) Fecha

(19) Lugar y fecha (Place and date)

(20) Fuel

(21) Combustible

(22) elección del combustible (choice of fuel)

(23) Engine type

(24) Potencia del motor principal en KW: Marca: Tipo (Main engine power in KW: Make: Type)

(25) Tipo de motor

There are cases where the segment retrieved from the sentence encoder is similar to the one retrieved from Okapi, but the sentence encoder is better at conveying subtle nuances. For instance, the proposed translation for sentence (25) by Okapi is (27) and the sentence retrieved from the Universal Sentence Encoder is (28). The nuance refers to the proposed translation for as appropriate.

Okapi returns (29), whereas the Universal Sentence Encoder retrieves the correct translation (30). Another similar example where Okapi fails is the retrieved translation of (31) as (32); the Universal Sentence Encoder acts correctly on this occasion.

(26) This Decision shall be kept under constant review and shall be renewed or amended, as appropriate, if the Council deems that its objectives have not been met.

(27) Se prorrogará o modificará, si procede, en caso de que el Consejo estime que no se han cumplido los objetivos de la misma. (This Decision shall be renewed or amended, if appropriate, if the Council deems that its objectives have not been met)

(28) Será prorrogada o modificada, según proceda, si el Consejo considera que no se han cumplido sus objetivos. (This Decision shall be renewed or amended, as appropriate, if the Council deems that its objectives have not been met)

(29) si procede (if appropriate)

(30) según proceda (as appropriate)

(31) if applicable

(32) no procede (not applicable)

There are a number of cases where Okapi returns a completely incorrect translation as opposed to the sentence encoders. For (33) Okapi proposed (34) which has nothing to do with the original meaning. The Universal Sentence Encoder offers a simple, yet good solution (35). Another similar example is the suggested match for (36): Okapi returns a completely wrong translation (37), the proposal by SBERT while incorrect, is semantically similar (38).

(33) None of the above

(34) Veánse los considerandos 92 a 94 (See items 92 to 94)

(35) Ninguna (None)

(36) limes

(37) Reducir al mínimo los tiempos de permanencia (Minimise dwell times)

(38) de lima (made of limes)

There are a limited number of cases where Okapi fares better than the sentence encoders. One such example is when encoders retrieve matches of named entities. By way of illustration, the translation the Universal Encoder retrieves for (39) is (40) instead of (41); SBERT retrieves (42) when the original segment is (43); and the proposal by InferSent for (44) is (45).

(39) Japan

(40) Israel

(41) Japón

(42) Singapur (Singapore)

(43) Philippines

(44) within municipality of Sitovo

(45) en el municipio de Alfatar (within municipality of Alfatar)

Finally, and occasionally, sentence encoders too could propose translations featuring redundant information which does not appear in the English original segment. The match InferSent returns for (46) is (47) and in this case Okapi retrieves a correct translation (48). On another, isolated occasion, SBERT also adds a redundant word mixto (joint) by proposing (49) as translation for (50). In this particular instance the retrieved match by Okapi is correct (51).

(46) Requirements

(47) Requisitos del Eurosistema (Eurosystem requirements)

(48) Requisitos

(49) El Comité mixto adoptará su reglamento interno (The joint Committee shall establish its own rules of procedures)

(50) The Committee shall establish its own rules of procedures

(51) El Comité dispondrá su reglamento interno

With this analysis too it is clear that sentence encoders provide better matches than Okapi in most of the scenarios. It further confirms our answer to **RQ2** that sentence encoders can be used to improve the matching and retrieval process in TMs.

7.5 Conclusions

Third-generation TM tools have addressed the limitations in traditional TM tools. Yet, they are not popular in the community since they are largely inefficient and there is not much performance gain in using them. To address this we propose to use deep learning based STS metrics that we experimented in Part I of the thesis in TMs. Considering the accuracy and efficiency, we pick three sentence encoders; Infersent, Universal Sentence Encoder and SBERT and design a TM matching algorithm based on them. We evaluate the proposed algorithm in a real world TM; DGT-TM. We compared the results from each of the sentence encoders with the results from Okapi which uses edit distance to acquire the best match from the translation memory. The results show that for sentences with a fuzzy match score less than 0.8 in Okapi, the sentence encoders return better matches than simple edit distance. Of the sentence encoders, InferSent fares best. We also discuss the results of the analysis of typical errors where three native Spanish speakers analysed the matches proposed by the sentence encoders and Okapi. The error analysis further confirms that sentence encoders can be used to improve the matching and retrieval process in TMs [179].

The main limitation of the proposed algorithm is the time taken to retrieve a

match can be high with a large TM. This is a common problem for Deep learning applications which is usually solved by employing GPUs. However, in this case it would not be feasible to use GPUs since they are expensive and translators using translation memory tools would not have access to GPUs on daily basis. To overcome this impediment, we envisage the deployment of algorithms to filter out the sentences from the TM before the retrieval process and to make the calculation of cosine similarity between vectors a computationally less intensive process. Faster algorithms generating sentence embeddings like averaging word embeddings which we discussed in Chapter 2 will be used in these experiments.

The automatic evaluation metric that we used in this study; METEOR has its own limitations that might have affect the evaluation of this study. Very recently, new automatic MT evaluation metrics such as BLEURT [177] which are based on transformers have been proposed. Unlike METEOR, these metrics do not largely rely on string overlap and would be more suitable for a study like this. Therefore, as future work, we hope incorporate these new automatic MT evaluation metrics in our study.

With this, we conclude Part II of the thesis; using deep learning based STS metrics in translation memories. We showed that the STS methods we experimented in Part I of thesis can be employed successfully in TMs. Our methods outperform edit distance based TM matching and retrieval algorithms. Furthermore, the proposed method is very efficient and can be used in real world scenarios. Therefore, we believe that the findings of the Part II of the thesis, would pave a new direction in third-generation TM systems.