Contents

List of Tables

List of Figures

List of Publications

Parts of this thesis have appeared in the following peer-reviewed publications:

Ranasinghe, Tharindu, Ruslan Mitkov, Constantin Orasan, and Rocío Caro Quintana (2021a). "Semantic textual similarity based on deep learning: Can it improve matching and retrieval for Translation Memory tools?" In: *Corpora in Translation and Contrastive Research in the Digital Age: Recent advances and explorations*. Ed. by Julia Lavid-López, Carmen Maíz-Arévalo, and Juan Rafael Zamorano-Mansilla. John Benjamins. Chap. 4. URL: https://benjamins.com/catalog/btl.158.04ran.

Ranasinghe, Tharindu, Constantin Orasan, and Ruslan Mitkov (Sept. 2019a). "Enhancing Unsupervised Sentence Similarity Methods with Deep Contextualised Word Representations". In: *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*. Varna, Bulgaria: INCOMA Ltd., pp. 994–1003. DOI: 10.26615/978-954-452-056-4_115. URL: https://www.aclweb.org/anthology/R19-1115.

Ranasinghe, Tharindu, Constantin Orasan, and Ruslan Mitkov (Sept. 2019b). "Semantic Textual Similarity with Siamese Neural Networks". In: *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*. Varna, Bulgaria: INCOMA Ltd., pp. 1004–1011. DOI:

10.26615/978-954-452-056-4_116. URL: https://www.aclweb.org/anthology/R19-1116.

Ranasinghe, Tharindu, Constantin Orasan, and Ruslan Mitkov (Nov. 2020a). "Intelligent Translation Memory Matching and Retrieval with Sentence Encoders". In: *Proceedings of the 22nd Annual Conference of the European Association for Machine Translation*. Lisboa, Portugal: European Association for Machine Translation, pp. 175–184. URL: https://www.aclweb.org/anthology/2020.eamt-1.19.

Ranasinghe, Tharindu, Constantin Orasan, and Ruslan Mitkov (Nov. 2020b). "TransQuest at WMT2020: Sentence-Level Direct Assessment". In: *Proceedings of the Fifth Conference on Machine Translation*. Online: Association for Computational Linguistics, pp. 1049–1055. URL: https://aclanthology.org/2020.wmt-1.122.

Ranasinghe, Tharindu, Constantin Orasan, and Ruslan Mitkov (Dec. 2020c). "TransQuest: Translation Quality Estimation with Cross-lingual Transformers". In: *Proceedings of the 28th International Conference on Computational Linguistics*. Barcelona, Spain (Online): International Committee on Computational Linguistics, pp. 5070–5081. DOI: 10.18653/v1/2020.coling-main.445. URL: https://aclanthology.org/2020.coling-main.445.

Ranasinghe, Tharindu, Constantin Orasan, and Ruslan Mitkov (Aug. 2021b). "An Exploratory Analysis of Multilingual Word-Level Quality Estimation with Cross-Lingual Transformers". In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Online: Association for Computational Linguistics, pp. 434–440. DOI: 10.18653/v1/2021.acl-short.55. URL: https://aclanthology.org/2021.acl-short.55.

Introduction

Semantic textual similarity (STS) is a natural language processing (NLP) task to assess the semantic similarity between two text snippets quantitatively. STS is a fundamental NLP task for many text-related applications, including text de-duplication, paraphrase detection, semantic searching, and question answering. Measuring STS is a machine learning (ML) problem, where an ML model predicts a value that represents the similarity of the two input texts. These machine learning models can be categorised into two main areas; supervised and unsupervised. Supervised STS ML models have been trained on an annotated STS dataset, while unsupervised models predict STS without trained on annotated STS data. In **Part I** of the thesis, we explore supervised and unsupervised ML models in STS. We explore embedding aggregation based STS methods, sentence encoders, Siamese neural networks and transformers in STS. Furthermore, for each STS method, we analyse the ability of the model to perform in a multilingual and multi-domain setting. In the process, we introduce a new state-of-the-art unsupervised vector aggregation based STS method developed on contextual word embeddings and a new state-of-the-art supervised STS method based on Siamese neural networks and classical word embedding models.

The second and third parts of the thesis focus on applying the developed STS methods in the applications of translation technology; translation memories

(TM) and translation quality estimation (QE). In **Part II** of the thesis, we identify that the edit distance based matching and retrieval algorithms in TMs are 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. To overcome this, we propose semantically powerful algorithms for TM matching and retrieval. Considering the efficiency, we employ a TM matching and retrieval algorithm based on sentence encoders we experimented with in Part I of the thesis. We empirically show that this algorithm outperforms edit distance-based matching algorithms paving a new direction for TMs.

As the next application, we utilise the STS architectures we developed in Part I of the thesis in translation quality estimation. We identify that the current state-of-the-art neural QE models are very complex and require a lot of computing resources. To overcome this, we remodel the QE task as a cross-lingual STS task. We show that the STS architectures can be successfully applied in QE by changing the input embeddings into cross-lingual embeddings, and they are very simple and efficient compared to the current state-of-the-art QE models. Based on that, we develop TransQuest - a new state-of-the-art QE framework that won the WMT 2020 QE shared task. TransQuest supports both word-level and sentence-level QE and has been evaluated on more than 15 language pairs. Furthermore, for the first time, we explore multilingual QE models with TransQuest, focussing on low resource languages. We release TransQuest as an open-source QE framework, and by the time of writing this, TransQuest has more than 9,000 downloads from the community.

The research questions in this work can be summarised as the following:

RQ1 What are the available supervised and unsupervised STS methods, and how do they perform in multilingual and multi-domain environments?

RQ2 Can the neural STS methods be applied in TMs? How efficient and effective are they compared to the real-world TM tools?

RQ3 Can the state-of-the-art STS methods be adopted in the QE task? Can these simple STS architectures outperform current complex QE methods?

Each part of the thesis will address these research questions separately.

Part I

Semantic Textual Similarity

Chapter 1

Introduction to Semantic Textual Similarity

Semantic Textual Similarity (STS) measures the equivalence of meanings between two textual segments. In Natural Language Processing (NLP), measuring semantic similarity between two textual segments plays an important role. It is one of the fundamental tasks for many NLP applications and their related areas. To measure STS, the input is always two textual segments, and the output is a continuous value that represents the degree of similarity of the two input textual segments (Agirre et al., 2012). These segments can be a short snippet of texts, complete sentences or even documents (Agirre et al., 2013).

STS is related to both textual entailment (TE) and paraphrasing but differs in several ways. TE is the task of identifying the relationship between two texts commonly addressed as text (t) and hypothesis (h). Entailment, contradiction, and neutral are the most popular relationship types in TE (Dagan et al., 2006; Marelli et al., 2014). On the other hand, paraphrasing identification is the task of recognising text fragments with approximately the same meaning within a specific context (Vrbanec and Meštrović, 2020). Therefore, TE and paraphrasing give a categorical output while STS identifies the degree of equivalence of two texts as a continuous value.

Measuring STS is an important research problem, having many applications in NLP such as information retrieval (IR) (Varelas et al., 2005; Subhashini and Kumar, 2010), text summarisation (Aliguliyev, 2009; Schallehn et al., 2004), question answering (Mohler et al., 2011), relevance feedback (Wang et al., 2020a), text classification (Li and Han, 2013; Albitar et al., 2014) and word sense disambiguation (Abdalgader and Skabar, 2011). In the field of databases, text similarity can be used for schema matching. In the document databases like Elasticsearch[1], there is a core module called *"Similarity module"* that defines the document matching process. Furthermore, STS is also useful for relational join operations in databases where join attributes are textually similar to each other (Cohen, 2000; Schallehn et al., 2004). Also, STS is widely used in semantic web applications like community extraction (Zhang et al., 2010), Twitter search (Feng et al., 2013) where it is required the ability to measure semantic relatedness between concepts or entities accurately.

These applications require to measure STS automatically, which means that computer programs should be developed to calculate STS between two textual inputs. The most natural way to approach this problem is to use a machine learning approach where the computer learns from examples. The development of rule-based methods would be too cumbersome, and it is unlikely to lead to robust solutions. Over the years, researchers have proposed numerous ML solutions for STS. These ML solutions can be categorised into two main

[1]Elasticsearch is a document database based on the Lucene library. It is available on https://www.elastic.co/. More information on the Similarity module is available on https://www.elastic.co/guide/en/elasticsearch/reference/current/index-modules-similarity.html

categories; (a) Linguistic feature-based (b) Vector/ Embedding-based . Most of the early approaches belong to the linguistic feature-based category. With this, features for the ML algorithm were hand-crafted. Such features include edge-distances between nodes in WordNet (Miller, 1995), number of named entities in two input texts, corpus pattern analysis features etc. Then these features would be fed into an ML algorithm such as Support Vector Machine (SVM), Linear Regression etc. (Béchara et al., 2015). This ML algorithm will be trained on an annotated STS dataset and then can be used to measure STS automatically. Despite being extremely popular before the neural network era, linguistic feature-based algorithms have limitations. Determining the best linguistic features for calculating STS is not an easy task as it requires a good understanding of the linguistic phenomenon and relies on researchers' intuition. In addition, most of these features depend on lexical knowledge bases like WordNet, which makes it difficult to adopt them in languages other than English. Furthermore, these features primarily rely on parsers that are not available in other languages (Ranjan et al., 2016). However, these methods' most significant limitation would be that they no longer provide strong results compared to the vector-based methods (Cer et al., 2017).

With the introduction of word embeddings (Mikolov et al., 2013a), ML solutions in NLP shifted from feature-based methods to vector-based methods. Pre-trained word embedding models such as word2vec (Mikolov et al., 2013a), GloVe (Pennington et al., 2014), fastText (Mikolov et al., 2018) etc. provide a learned representation for texts where the words with the same meaning have a

similar representation. Since these word embeddings are already semantically powerful, ML solutions no longer require to depend on lexical knowledge bases. As a result, embedding based ML solutions are easy to adopt in different languages as long as the pre-trained embeddings are available in that language. Furthermore, these solutions are now state-of-the-art in NLP tasks, including STS, providing stronger results than feature-based ML solutions (Cer et al., 2017). Therefore, as STS solutions in this part of the thesis, we mainly explore embedding based ML approaches.

Similar to general ML algorithms, vector-based ML STS algorithms can too be classified into two main categories; Supervised and Unsupervised. In supervised learning, ML models will be trained using labelled data. Therefore, supervised ML algorithms require data that the humans already annotated with the closest answer. On the other hand, for unsupervised learning, you do not need an annotated dataset. Unsupervised ML approaches would discover the features by themselves. Given that annotated STS data is not commonly available in many languages and domains, exploring both supervised and unsupervised STS methods is essential. Therefore, the first two chapters in this part of the thesis explore unsupervised STS methods, while the last two chapters explore supervised STS methods.

The most common unsupervised STS approaches are vector aggregation methods like Word Vector Averaging, Word Mover's Distance (Kusner et al., 2015) and Smooth Inverse Frequency (Arora et al., 2017). In Chapter 2, we explore them in detail. We identify the best vector aggregation method empirically by

analysing them in different STS datasets. Finally, we propose a new state-of-the-art vector aggregation method based on contextual word embeddings that outperforms other methods.

In Chapter 3, we explore another unsupervised STS method using sentence encoders. Sentence encoders are different from vector aggregation methods as they have end-to-end models to get sentence embeddings rather than a simple aggregation method. They provide strong results compared to other unsupervised STS methods. We use three different sentence encoders and analyse their performance in various aspects of English STS and also evaluate their portability to different languages and domains.

In Chapters 4 and 5, we explore most popular supervised STS approaches. Usually, in supervised vector-based STS approaches, word embeddings would be fed into a neural network like tree-structured neural networks (Tai et al., 2015) and Siamese neural networks (Mueller and Thyagarajan, 2016). Among them, Siamese neural networks have been widely used in STS and have additional advantages compared to other structures. Therefore, we discuss them comprehensively in Chapter 4. We evaluate the existing Siamese Neural Network architectures in STS datasets and propose a novel Siamese Neural Network architecture for smaller STS datasets that outperforms current state-of-the-art Siamese neural models. We also assess its performance in different languages and domains.

In the final chapter of Part I of this thesis, we explore the newly released transformers in STS tasks. Transformers have taken the NLP field by storm,

providing very successful results in various NLP tasks. In Chapter 5, we bring together various transformer architectures (Devlin et al., 2019; Yang et al., 2019b; Liu et al., 2019) and investigate their performance in various STS datasets. We explore the strengths and weaknesses of transformer models regarding accuracy and efficiency and discover the possible solutions for their limitations.

The main contributions of this part of the thesis are as follows.

1. Each chapter covers various supervised and unsupervised techniques to compute semantic textual similarity that benefits a wide range of NLP applications. We empirically evaluate all of them in three English datasets, two non-English datasets and an out of domain dataset to explore their adaptability.

2. We propose a novel unsupervised STS method based on contextual word embeddings that outperforms current state-of-the-art unsupervised vector aggregation STS methods in all the English datasets, non-English datasets, and datasets in other domains.

3. We propose a novel Siamese neural network architecture that is efficient and outperforms current state-of-the-art Siamese neural network architectures in smaller STS datasets.

4. We provide important resources to the community. The code of each chapter as an open-source GitHub repository and the pre-trained STS models will be freely available to the community. The link to the GitHub

repository and the models will be unveiled in the introduction section of each chapter.

The remainder of this chapter is structured as follows. Section 1.1 discusses the various datasets we used in *"Semantic Textual Similarity"* part of the thesis and briefly analyse the datasets for common properties. In Section 1.2, we discuss the main evaluation metrics we used in the *"Semantic Textual Similarity"* part of the thesis. The chapter finishes with the conclusions.

1.1 Datasets

The popularity of STS is partially owed to the large number of shared tasks organised in SemEval from 2012-2017 (Agirre et al., 2012; Agirre et al., 2013; Agirre et al., 2014; Agirre et al., 2015; Agirre et al., 2016; Cer et al., 2017). First, they have provided annotated datasets that can train STS ML models and evaluate them. Second, at the end of each shared task, the solutions submitted by the participants are published, and the best solutions can be considered as state-of-the-art STS methods.

To maintain the versatility of our methods, we experimented with several English STS datasets as well as several non-English datasets and a dataset from a different domain which we will discuss later in this section. These datasets carry different and interesting characteristics. Therefore, with the introduction, we also do an exploratory analysis of the dataset focussing on various properties. All of the datasets which are described here are publicly available and can be considered as STS benchmarks.

1.1.1 English Datasets

1. **SICK dataset**[2] - The SICK data contains 9927 sentence pairs with a 5,000/4,927 training/test split which were employed in the *SemEval 2014 Task1: Evaluation of Compositional Distributional Semantic Models on Full Sentences through Semantic Relatedness and Textual Entailment* (Marelli et al., 2014). The dataset has two types of annotations: Semantic Relatedness and Textual Entailment. We only use Semantic Relatedness annotations in our research. SICK was built based on two existing datasets: the 8K ImageFlickr dataset (Rashtchian et al., 2010)[3] and the SemEval-2012 STS MSR-Video Descriptions dataset (Agirre et al., 2012)[4]. The 8K ImageFlickr dataset is a dataset of images, where each image is associated with five descriptions. To derive SICK sentence pairs, the organisers randomly selected 750 images and sampled two descriptions from each of them. The SemEval2012 STS MSR-Video Descriptions data set is a collection of sentence pairs sampled from the short video snippets which compose the Microsoft Research Video Description Corpus[5]. A subset of 750 sentence pairs has been randomly chosen from this data set to be used in SICK.

   To generate SICK data from the 1,500 sentence pairs taken from the

[2]The SICK dataset is available to download at https://wiki.cimec.unitn.it/tiki-index.php?page=CLIC

[3]The 8K ImageFlickr data set is available at http://hockenmaier.cs.illinois.edu/8k-pictures.html

[4]The SemEval-2012 STS MSR-Video Descriptions dataset is available at https://www.cs.york.ac.uk/semeval-2012/task6/index.html

[5]The Microsoft Research Video Description Corpus is available to download at https://research.microsoft.com/en-us/downloads/38cf15fd-b8df-477e-a4e4-a4680caa75af/

source data sets, a 3-step process has been applied to each sentence pair, namely *(i) normalisation, (ii) expansion and (iii) pairing* (Marelli et al., 2014). The *normalisation* step has been carried out on the original sentences to exclude or simplify instances that contained lexical, syntactic or semantic phenomena such as named entities, dates, numbers, multiword expressions etc. In the *expansion* step, syntactic and lexical transformations with predictable effects have been applied to each normalised sentence, to obtain *(i)* a sentence with a similar meaning, *(ii)* a sentence with a logically contradictory or at least highly contrasting meaning, and *(iii)* a sentence that contains most of the same lexical items, but has a different meaning. Finally, in the *pairing* step, each normalised sentence in the pair has been combined with all the sentences resulting from the expansion phase and with the other normalised sentence in the pair. Furthermore, several pairs composed of completely unrelated sentences have been added to the data set by randomly taking two sentences from two different pairs (Marelli et al., 2014).

Each pair in the SICK dataset has been annotated to mark the degree to which the two sentence meanings are related (on a 5-point scale). The ratings have been collected through a large crowdsourcing study, where ten different annotators have evaluated each pair. Once all the annotations were collected, the relatedness gold score has been computed for each pair as the average of the ten ratings assigned by the annotators (Marelli et al., 2014). Table 1.1 shows examples of sentence pairs with different degrees

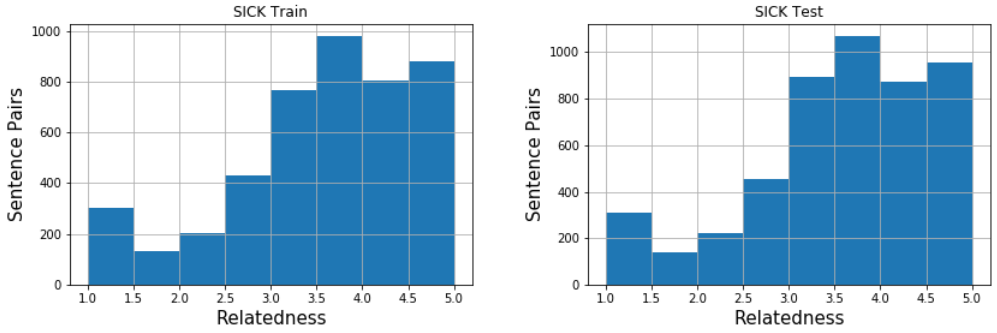
of semantic relatedness; gold relatedness scores are expressed on a 5-point rating scale. Given a test sentence pair, the machine learning models require to predict a value between 0-5, which reflects the relatedness of the given sentence pair.

Sentence Pair	Relatedness
1. A little girl is looking at a woman in costume. 2. A young girl is looking at a woman in costume.	4.7
1. Nobody is pouring ingredients into a pot. 2. Someone is pouring ingredients into a pot.	3.5
1. Someone is pouring ingredients into a pot. 2. A man is removing vegetables from a pot.	2.8
1. A man is jumping into an empty pool. 2. There is no biker jumping in the air.	1.6

Table 1.1: Example sentence pairs from the SICK dataset with their gold relatedness scores (on a 5-point rating scale). **Sentence Pair** column shows the two sentence and **Relatedness** column denotes the annotated relatedness score.

Figure 1.1 shows the distribution of the relatedness value in the SICK training and SICK testing set. It is clear that there are more sentence pairs with high relatedness values compared to low relatedness values. SICK train and SICK test follow a similar distribution.

The SICK dataset consists of pairs of sentences. We will refer to the first sentence in the pair as *sentence 1* and to the second sentence as *sentence 2*. In Figure 1.2 we visualise the normalised distribution of word count for both *sentence 1* and *sentence 2* in the SICK train and SICK test. Both sentences have a similar distribution reaching a maximum of around nine words. SICK train and SICK test follow a similar pattern in word count



(a) Relatedness distribution of SICK training test

(b) Relatedness distribution of SICK testing set

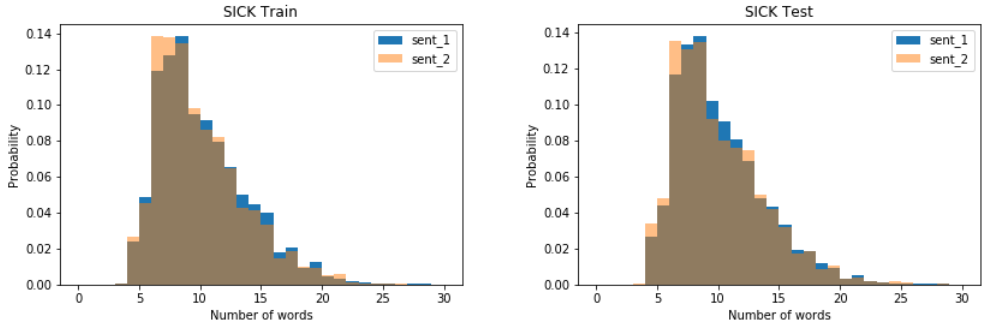
Figure 1.1: Relatedness distribution of SICK train and SICK test. *Sentence Pairs* shows the number of sentence pairs that a certain *Relatedness bin* has.

Measure	SICK Train		SICK Test	
	Sent_1	**Sent_2**	**Sent_1**	**Sent_2**
Word Count Mean	9.73	9.52	9.69	9.53
Word Count STD	3.66	3.70	3.69	3.65
Word Count MAX	28	32	28	30
Word Count MIN	3	3	3	3

Table 1.2: Word count stats in SICK training and SICK testing. *STD* indicates the standard deviation and the other acronyms indicate the common meaning

distribution too. Additionally we show some word count statistics in Table 1.2. In SICK train number of words for a sentence ranges from 3 to 32 and have the mean number of words around 9.5. These statistics are extremely close in the SICK test too.

The common judgement in STS is that when two sentences share a large number of words, the relatedness of that two sentences should be higher. In fact, in early feature-based approaches of calculating semantic textual similarity, the number of overlapping words between the two sentences was a common feature (Vilariño et al., 2014; Gupta et al., 2014a; Lynum et



(a) Normalised distribution of word count in SICK train

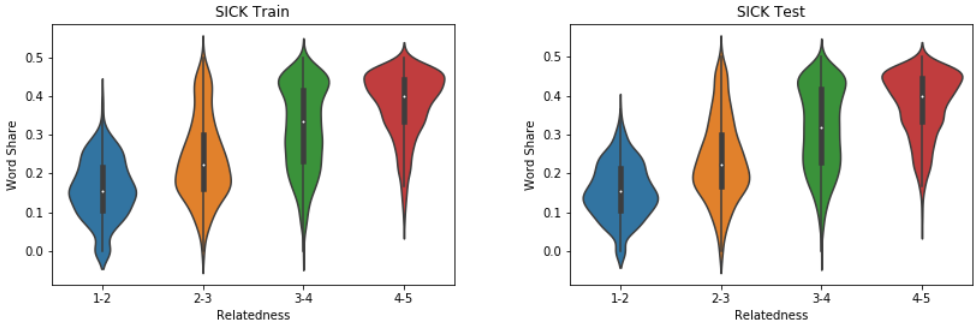
(b) Normalised distribution of word count in SICK test

Figure 1.2: Normalised distribution of word count in SICK train and SICK test. *Number of words* indicates the word count and *Probability* shows the total probability of a sentence with that word count appearing in the dataset.

al., 2014; Chávez et al., 2014). Systems like Vilariño et al. (2014) and Lynum et al. (2014) use the number of words common in two sentences as a feature directly, while systems like Gupta et al. (2014a) and Chávez et al. (2014) use Jaccard Similarity Coefficient as a feature, which is a measurement based on word overlap. To observe whether the number of words common in the two sentences has a relationship on the relatedness, we draw a violin plot[6] for each relatedness score bins with word share in Figure 1.3.

In figure 1.3, it is clear that sentence pairs with a higher relatedness tend to have a high word share. However, it should be noted that, in the "2-3" relatedness score bin, there are some sentence pairs with a high word share. The most common example for such a case would be *sentence 2* is the complete negation of the *sentence 1* (Marelli et al., 2014). In such cases,

[6]Violin plots are similar to box plots, except that they also show the probability density of the data at different values, usually smoothed by a kernel density estimator.



(a) Word share against relatedness bins in SICK train

(b) Word share against relatedness bins in SICK test

Figure 1.3: Word share against relatedness bins in SICK train and SICK test. *Word Share* indicates the ratio between number of common words in the two sentences to total number of words in the two sentences against each *Relatedness* bins

the two sentences share a large portion of the words, and one sentence has the *"not"* word that gives a completely opposite meaning compared to the other sentence. Similarly "4-5" relatedness score bin has some sentence pairs with a low word share. Those sentence pairs do not contain the same words but have synonyms or even paraphrases and possess the same overall meaning (Marelli et al., 2014). Therefore, the STS methods that focus on word share as a feature will not perform well in the SICK dataset (Ranasinghe et al., 2019a).

A clear strength in the SICK dataset is that the training set and the testing set reflect similar properties with regards to sentence length, relatedness distribution etc. Therefore, a properly trained machine learning model on the SICK train should give good results to the SICK test set as well (Marelli et al., 2014).

2. **STS 2017 English Dataset**[7] The second English STS dataset we used to experiment in this thesis is STS 2017 English Dataset, which was employed in *SemEval-2017 Task 1: Semantic Textual Similarity Multilingual and Cross-lingual Focused Evaluation* which is the most recent STS task in SemEval (Cer et al., 2017). As the training data for the competition, participants were encouraged to make use of all existing data sets from prior STS evaluations, including all previously released trial, training and evaluation data from SemEval 2012 - 2016 (Agirre et al., 2012; Agirre et al., 2013; Agirre et al., 2014; Agirre et al., 2015; Agirre et al., 2016). Once combined, we had 8277 sentence pairs for training. More information about the datasets used to build the training set is available in Table 1.3.

   On the other hand, a fresh test set of 250 sentence pairs was provided by SemEval-2017 STS Task organisers (Cer et al., 2017). The Stanford Natural Language Inference (SNLI) corpus (Bowman et al., 2015) was the primary data source for this test set. Similar to the SICK dataset, each pair in the STS 2017 English Test set has been annotated to mark the degree to which the two sentence meanings are related (on a 5-point scale). The ratings have been collected through crowdsourcing on Amazon Mechanical Turk[8]. Five annotations have been collected per pair, and the gold score has been computed for each pair as the average of the five ratings assigned by the

[7] The STS 2017 English Dataset is available to download at http://ixa2.si.ehu.es/stswiki/

[8] Amazon Mechanical Turk is a crowdsourcing website for businesses to hire remotely located *crowd workers* to perform discrete on-demand tasks. It is available at https://www.mturk.com/

Year	Dataset	Pairs	Source
2012 (Agirre et al., 2012)	MSRpar	1500	newswire
	MSRvid	1500	videos
	OnWN	750	glosses
	SMTnews	750	WMT eval.
	SMTeuroparl	750	WMT eval.
2013 (Agirre et al., 2013)	HDL	750	newswire
	FNWN	189	glosses
	OnWN	561	glosses
	SMT	750	MT eval.
2014 (Agirre et al., 2014)	HDL	750	newswire headlines
	OnWN	750	glosses
	Deft-forum	450	forum posts
	Deft-news	300	news summary
	Images	750	image descriptions
	Tweet-news	750	tweet-news pairs
2015 (Agirre et al., 2015)	HDL	750	newswire headlines
	Images	750	image descriptions
	Ans.-student	750	student answers
	Ans.-forum	375	Q&A forum answers
	Belief	375	committed belief
2016 (Agirre et al., 2016)	HDL	249	newswire headlines
	Plagiarism	230	short-answer plag.
	post-editing	244	MT postedits
	Ans.-Ans.	254	Q&A forum answers
	Quest.-Quest.	209	Q&A forum questions
2017 (Cer et al., 2017)	Trial	23	Mixed STS 2016

Table 1.3: Information about the datasets used to build the English STS 2017 training set. The **Year** column shows the year of the SemEval competition that the dataset got released. **Dataset** column expresses the acronym used describe a dataset in that year. **Pairs** is the number of sentence pairs in that particular dataset and **Source** shows the source of the sentence pairs.

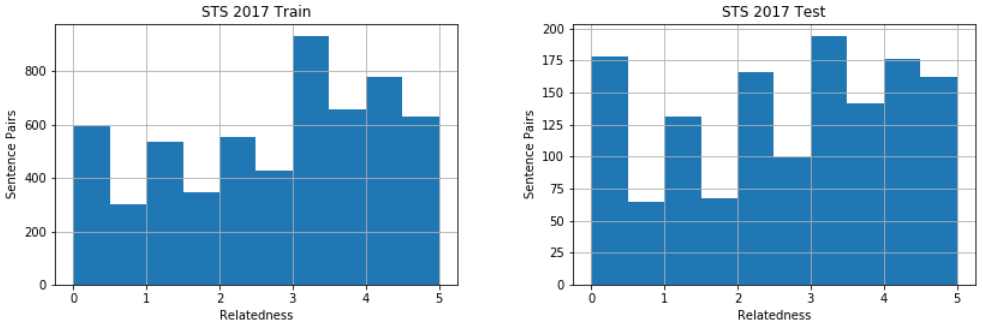
annotators. However, unlike the SICK dataset, the organisers have a clear explanation for the score ranges. Table 1.4 shows some example sentence pairs from the dataset with the gold labels and their explanations. Similar

to the SICK dataset, the machine learning models require predicting a value between 0-5 which reflects the similarity of the given sentence pair (Cer et al., 2017).

Sentence Pair	Relatedness
The two sentences are completely equivalent as they mean the same thing. 1. The bird is bathing in the sink. 2. Birdie is washing itself in the water basin.	5
The two sentences are completely equivalent as they mean the same thing. 1. The bird is bathing in the sink. 2. Birdie is washing itself in the water basin.	4
The two sentences are roughly equivalent, but some important information differs/missing. 1. John said he is considered a witness but not a suspect. 2. "He is not a suspect anymore." John said.	3
The two sentences are not equivalent, but share some details. 1. They flew out of the nest in groups. 2. They flew into the nest together.	2
The two sentences are not equivalent, but are on the same topic. 1. The woman is playing the violin. 2. The young lady enjoys listening to the guitar.	1
The two sentences are completely dissimilar 1. The black dog is running through the snow. 2. A race car driver is driving his car through the mud.	0

Table 1.4: Example sentence pairs from the STS2017 English dataset with their gold relatedness scores (on a 5-point rating scale) and explanations. **Sentence Pair** column shows the two sentence and **Relatedness** column denotes the annotated relatedness score.

Similar to the SICK dataset, we conduct an exploratory data analysis on



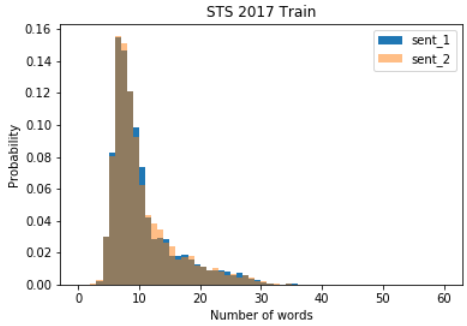
(a) Relatedness distribution of STS 2017 training test

(b) Relatedness distribution of STS 2017 testing set

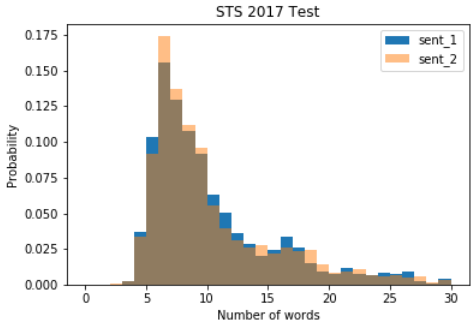
Figure 1.4: Relatedness distribution of STS 2017 train and STS 2017 test. *Sentence Pairs* shows the number of sentence pairs that a certain *Relatedness bin* has.

STS2017 dataset too. Figure 1.4 shows the relatedness distribution, and Figure 1.5 shows the normalised distribution of word count for *sentence 1* and *sentence 2* in STS 2017 train and test sets. Most of these statistics are similar to the SICK dataset. One notable change is the maximum word count in STS 2017 training dataset, which is 57 in *sentence 1* and 48 in *sentence 2* according to Table 1.5 while both SICK datasets' and STS 2017 test set's maximum word count is limited to 30. We believe that the reason is STS training dataset is composed with many sources including news articles that can have lengthy sentences. However, the STS algorithm should be able to properly handle this imbalance nature between the STS2017 train and test set (Cer et al., 2017).

In Figure 1.6, we draw a violin plot for each relatedness score bin with word share. We can see that generally, higher word share leads to higher relatedness, but still, there can be sentence pairs that contradict this, which



(a) Normalised distribution of word count in STS 2017 train



(b) Normalised distribution of word count in STS 2017 test

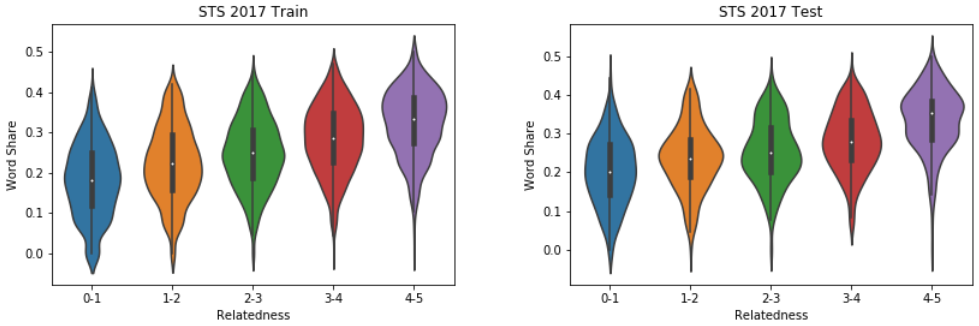
Figure 1.5: Normalised distribution of word count in STS 2017 train and STS 2017 test. *Number of words* indicates the word count and *Probability* shows the total probability of a sentence with that word count appearing in the dataset.

Measure	STS 2017 Train		STS 2017 Test	
	Sent_1	**Sent_2**	**Sent_1**	**Sent_2**
Word Count Mean	10.01	9.94	9.83	9.80
Word Count STD	5.52	5.36	5.14	5.14
Word Count MAX	57	48	30	30
Word Count MIN	3	2	3	2

Table 1.5: Word count stats in STS 2017 training and STS 2017 testing. *STD* indicates the standard deviation and the other acronyms indicate the common meaning

is similar to the observation we had with the SICK dataset.

Since the statistics of SICK and STS 2017 datasets are similar, one dataset can be used to augment the training data in the other dataset or perform transfer learning, which can lead to better results as neural networks perform stronger with more data (Wang et al., 2020c; Li et al., 2021). We hope to experiment this with supervised machine learning models in Chapters 4 and 5.



(a) Word share against relatedness bins in STS 2017 train

(b) Word share against relatedness bins in STS 2017 test

Figure 1.6: Word share against relatedness bins in STS 2017 train and STS 2017 test. *Word Share* indicates the ratio between number of common words in the two sentences to total number of words in the two sentences against each *Relatedness* bins

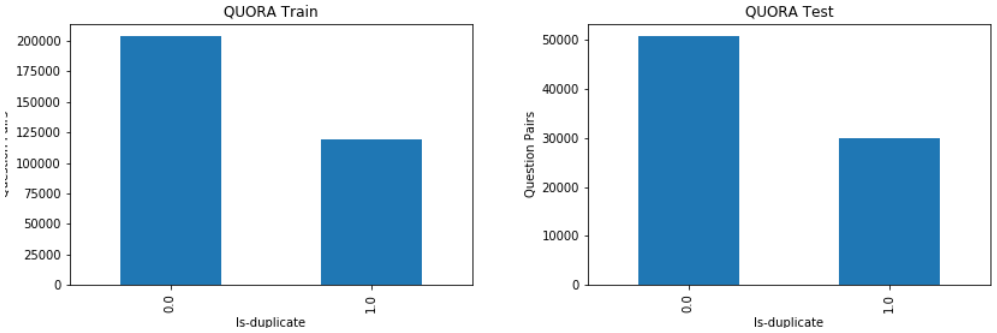
Question Pair	is-duplicate
1. What are natural numbers? 2. What is a least natural number?	0
1. Which Pizzas are most popularly ordered in Dominos menu? 2. How many calories does a Dominos Pizza have?	0
1. How do you start a bakery? 2. How can one start a bakery business?	1
1. Should I learn Python or Java first? 2. If I had to choose between learning Java and Python what should I choose to learn first?	1

Table 1.6: Example question pairs from the Quora Question Pairs dataset with their gold is-duplicate value. **Question Pair** column shows the two questions and **is-duplicated** column denotes whether it is a duplicated pair or not.

3. **Quora Question Pairs**[9] The Quora Question Pairs dataset is a big dataset that was first released for a Kaggle Competition[10]. Quora is a question-

[9]The Quora Question Pairs Dataset is available to download at http://qim.fs.quoracdn.net/quora_duplicate_questions.tsv

[10]Kaggle is an online community of data scientists and machine learning practitioners that hosts machine learning competitions. The Quora Question Pairs competition is available on https://www.kaggle.com/c/quora-question-pairs



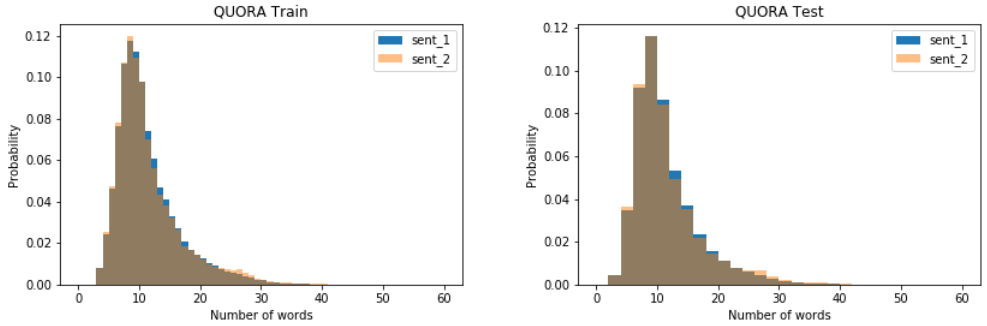
(a) Is-duplicate distribution of QUORA training test

(b) Is-duplicate distribution of QUORA testing set

Figure 1.7: Is-duplicate distribution of QUORA train and QUORA test. *Sentence Pairs* shows the number of sentence pairs that a certain *Is-duplicate* has.

and-answer website where internet users ask, answer, follow, and edit questions, either factually or in the form of opinions. If a particularly new question has been asked before, users merge the new question to the original question flagging it as a duplicate. The organisers used this functionality to create the dataset and did not use a separate annotation process. Their original sampling method has returned an imbalanced dataset with many more true examples of duplicate pairs than non-duplicates. Therefore, the organisers have supplemented the dataset with negative examples. One source of negative examples has been pairs of *related question* which belongs to similar topics but are not truly semantically equivalent.

The dataset has 400,000 question pairs, and we used 4:1 split on that to separate it into a training set and a testing set, resulting in 320,000 questions pairs in the training set and 80,000 sentence pairs in the testing



(a) Normalised distribution of word count in QUORA train

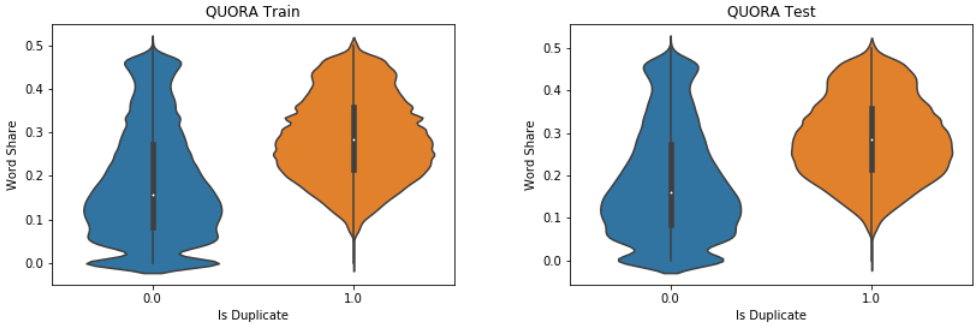
(b) Normalised distribution of word count in QUORA test

Figure 1.8: Normalised distribution of word count in QUORA train and QUORA test. *Number of words* indicates the word count and *Probability* shows the total probability of a sentence with that word count appearing in the dataset.

set. The machine learning models need to predict a continuous value between 0 and 1 that reflects whether it is a duplicate question pair or not. 1 indicates that a particular question pair is a duplicate, and 0 means it is not a duplicate.

This dataset is different from the previous datasets since it is not artificially created and uses day-to-day language. Since it has more than 300,000 training instances, deep learning systems will benefit more when used on this dataset.

In Figure 1.7 we show the distribution of the two classes in the QUORA dataset. The dataset seems to have more non-duplicate question pairs than duplicate sentence pairs, similar to the real-world scenario. According to the word count distribution in Figure 1.8 and word count statistics in Table 1.7, it is clear that QUORA datasets contain longer texts than SICK and STS 2017 datasets. Therefore, the QUORA dataset should be able to test



(a) Word share against is-relatedness value in QUORA train

(b) Word share against is-relatedness value in QUORA test

Figure 1.9: Word share against Is-duplicate values in QUORA train and QUORA test. *Word Share* indicates the ratio between number of common words in the two sentences to total number of words in the two sentences against each *Is-duplicate*

machine learning models' ability to handle lengthy texts properly.

In Figure 1.9 we show a violin plot for each *"is-duplicate"* value with word share. We can see that duplicate questions have a high word share. However, it should be noted that there are non-duplicate question pairs that still have a high word share. This shows that determining STS is not a trivial task.

According to statistics provided by the Director of Product Management at Quora on 17 September 2018, over 100 million people visit Quora every month, which raises the problem of different users asking similar questions with the same intent but in different words (Imtiaz et al., 2020). Multiple questions with the same intent can cause seekers to spend more time finding the best answer to their question and make writers feel they need to answer multiple versions of the same question. Therefore, identifying duplicate questions will make finding high-quality answers to questions

Measure	QUORA Train		QUORA Test	
	Ques_1	**Ques_2**	**Ques_1**	**Ques_2**
Word Count Mean	10.95	11.20	10.92	11.14
Word Count STD	5.44	6.31	5.40	6.31
Word Count MAX	125	237	73	237
Word Count MIN	1	1	1	1

Table 1.7: Word count stats in QUORA training and QUORA testing.*STD* indicates the standard deviation and the other acronyms indicate the common meaning

easier, resulting in an improved experience for Quora writers, seekers, and readers.

1.1.2 Datasets on Other Languages

One of the main requirements in our research was to build an STS method without depending on the language. Therefore throughout our study, we worked on several datasets from different languages. Those non-English datasets are described below.

1. **Arabic STS Dataset** [11] The Arabic STS dataset we selected was also used for the Arabic STS subtask in *SemEval 2017 Task 1: Semantic Textual Similarity Multilingual and Cross-lingual Focused Evaluation* (Cer et al., 2017). Unlike English, no data from previous SemEval competitions were available since this was the first time an Arabic STS task was organised in SemEval. More information about the extracted sentences will be shown in Table 1.9.

   A subset of the English STS 2017 dataset has been selected and human

[11]The Arabic STS dataset can be downloaded at http://alt.qcri.org/semeval2017/task1/index.php?id=data-and-tools

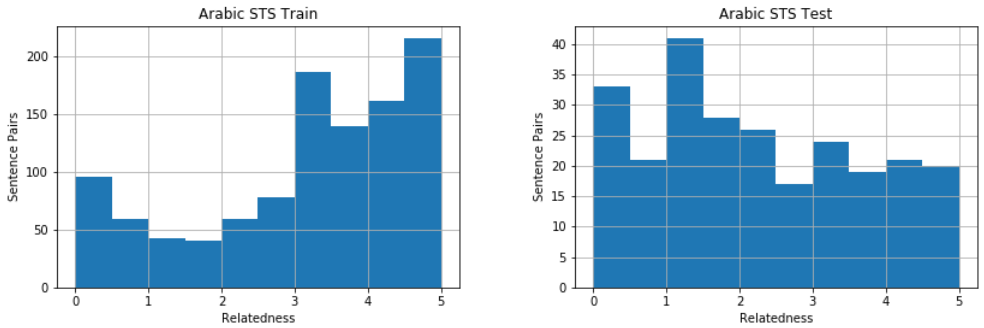
Sentence Pair	Similarity
1. أحدهم يقلي لحما. *Someone is frying meat.* 2. أحدهم يعزف البيانو. *Someone plays the piano.*	0.250
1. أمرأة تظيف المكونات في الإناء. *A woman cleaning ingredients in the bowl.* 2. إمرأة تكسر ثلاثة بيضات في الإناء. *A woman breaks three eggs in a bowl.*	1.750
1. طفلة تعزف القيثارة. *A Child is playing harp.* 2. . رجل يعزف القيثارة *A man plays the harp.*	2.250
1. المرأة تقطع البصل الأخضر. *The woman chops green onions.* 2. إمرأة تقشر بصلة. *A woman peeling an onion.*	3.250
1. الأيل قفز فوق السياج. *The deer jumped over the fence.* 2. أيل يقفز فوق سياج الإعصار. *Deer Jumps Over Hurricane Fence*	4.800

Table 1.8: Example question pairs from the Arabic STS dataset. **Sentence Pair** column shows the two sentences. We also included their translations in the table. The translations were done by a native Arabic speaker. **Similarity** column indicates the annotated similarity of the two sentences.

translated into Arabic to prepare the annotated instances. Sentences have been translated independently from their pairs. Arabic translations have been provided by native Arabic speakers with strong English skills at Carnegie Mellon University in Qatar. Translators have been given an English sentence and its Arabic machine translations where they have

Dataset	Pairs	Source
Trial	23	Mixed STS 2016
MSRpar	510	newswire
MSRvid	368	videos
SMTeuroparl	203	WMT eval.

Table 1.9: Information about the datasets used to build the Arabic STS training set. **Dataset** column expresses the acronym used describe the dataset. **Pairs** is the number of sentence pairs in that particular dataset and **Source** shows the source of the sentence pairs.



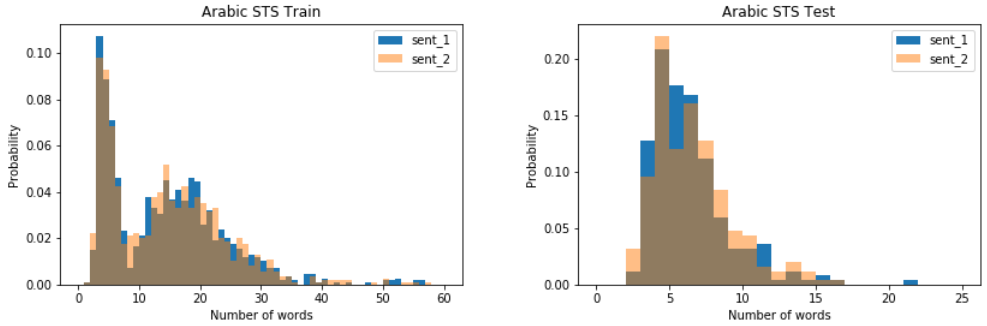
(a) Relatedness distribution of Arabic STS training test

(b) Relatedness distribution of Arabic STS testing set

Figure 1.10: Relatedness distribution of Arabic STS train and Arabic STS test. *Sentence Pairs* shows the number of sentence pairs that a certain *Relatedness bin* has.

performed post-editing to correct errors. STS labels have been then transferred to the translated pairs. Therefore, annotation guidelines and the template are similar to the English STS 2017 dataset. 1103 sentence pairs were available for training, and 250 sentence pairs were available in the test set. Table 1.8 shows few pairs of sentences with their similarity scores. The machine learning models require predicting a value between 0-5, which reflects the similarity of a given Arabic sentence pair.

Similar to the English STS datasets, we also analysed the Arabic STS dataset



(a) Normalised distribution of word count in Arabic STS train

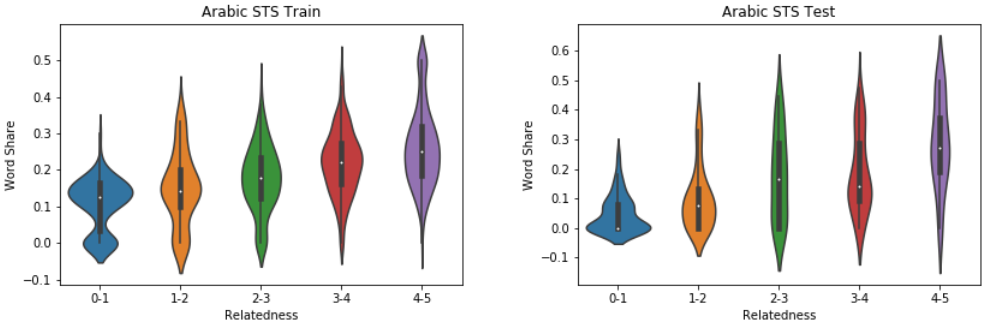
(b) Normalised distribution of word count in Arabic STS test

Figure 1.11: Normalised distribution of word count in Arabic STS train and Arabic STS test. *Number of words* indicates the word count and *Probability* shows the total probability of a sentence with that word count appearing in the dataset.

considering the same set of properties. As can be seen in Figure 1.10, the relatedness distribution is different in the training and test sets. In the training set, there are many sentences with high relatedness scores compared to low relatedness scores. On the other hand, there are many sentences with low relatedness scores compared to the high relatedness scores in the test set.

Word count distribution in the training and test sets of the Arabic dataset is different too. As shown in Figure 1.11 the sentences in the training set are longer than the sentences in the test set. This is further confirmed by the stats in Table 1.10. The average word count in the training set is 31, while this is 9 in the test set. With these observations, we can conclude that the Arabic training set and test are different with regard to several properties. This nature of the dataset can be a challenge for ML systems.

In Figure 1.12, we draw a violin plot for each relatedness bin with



(a) Word share against relatedness bins in Arabic STS train

(b) Word share against relatedness bins in Arabic STS test

Figure 1.12: Word share against relatedness bins in Arabic STS train and Spanish STS test. *Word Share* indicates the ratio between number of common words in the two sentences to total number of words in the two sentences against each *Relatedness* bins

Measure	Arabic STS Train		Arabic STS Test	
	Sent_1	**Sent_2**	**Sent_1**	**Sent_2**
Word Count Mean	31.23	31.02	9.03	9.34
Word Count STD	12.15	12.37	3.66	3.74
Word Count MAX	90	90	22	24
Word Count MIN	5	1	3	3

Table 1.10: Word count stats in Arabic STS training and Arabic STS testing.*STD* indicates the standard deviation and the other acronyms indicate the common meaning

word share. The higher word share generally leads to higher similarity. However, there are sentence pairs that contradict this theory. This observation is similar to the English datasets.

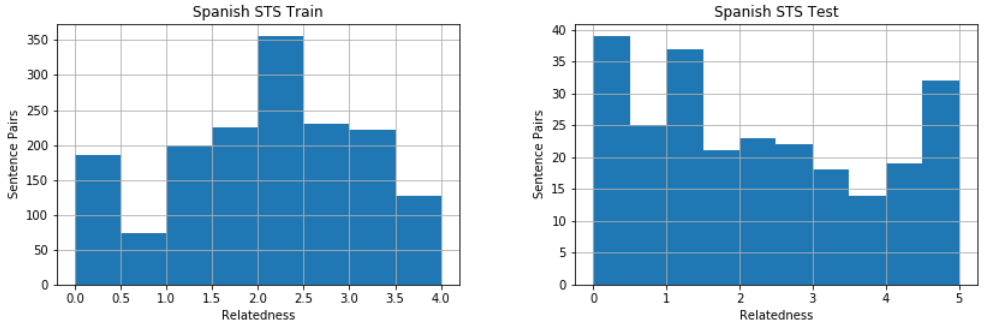
2. **Spanish STS Dataset**[12] - Spanish STS dataset that we used was employed for Spanish STS subtask in *SemEval 2017 Task 1: Semantic Textual Similarity Multilingual and Cross-lingual Focused Evaluation* (Cer et al., 2017). The training set has 1250 sentence pairs annotated with a relatedness score

[12]The Spanish STS dataset can be downloaded at http://alt.qcri.org/semeval2017/task1/index.php?id=data-and-tools

Sentence Pair	Similarity
1. Amás, los misioneros apunten que los númberos d'infectaos puen ser shasta dos o hasta cuatro veces más grandess que los oficiales. *(Furthermore, missionaries point out that the numbers of infected can be up to two or up to four times larger than the official ones.)* 2. Los cadáveres de personas fallecidas pueden ser hasta diez veces más contagiosos que los infectados vivos. *(The corpses of deceased people can be up to ten times more contagious than those infected alive.)*	0.6
1. La policía abatió a un caníbal cuando devoraba a una mujer Matthew Williams, de 34 años, fue sorprendido en la madrugada mordiendo el rostro de una joven a la que había invitado a su hotel. *(Police killed a cannibal while devouring a woman Matthew Williams, 34, was caught early in the morning biting the face of a young woman he had invited to his hotel.)* 2. La policía de Gales del Sur mató a un caníbal cuando se estaba comiendo la cara de una mujer de 22 años en la habitación de un hotel. *(South Wales police killed a cannibal when he was eating the face of a 22-year-old woman in a hotel room.)*	2
1. Ollanta Humala se reúne mañana con el Papa Francisco. *(Ollanta Humala meets tomorrow with Pope Francis.)* 2. El Papa Francisco mantuvo hoy una audiencia privada con el presidente Ollanta Humala, en el Vaticano. *(Pope Francis held a private audience today with President Ollanta Humala, at the Vatican.)*	3

Table 1.11: Example sentence pairs from the Spanish STS dataset. **Sentence Pair** column shows the two sentences. We also included their translations in the table. The translations were done by a native Spanish speaker. **Similarity** column indicates the annotated similarity of the two sentences.

between 0 and 4. The training set combined several datasets from previous SemEval STS shared tasks (Cer et al., 2017). Table 1.12 shows more information about the training set. There were two sources for the test set



(a) Relatedness distribution of Spanish STS training test

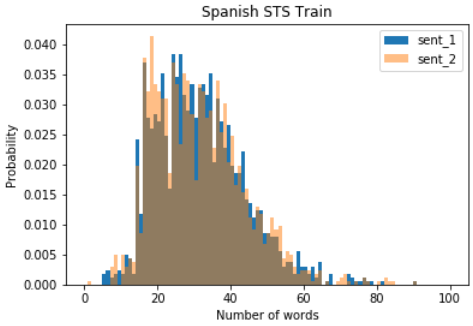
(b) Relatedness distribution of Spanish STS testing set

Figure 1.13: Relatedness distribution of Spanish STS train and Spanish STS test. *Sentence Pairs* shows the number of sentence pairs that a certain *Relatedness bin* has.

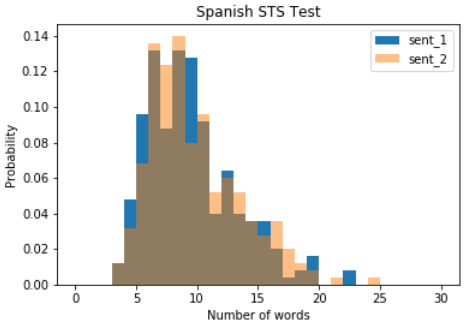
- Spanish news and Spanish Wikipedia dump having 500 and 250 sentence pairs respectively (Cer et al., 2017). Both datasets were annotated with a relatedness score between 0 and 5. Table 1.11 shows few pairs of sentences with their similarity score. The machine learning models require to predict a value between 0-5, which reflects the similarity of the given Spanish sentence pair.

Year	Dataset	Pairs	Source
2014 (Agirre et al., 2014)	Trial	56	NR
	Wiki	324	Spanish Wikipedia
	News	480	Newswire
2015 (Agirre et al., 2015)	Wiki	251	Spanish Wikipedia
	News	500	Newswire

Table 1.12: Information about the datasets used to build the Spanish STS training set. The **Year** column shows the year of the SemEval competition that the dataset got released. **Dataset** column expresses the acronym used describe a dataset in that year. **Pairs** is the number of sentence pairs in that particular dataset and **Source** shows the source of the sentence pairs.



(a) Normalised distribution of word count in Spanish STS train

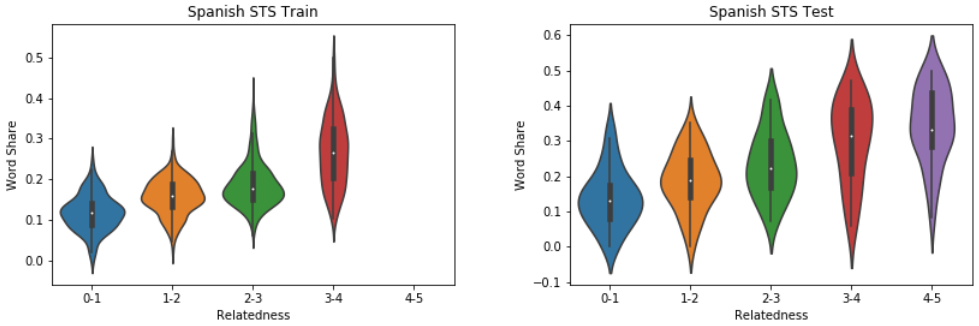


(b) Normalised distribution of word count in Spanish STS test

Figure 1.14: Normalised distribution of word count in Spanish STS train and Spanish STS test. *Number of words* indicates the word count and *Probability* shows the total probability of a sentence with that word count appearing in the dataset.

Similar to the Arabic STS dataset, we also analysed the Spanish STS dataset considering the same set of properties. A key challenge in the Spanish STS dataset too is that the test set is very different from the training set. As can be seen in Figure 1.13 training set has been annotated with relatedness scores 0-4 while the test set has been annotated with relatedness scores 0-5. Therefore, STS methods need to be developed in such a way that they can handle this situation. This can be observed as a weakness in this dataset, but at the same time, this property of the dataset can be exploited to measure the robustness of an STS system as well.

Furthermore, as shown in Figure 1.14 and Table 1.13 sentence pairs in the test set are shorter in word length than the sentence pairs in the train set. Therefore, STS methods working on this dataset should be able to properly handle that too. This is similar to what we observed with Arabic STS dataset.



(a) Word share against relatedness bins in Spanish STS train

(b) Word share against relatedness bins in Spanish STS test

Figure 1.15: Word share against relatedness bins in Spanish STS train and Spanish STS test. *Word Share* indicates the ratio between number of common words in the two sentences to total number of words in the two sentences against each *Relatedness* bins

Measure	Spanish STS Train		Spanish STS Test	
	Sent_1	**Sent_2**	**Sent_1**	**Sent_2**
Word Count Mean	31.23	31.02	9.03	9.34
Word Count STD	12.15	12.37	3.66	3.74
Word Count MAX	90	90	22	24
Word Count MIN	5	1	3	3

Table 1.13: Word count stats in Spanish STS training and Spanish STS testing.*STD* indicates the standard deviation and the other acronyms indicate the common meaning

The violin plot between the word share against the relatedness bin in Spanish STS is similar to the previous datasets we analysed. As can be seen in Figure 1.15, higher word share leads to a higher similarity, but some sentence pairs contradict this.

1.1.3 Datasets on Different Domains

To experiment with how our STS methods can be adopted into different domains, we also used a dataset from a different discipline which we introduce in this section.

1. **Bio-medical STS Dataset: BIOSSES**[13] - BIOSSES is the first and only benchmark dataset for biomedical sentence similarity estimation (Soğancıoğlu et al., 2017). The dataset comprises of 100 sentence pairs, in which each sentence has been selected from the TAC (Text Analysis Conference) Biomedical Summarisation Track - training dataset containing articles from the biomedical domain [14]. The sentence pairs have been evaluated by five different human experts that judged the similarity and gave scores ranging from 0 (no relation) to 4 (equivalent). The score range was described based on the guidelines of SemEval 2012 Task 6 on STS (Agirre et al., 2012). Besides the annotation instructions, example sentences from the bio-medical literature have also been provided to the annotators for each similarity degree. To represent the similarity between two sentences, we took the average of the scores provided by the five human experts. Table 1.14 shows few examples in the dataset. The machine learning models require to predict a value between 0-4, which reflects the similarity of the given biomedical sentence pair.

   The relatedness distribution is shown in Figure 1.16. It is similar to the relatedness distribution we saw in SICK and STS2017, where there were more sentences with high relatedness scores than low relatedness scores.

   As shown in Figure 1.16, sentences in the BIOSSES dataset are longer than

[13] Bio-medical STS Dataset: BIOSSES can be downloaded from https://tabilab.cmpe.boun.edu.tr/BIOSSES/DataSet.html

[14] Biomedical Summarisation Track is a shared task organised in TAC 2014 - https://tac.nist.gov/2014/BiomedSumm/

Sentence Pair	Similarity
1. It has recently been shown that Craf is essential for Kras G12D-induced NSCLC. 2. It has recently become evident that Craf is essential for the onset of Kras-driven non-small cell lung cancer.	4
1. Up-regulation of miR-24 has been observed in a number of cancers, including OSCC. 2. In addition, miR-24 is one of the most abundant miRNAs in cervical cancer cells, and is reportedly up-regulated in solid stomach cancers.	3
1. These cells (herein termed TLM-HMECs) are immortal but do not proliferate in the absence of extracellular matrix (ECM) 2. HMECs expressing hTERT and SV40 LT (TLM-HMECs) were cultured in mammary epithelial growth medium (MEGM, Lonza)	1.4
1.The up-regulation of miR-146a was also detected in cervical cancer tissues. 2. Similarly to PLK1, Aurora-A activity is required for the enrichment or localisation of multiple centrosomal factors which have roles in maturation, including LATS2 and CDK5RAP2/Cnn.	0.2

Table 1.14: Example question pairs from the BIOSSES dataset. **Sentence Pair** column shows the two sentences. **Similarity** column indicates the averaged annotated similarity of the two sentences.

the sentences in the English datasets we mentioned before. The average length of a sentence in English datasets was below 15, while in the BIOSSES dataset, the average length is around 20.

As we mentioned before, the BIOSSES dataset only has 100 sentence pairs. A dataset as small as this one can not be used to train a supervised ML method, requiring alternative approaches such as unsupervised methods and transfer learning techniques which we will be exploring in the

Figure 1.16: Relatedness distribution of BIOSSES. *Sentence Pairs* shows the number of sentence pairs that a certain *Relatedness bin* has.

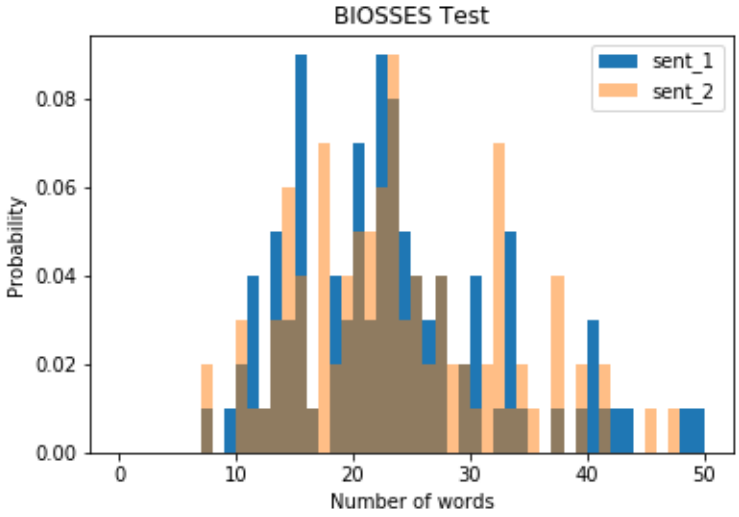


Figure 1.17: Normalised distribution of word count in BIOSSES. *Number of words* indicates the word count and *Probability* shows the total probability of a sentence with that word count appearing in the dataset.

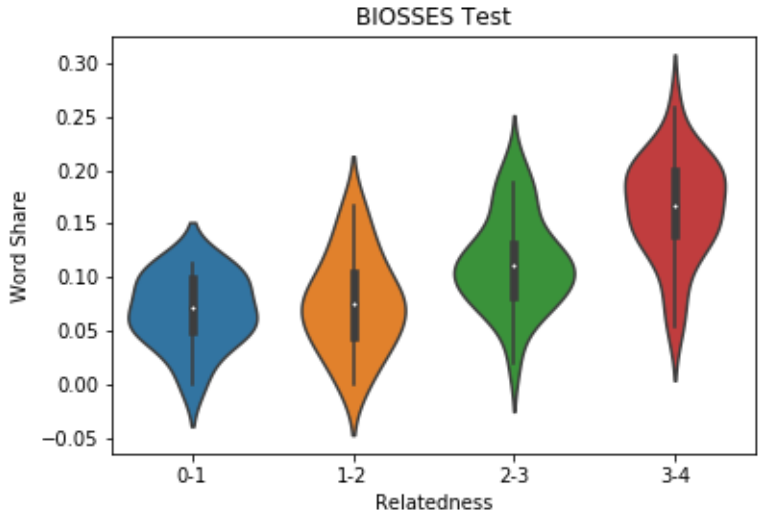


Figure 1.18: Word share against relatedness bins in BIOSSES. *Word Share* indicates the ratio between number of common words in the two sentences to total number of words in the two sentences against each *Relatedness* bins

following few chapters.

1.2 Evaluation Metrics

While training a machine learning model is a crucial step, how the model generalises on unseen data is an equally important aspect that should be considered in every machine learning model. We need to know whether it actually works and, consequently, if we can trust its predictions. This is typically called as *evaluation*. All of the datasets that we introduced in the previous section has what we call a *test* set. The machine learning models need to provide their predictions for the test set, and the predictions will be evaluated against the gold values of the test set.

There are three common evaluation metrics that are employed in Semantic

Textual Similarity tasks, which we explain in this section. We will be using them to evaluate our models throughout the first part of our research.

In the equations presented for each evaluation metrics, we represent the gold labels with X and predictions with Y. Therefore, a gold label in the i^{th} position will be represented by X_i and the prediction in i^{th} position will be represented by Y_i.

1. **Pearson's Correlation Coefficient** - Correlation is a technique for investigating the relationship between two quantitative, continuous variables. Pearson's correlation coefficient (ρ) measures the strength of the linear association between the two variables. A value of +1 is the total positive linear correlation between the variables, 0 is no linear correlation, and -1 is the total negative linear correlation.

   Pearson's Correlation Coefficient is one of the most common evaluation metrics in STS shared tasks (Marelli et al., 2014; Agirre et al., 2012; Agirre et al., 2013; Agirre et al., 2014; Agirre et al., 2015; Agirre et al., 2016). A machine learning model with a Pearson's Correlation Coefficient close to 1 indicates that the predictions of that model and gold labels have a strong positive linear correlation. Therefore, it is a good model to predict STS. Pearson's Correlation Coefficient equation is shown in Equation 1.1 where cov is the covariance, σ_X is the standard deviation of X, and σ_Y is the

standard deviation of Y.

$$\rho = \frac{\text{cov}(X, Y)}{\sigma_X \sigma_Y} \tag{1.1}$$

2. **Spearman's Correlation Coefficient** - Spearman's Correlation Coefficient (τ) is another common evaluation metric in STS shared tasks (Marelli et al., 2014; Agirre et al., 2012; Agirre et al., 2013; Agirre et al., 2014; Agirre et al., 2015; Agirre et al., 2016). It assesses how well the relationship between two variables can be described using a monotonic function. A monotonic relationship is a relationship that does one of the following:

   (a) as the value of one variable increases, so does the value of the other variable, *OR*,

   (b) as the value of one variable increases, the other variable value decreases.

   However, a monotonic relationship does not require a constant rate, whereas in a linear relationship, the rate of increase/decrease is constant. The fundamental difference between Pearson's correlation coefficient and Spearman's correlation coefficient is that the Pearson's correlation coefficient only works with a linear relationship between the two variables, whereas the Spearman's correlation coefficient works with the monotonic relationships as well. Spearman's correlation coefficient is shown in

Equation 1.2 where D_i is the pairwise distances of the ranks of the variables X_i and Y_i and n is the number of elements in X or Y.

$$\tau = 1 - \frac{6 \sum D_i^2}{n(n^2 - 1)} \tag{1.2}$$

In Spearman's correlation coefficient, a value of +1 is the total positive correlation between the variables, 0 is no correlation, and -1 is the total negative correlation. Therefore, similar to the Pearson's correlation coefficient, a machine learning model with a Spearman's Correlation Coefficient close to 1 indicates that the predictions of that model and gold labels have a strong positive correlation, and it is an excellent model to predict STS.

3. **Root Mean Squared Error** - Both Pearson's Correlation Coefficient and Spearman's Correlation Coefficient works only when both gold labels(X) and predictions (Y) are continuous. Therefore, for the datasets like Quora Question Pairs, where the gold labels are discrete values, Root Mean Squared Error (RMSE) is preferred for evaluation than Correlation Coefficient values. RMSE measures the distance between the gold labels and the predictions. RMSE equation is shown in Equation 1.3 where n is the number of elements in X or Y.

$$RMSE = \sqrt{(\frac{1}{n}) \sum_{i=1}^{n} (Y_i - X_i)^2} \tag{1.3}$$

In RMSE, a value close to 0 means that the error between the predictions and the gold labels are minimal. Therefore, a machine learning model with an RMSE value close to 1 indicates fewer errors and is an excellent model to predict STS.

1.3 Conclusion

Calculating the STS is an important research area in NLP, which plays a vital role in many applications such as question answering, document summarisation, information retrieval and information extraction. Most of the early approaches were based on traditional machine learning and involved heavy feature engineering. However, these approaches are difficult to be adopted in different languages and do not provide competitive results anymore. With the advances of word embeddings, and due to the success neural networks have achieved in other fields, most of the methods proposed in recent years rely on word vectors. These methods can be further categorised into supervised and unsupervised methods. Analysing STS methods belong to both of these categories would be beneficial to the community. Furthermore, exploring the ability of these methods to perform in a multilingual setting and a multi-domain setting would be a timely contribution to the NLP field.

The introduction of competitive STS shared tasks led to the development of standard datasets. We selected three recently released English STS datasets; SICK, STS2017 and Quora Question Pairs. They carry different characteristics. We exploratory analysed these datasets focussing on common properties like

the size of the dataset, sentence length, common number of words etc. Furthermore, we identified specific properties of these datasets that would limit the performance of traditional STS methods like edit distance. For the multilingual experiments, we selected a Spanish and an Arabic dataset. Similar to the English STS datasets, we exploratory analysed them for certain characteristics. For the multi-domain experiments, we chose a Biomedical STS Dataset. This dataset brings a key challenge to the STS methods as it does not have a separate training set. Therefore, this dataset would provide the opportunity to evaluate various STS methods in an out-of-domain and in an unsupervised setting.

The STS shared tasks has further contributed to the development of evaluation measures in STS. In all the datasets except Quora Question Pairs, Pearson Correlation and Spearman Correlation has been used to evaluate STS methods. In the Quora dataset, Root Mean Squared Error has been used to assess the methods. We followed the same evaluation measures in order to compare our methods with other systems submitted to the competition.

In the next few chapters, we will be exploring different unsupervised and supervised STS methods. We will be evaluating them in English STS datasets, non-English STS datasets as well as out of domain STS datasets to investigate their adaptability in different environments.

Chapter 2

Vector Aggregation based STS Methods

One of the key factors that contributed to the success of neural architectures in NLP is the fact that they are trained on large datasets. The biggest challenge that neural-based architectures face when applied to STS tasks is the small size of the datasets available to train them. As a result, in many cases, the networks cannot be trained properly. Given the huge amount of human labour required to produce STS datasets, it is not feasible to have high-quality large training datasets. As a result, researchers working in the field have considered unsupervised methods for STS. Recent unsupervised approaches use pre-trained word/sentence embeddings directly for the similarity task, without training a neural network model on them. Such approaches have used cosine similarity on sent2vec (Pagliardini et al., 2018), InferSent (Conneau et al., 2017), Word Mover's Distance (Le and Mikolov, 2014), Doc2Vec (Le and Mikolov, 2014) and Smooth Inverse Frequency with GloVe vectors (Arora et al., 2017). While these approaches have produced decent results in the final rankings of shared tasks, they also act as strong baselines for the STS task.

This chapter explores the performance of three unsupervised STS methods - cosine similarity using average vectors (Mitchell and Lapata, 2008), Word

Mover's Distance (Kusner et al., 2015) and cosine similarity using Smooth Inverse Frequency (Arora et al., 2017), and how to improve these methods using contextual word embeddings which will be explained more in Section 2.1.

We address four research questions in this chapter:

RQ1: Can contextual word embedding models like BERT be used to improve unsupervised STS methods?

RQ2: How well such an unsupervised method performs compared to other popular supervised/ unsupervised STS methods?

RQ3: Can the proposed unsupervised STS method be easily adapted into different languages?

RQ4: How well the proposed unsupervised STS method performs in a different domain?

The main contributions of this chapter are as follows.

1. The Related Work Section (Section 2.1) covers three unsupervised STS techniques to compute semantic similarity at sentence level.

2. We propose an improved unsupervised STS method based on contextual word embeddings and evaluate it on three English STS datasets, two non-English STS datasets and a bio-medical STS dataset which were introduced in Chapter 1.

3. The code used for the experiments conducted is publicly available to the community[1].

[1]The public GitHub repository is available on https://github.com/tharindudr/simple-sentence-similarity

The rest of this chapter is organised as follows. Section 2.1 describes the three unsupervised STS methods we experimented with in this section. Section 2.2 presents the methodology, the contextual word embeddings we used, followed by the results from comparing the English datasets with the baselines. Sections 2.3 and 2.4 show how our method can be applied to different languages and domains in addition to their results. The chapter finishes with conclusions and ideas for future research directions in unsupervised STS methods.

2.1 Related Work

Given that a good STS metric is required for a variety of natural language processing fields, researchers have proposed a large number of such metrics. Before the shift of interest to neural networks, the majority of the proposed methods relied heavily on feature engineering. With the introduction of word embedding models, researchers focused more on neural representation for this task.

As we mentioned before, there are two main approaches that employ neural representation models: unsupervised and supervised. Unsupervised methods use pre-trained word/sentence embeddings directly for the similarity task without training a neural network model on them. In contrast, supervised approaches use a machine learning model trained to predict the similarity using word embeddings (Ranasinghe et al., 2019a). Since this chapter focuses on unsupervised STS methods, this section contains previous research on unsupervised STS methods.

The three unsupervised STS methods explored in this chapter: Cosine similarity on average vectors (Mitchell and Lapata, 2008), Word Mover's Distance (Kusner et al., 2015) and Cosine similarity using Smooth Inverse Frequency (Arora et al., 2017), are the most common unsupervised methods explored in STS tasks. Apart from these methods, cosine similarity of the output from Infersent (Conneau et al., 2017), sent2vec (Pagliardini et al., 2018) and doc2vec (Le and Mikolov, 2014) have also been used to represent the similarity between two sentences which we discuss in the next chapter.

2.1.1 Cosine Similarity on Average Vectors

The first unsupervised STS method that we considered to estimate the semantic similarity between a pair of sentences takes the average of the word embeddings of all words in the two sentences and calculates the cosine similarity between the resulting embeddings (Mitchell and Lapata, 2008). This method is a common way to acquire sentence embeddings from word embeddings (Orăsan, 2018). Obviously, this simple baseline leaves considerable room for variation. Researchers have investigated the effects of ignoring stopwords and computing an average weighted by tf-idf in particular (Mitchell and Lapata, 2010).

2.1.2 Word Mover's Distance

The second state-of-the-art STS method that we have considered is Word Mover's Distance introduced by Kusner et al. (2015). Word Mover's Distance uses the word embeddings of the words in two texts to measure the minimum distance that the words in one text need to "travel" in semantic space to reach the words in

the other text as shown in Figure 2.1. Kusner et al. (2015) show that this is a better approach than vector averaging since this technique keeps the word vectors as it is, throughout the operation.

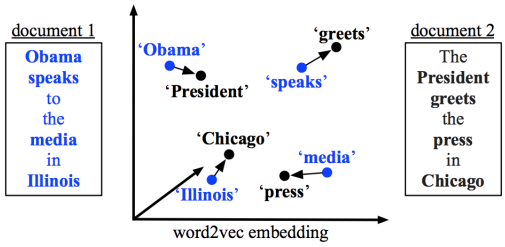


Figure 2.1: The Word Mover's Distance between two sentences (Kusner et al., 2015)

2.1.3 Cosine Similarity Using Smooth Inverse Frequency

The third and the last unsupervised STS method we considered is to acquire sentence embeddings using Smooth Inverse Frequency proposed by Arora et al. (2017) and then calculate the cosine similarity between those sentence embeddings. Semantically speaking, taking the average of the word embeddings in a sentence tends to give too much weight to words that are fairly irrelevant. Smooth Inverse Frequency tries to solve this problem in two steps.

1. **Weighting**: Smooth Inverse Frequency takes the weighted average of the word embeddings in the sentence. Every word embedding is weighted by $\frac{a}{a+p(w)}$, where a is a parameter that is typically set to 0.001 and $p(w)$ is the estimated frequency of the word in a reference corpus.

2. **Common component removal**: After weighting, Smooth Inverse

Frequency computes the principal component of the resulting embeddings for a set of sentences. It then subtracts their projections on the *first principal component* from these sentence embeddings. This step should remove variations related to frequency and syntax that are less relevant semantically.

As a result, Smooth Inverse Frequency downgrades non-content bearing words such as *but, just*, etc., and keeps the information that contributes most to the semantics of the sentence (Arora et al., 2017). After acquiring the sentence embeddings for a pair of sentences, the cosine similarity between those two vectors was taken to represent their similarity.

All of these STS methods are based on word embeddings/vectors. The main weakness of word vectors is that each word has the same unique vector regardless of the context it appears. Consider the word "bank", which has several meanings. Still, in standard word embeddings such as GloVe (Pennington et al., 2014), fastText (Mikolov et al., 2018) or Word2Vec (Mikolov et al., 2013b) each instance of the word has the same representation regardless of the meaning which is used. For example, the word ‘bank’ in two sentences - “I am walking by the river bank” and “I deposited money to the bank” would have the same embeddings, which can be confusing for machine learning models. The recent introduction of contextualised word representations such as BERT (Devlin et al., 2019) and XLNet (Yang et al., 2019b) solved this problem by providing vectors for words taking their context into consideration. In this way, the word ’bank’ in the above sentences would have two different embeddings. Contextual

word embedding models have improved the results of many natural language processing tasks over traditional word embedding models (Peters et al., 2018; Devlin et al., 2019). However, they have not been applied to unsupervised vector aggregation-based STS methods to the best of our knowledge.

Therefore, we explore how contextualised word representations can improve the above mentioned unsupervised STS methods. We will explain the neural network architectures of these contextual word embeddings in Chapter 5. For this chapter, we considered these architectures as a black box where we feed the words to get the embeddings. We considered these contextualised word representations in terms of their popularity by the time we were doing the experiments.

1. **ELMo**[2] introduced by Peters et al. (2018), uses bidirectional language model (biLM) to learn both the word (e.g., syntax and semantics) and linguistic context. After pre-training, an internal state of vectors can be transferred to downstream natural language processing tasks. ELMo vectors have been successfully used in many natural language processing tasks such as text classification (Jiang et al., 2019) and named entity recognition (Luo et al., 2018), which motivated us to explore ELMo in unsupervised STS methods. Also, we were aware of the fact that ELMo has been pre-trained on different languages (Che et al., 2018) and different domains (Jin et al., 2019) which will be useful for when we are adapting our methodology for different languages and domains in Sections 2.3 and

[2]More details about ELMo can be viewed on https://allennlp.org/elmo

2.4.

2. **BERT**[3] Introduced by Devlin et al. (2019), BERT is probably the most popular contextualised word embedding model. In contrast to ELMo which uses a shallow concatenation layer, BERT employs a deep concatenation layer. As a result, BERT is considered a very powerful embedding architecture. BERT has been successfully applied in many natural language processing tasks such as text classification (Ranasinghe et al., 2019c), word similarity (Hettiarachchi and Ranasinghe, 2020), named entity recognition (Liang et al., 2020) and question and answering (Yang et al., 2019a). Similarly to ELMo, BERT has been widely adapted for different languages[4] such as Arabic (Antoun et al., 2020), French (Martin et al., 2020), Spanish (Cañete et al., 2020), Greek (Koutsikakis et al., 2020) etc. and different domains such as SciBERT (Beltagy et al., 2019), BioBERT (Lee et al., 2019), LEGAL-BERT (Chalkidis et al., 2020) etc.

3. **Flair**[5] is another popular contextualised word embedding model introduced by Akbik et al. (2018). It takes a different approach, using a character-level language model rather than the word level language model used in ELMo and BERT. Flair has also been used successfully in natural language processing tasks such as named entity recognition (Akbik et al.,

[3]The GitHub repository of BERT is available on https://github.com/google-research/bert

[4]Information about pre-trained BERT models for different languages can be found on https://bertlang.unibocconi.it/

[5]The GitHub repository of Flair is available on https://github.com/flairNLP/flair

2019b), part-of-speech tagging (Akbik et al., 2018) and has been widely adapted for different languages and domains (Akbik et al., 2018; Sharma and Daniel Jr, 2019).

Apart from using these contextual word embedding models individually, we also considered **Stacked Embeddings** of these models. Stacked Embeddings are obtained by concatenating different embeddings. According to Akbik et al. (2018), stacking the embeddings can provide powerful embeddings to represent words. Therefore, we experimented with several combinations of Stacked Embeddings.

Even though these contextual word embedding models have shown promising results in many natural language processing tasks, to the best of our knowledge, none of these contextual word representations have been applied to unsupervised vector aggregation-based STS methods.

2.2 Improving State of the Art STS Methods

As mentioned before, we applied different contextual word embeddings on three unsupervised STS methods and their variants. First, we experimented with English STS datasets that we explained in Section 1.1. Our implementation was based on the *Flair-NLP* Framework (Akbik et al., 2019a) which makes it easier to switch between different word embedding models when acquiring word embeddings. Additionally, *Flair-NLP* has its own *model zoo* of pre-trained models to allow researchers to use state-of-the-art NLP models in their applications. For English, all of these contextualised word embedding models come with different

variants such as *small, large etc.* Usually, the larger models provide a better accuracy since they have been trained on a larger dataset than the smaller models. However, this comes with the disadvantage that these larger models are more resource-intensive than the smaller models. To achieve better accuracy, we used the largest model available in each contextual word embedding model. We will describe these models in the following paragraphs.

For ELMo, we used the 'original (5.5B)' pre-trained model provided by Peters et al. (2018) which was trained on a dataset of 5.5B tokens consisting of Wikipedia (1.9B) and all of the monolingual news crawl data from WMT[6] 2008-2012 (3.6B). Peters et al. (2018) mention that ELMo original (5.5B) performs slightly better than other ELMo models and recommend it as the default model. Using this model, we represented each word as a vector with a size of 3072 dimensions.

For BERT, we used the 'bert-large-cased' pre-trained model. Compared to the 'bert-base-cased' model, this model provided slightly better results in all the NLP tasks experimented in Devlin et al. (2019). We represented each word as a 4096 lengthened vector using this model.

As suggested in Akbik et al. (2018), the recommended way to use Flair embeddings is to stack pre-trained 'news-forward' flair embeddings and pre-trained flair 'news-backward' embeddings with GloVe (Pennington et al., 2014) word embeddings. We used the stacked model to represent each word as a 4196 lengthened vector.

[6]WMT: Workshop on Statistical Machine Translation is a leading conference in NLP that is being organised annually.

As mentioned before, we also considered stacked embeddings of ELMo and BERT. For this, we used the pre-trained 'bert-large-uncased' model and 'original (5.5B)' pre-trained ELMo model to represent each word as a 4096 + 3072 vector.

To compare the results of the contextualised word embeddings, we used a standard word representation model as a baseline in each experiment. In this research, we used Word2vec embeddings (Mikolov et al., 2013a) pre-trained on Google news corpus[7]. We represented each word as a 300 lengthened vector using this model.

In the following list, we show the performance of each unsupervised STS method with contextual word embeddings on the different English STS datasets.

1. **Cosine Similarity on Average Vectors** - The first unsupervised STS method we tried to improve using contextual word embeddings is Cosine Similarity on Average Vectors, as explained in Section 2.1. Table 2.1 shows the results for the SICK dataset, Table 2.2 shows the results for the STS 2017 dataset and Table 2.3 shows the results for the Quora Question Pairs dataset. To compare our results with other systems, we conducted the experiments only on the test data of the three above mentioned datasets. Since this method leaves considerable room for variation, we have investigated the following variations and reported their results in each table.

   (a) All the word vectors were considered for averaging. Results are

[7]Pretrained Word2vec can be downloaded from https://code.google.com/archive/p/word2vec/

shown in column I of Tables 2.1, 2.2 and 2.3

(b) All the word vectors except the vectors for stop words were considered for averaging. Column II of Tables 2.1, 2.2 and 2.3 shows the results.

(c) All the word vectors were weighted from their tf-idf scores and considered averaging. Results are shown in column III of Tables 2.1, 2.2 and 2.3

(d) Stop words were removed first and the remaining word vectors were weighted from their tf-idf scores and considered averaging. Column IV of Tables 2.1, 2.2 and 2.3 shows the results.

	I		**II**		**III**		**IV**	
Model	ρ	τ	ρ	τ	ρ	τ	ρ	τ
Word2vec	**0.730**†	0.624	**0.714**	0.583	**0.693**	0.570	**0.687**	0.555
ELMo	0.669	0.592	0.693	0.603	0.676	**0.579**	0.668	**0.572**
Flair	0.646	0.568	0.670	0.562	0.644	0.535	0.643	0.531
BERT	0.683	0.633	0.686	0.606	0.557	0.552	0.539	0.538
ELMo $\bigoplus$ *BERT*	0.696	**0.634**†	0.702	**0.614**	0.607	0.562	0.591	0.551

Table 2.1: Results for SICK dataset with Vector Averaging. Columns **I**, **II**, **III** and **IV** indicate the different variations as explained above. For each word embedding model, Pearson Correlation (ρ) and Spearman Correlation (τ) are reported for all variations between the predicted values and the gold labels of the test set. $\bigoplus$ indicates a stacked word embedding model. The best result in each variation is highlighted in **Bold**. The best result from all of the variations is marked with †.

From the results in Tables 2.1, 2.2 and 2.3 there is no clear indication that contextualised word embeddings perform better than the standard word embeddings. In all of the datasets considered, the best result was provided

	I		II		III		IV	
Model	ρ	τ	ρ	τ	ρ	τ	ρ	τ
Word2vec	**0.625**	0.583	0.609	**0.635**†	**0.640**†	**0.591**	**0.588**	**0.573**
ELMo	0.575	0.574	**0.618**	0.609	0.374	0.395	0.352	0.376
Flair	0.411	0.444	0.584	0.586	0.325	0.374	0.336	0.386
BERT	0.575	0.574	0.555	0.588	0.355	0.401	0.309	0.386
ELMo $\bigoplus$ *BERT*	0.600	**0.597**	0.591	0.608	0.391	0.413	0.354	0.398

Table 2.2: Results for STS 2017 dataset with Vector Averaging. Columns **I**, **II**, **III** and **IV** indicate the different variations as explained above. For each word embedding model, Pearson Correlation (ρ) and Spearman Correlation (τ) are reported for all variations between the predicted values and the gold labels of the test set. $\bigoplus$ indicates a stacked word embedding model. The best result in each variation is highlighted in **Bold**. The best result from all of the variations is marked with †.

	I	II	III	IV
Model	RMSE	RMSE	RMSE	RMSE
Word2vec	**0.621**	**0.591**†	**0.646**	**0.607**
ELMo	0.629	0.615	0.652	0.649
Flair	0.720	0.711	0.743	0.735
BERT	0.651	0.643	0.673	0.662
ELMo $\bigoplus$ *BERT*	0.625	0.611	0.650	0.647

Table 2.3: Results for QUORA dataset with Vector Averaging. Columns **I**, **II**, **III** and **IV** indicate the different variations as explained above. For each word embedding model, Root Mean Squared Error (RMSE) is reported for all variations. $\bigoplus$ indicates a stacked word embedding model. The best result in each variation is highlighted in **Bold**. The best result from all of the variations is marked with †.

by Word2vec.

All the contextualised word embedding models we considered have more than 3000 dimensions for the word representation, which is higher than the number of dimensions for the word representation we had for standard embeddings - 300. As the vector averaging model is highly dependent on the number of dimensions that a vector can have, the curse of dimensionality might be the reason for the poor performance of

contextualised word embeddings in vector averaging variants (Ranasinghe et al., 2019a).

2. **Word Mover's Distance** - The second unsupervised STS method we experimented with is Word Mover's Distance, as explained in Section 2.1. Similarly to the average vectors, we compared having contextualised word embeddings in place of traditional word embeddings in Word Mover's Distance. Table 2.4 shows the results for the SICK dataset. Table 2.5 shows the results for the STS 2017 dataset and Table 2.6 shows the results for the Quora Questions Pairs dataset. We have investigated the effects of considering/ ignoring stop words before calculating the Word Mover's Distance as detailed below.

   (a) Considering all the words to calculate the Word Mover's Distance. Results are shown in column I of Tables 2.4, 2.5 and 2.6

   (b) Removing stop words before calculating the Word Mover's Distance. Column II of Tables 2.4, 2.5 and 2.6 shows the results.

   As depicted in the Tables 2.4, 2.5 and 2.6, contextualised word representations could not improve Word Mover's method over standard word representations. Even though, ELMo $\bigoplus$ BERT model outperforms Word2vec in the SICK dataset with regards to Spearman Correlation (τ), there is no clear indication that contextual word representations would outperform standard word representations in Word Mover's method. Since the travelling distance is dependent on the number of dimensions, the

	I		II	
Model	ρ	τ	ρ	τ
Word2vec	**0.730**†	0.624	**0.714**	0.583
ELMo	0.669	0.592	0.693	0.603
Flair	0.646	0.568	0.670	0.562
BERT	0.683	0.633	0.686	0.606
ELMo $\bigoplus$ *BERT*	0.696	**0.634**†	0.702	**0.614**

Table 2.4: Results for SICK dataset with Word Mover's Distance. Columns **I** and **II** indicate the different variations as explained above. For each word embedding model, Pearson Correlation (ρ) and Spearman Correlation (τ) are reported on all variations between the predicted values and the gold labels of the test set. $\bigoplus$ indicates a stacked word embedding model. The best result in each variation is highlighted in **Bold**. The best result from all of the variations is marked with †.

	I		II	
Model	ρ	τ	ρ	τ
Word2vec	**0.625**†	0.583	0.609	**0.635**†
ELMo	0.575	0.574	**0.618**	0.609
Flair	0.411	0.444	0.584	0.586
BERT	0.575	0.574	0.555	0.588
ELMo $\bigoplus$ *BERT*	0.600	**0.597**	0.591	0.608

Table 2.5: Results for STS 2017 dataset with Word Mover's Distance. Columns **I** and **II** indicate the different variations as explained above. For each word embedding model, Pearson Correlation (ρ) and Spearman Correlation (τ) are reported on all variations between the predicted values and the gold labels of the test set. $\bigoplus$ indicates a stacked word embedding model. The best result in each variation is highlighted in **Bold**. The best result from all of the variations is marked with †.

curse of dimensionality might be the reason for the poor performance of contextualised word representations in this scenario too.

3. **Smooth Inverse Frequency** As the third and final unsupervised STS method, we experimented with Smooth Inverse Frequency as explained in Section 2.1. Similarly to the previous STS methods, we compared having contextualised word embeddings in place of traditional word embeddings

	I	II
Model	RMSE	RMSE
Word2vec	**0.621**	**0.591**†
ELMo	0.629	0.615
Flair	0.720	0.711
BERT	0.651	0.643
ELMo $\bigoplus$ *BERT*	0.625	0.611

Table 2.6: Results for QUORA dataset with Word Mover's Distance. Columns **I** and **II** indicate the different variations as explained above. For each word embedding model, Root Mean Squared Error (RMSE) is reported on all variations. $\bigoplus$ indicates a stacked word embedding model. The best result in each variation is highlighted in **Bold**. The best result from all of the variations is marked with †.

with the Smooth Inverse Frequency method. Since the Smooth Inverse Frequency method takes care of stop words, we did not consider any variations that we experimented with previous STS methods. Table 2.7 shows the results for the SICK dataset. Table 2.8 shows the results for the STS 2017 dataset and Table 2.9 shows the results for the Quora Questions Pairs dataset.

Model	ρ	τ
Word2vec	0.734	0.632
ELMo	0.740	0.654
Flair	0.731	0.634
BERT	0.746	0.661
ELMo $\bigoplus$ *BERT*	0.753†	0.669†

Table 2.7: Results for SICK dataset with Smooth Inverse Frequency. For each word embedding model, Pearson Correlation (ρ) and Spearman Correlation (τ) are reported between the predicted values and the gold labels of the test set. $\bigoplus$ indicates a stacked word embedding model. The best result from all of the variations is marked with †.

As can be seen in the results, unlike previous unsupervised STS methods, contextualised word embeddings improved the Smooth Inverse Frequency

Model	ρ	τ
Word2vec	0.638	0.601
ELMo	0.641	0.609
Flair	0.639	0.606
BERT	0.650	0.612
ELMo $\bigoplus$ *BERT*	$0.654^{\dagger}$	$0.616^{\dagger}$

Table 2.8: Results for STS 2017 dataset with Smooth Inverse Frequency. For each word embedding model, Pearson Correlation (ρ) and Spearman Correlation (τ) are reported between the predicted values and the gold labels of the test set. $\bigoplus$ indicates a stacked word embedding model. The best result from all of the variations is marked with †.

Model	RMSE
Word2vec	0.599
ELMo	0.585
Flair	0.589
BERT	0.572
ELMo $\bigoplus$ *BERT*	$0.566^{\dagger}$

Table 2.9: Results for QUORA dataset with Smooth Inverse Frequency. For each word embedding model, Root Mean Squared Error (RMSE) is reported. $\bigoplus$ indicates a stacked word embedding model. The best result from all of the variations is marked with †.

method results when compared to the standard word embeddings in all three datasets considered. It can be observed that the Smooth Inverse Frequency method is less sensitive to the number of dimensions in the word embedding model as it has a common component removal step. Due to this reason, contextualised word embedding models do not suffer the *Curse of dimensionality* (Indyk and Motwani, 1998) with Smooth Inverse Frequency. In all of the datasets, the stacked embedding model of ELMo and BERT (ELMo $\bigoplus$ BERT) performed best. Furthermore, from all the unsupervised STS methods we experimented with including Vector Averaging and Word Movers Distance, ELMo $\bigoplus$ BERT with the

Smooth Inverse Frequency method provided the best results. With these observations, we address our **RQ1**, contextualised embeddings can be used to improve the unsupervised STS methods. Even though the contextual word embedding models did not improve the results in Vector Averaging and Word Mover's Distance, there was clear improvement when they were applied with Smooth Inverse Frequency.

With regards to our **RQ2**: *How well does the proposed unsupervised STS method perform when compared to various other STS methods?*, we compared our best results from the SICK dataset to the results from the *SemEval 2014 Task 1* (Marelli et al., 2014). This was the original task that experimented with the SICK dataset, as mentioned previously. Our unsupervised method had 0.753 Pearson correlation score, whilst the best result in the competition had 0.828 Pearson correlation (Marelli et al., 2014). Our approach would be ranked in the ninth position from the top results out of 18 participants, and it is the best unsupervised STS method among the results (Marelli et al., 2014). Our method even outperformed systems that rely on additional feature generation (e.g. dependency parses) or data augmentation schemes. For example, our method is just above the UoW system, which relied on 20 linguistics features fed into a Support Vector Machine and that obtained a 0.714 Pearson correlation (Gupta et al., 2014a). Compared to these complex approaches, our simple unsupervised approach provides a strong baseline to STS tasks. These observations answer our **RQ2**, that the proposed

unsupervised STS method is competitive with the other supervised and unsupervised STS methods.

2.3 Portability to Other Languages

Our **RQ3** targets the multilingual aspect of the proposed approach; *How well the proposed unsupervised STS method performs in different languages?*. To answer this, we evaluated our method in Arabic STS and Spanish STS datasets that were introduced in Chapter 1. Our approach has the advantage of not relying on language-dependent features and not needing a training set as the approach is unsupervised. As a result, the approach is easily portable to other languages, given the availability of ELMo and BERT models in that particular language.

As the contextual word embedding models, for ELMo embeddings, we used the Arabic and Spanish Elmo models released by Che et al. (2018). Che et al. (2018) have trained ELMo models for 44 languages, including Arabic and Spanish, using the same hyperparameter settings as Peters et al. (2018) on Common Crawl and a Wikipedia dump of each language[8]. The models are hosted in NLPL Vectors Repository (Fares et al., 2017)[9]. As for BERT, we used the "BERT-Base, Multilingual Cased" model (Devlin et al., 2019) which has been built on the top 100 languages with the largest Wikipedias. This also includes Arabic and Spanish languages. Similarly to the English experiments, we conducted the experiments through the *Flair-NLP* Framework (Akbik et al., 2019a). To compare the results,

[8]The GitHub repository for the ELMo for many languages project is available on https://github.com/HIT-SCIR/ELMoForManyLangs

[9]More information on the NLPL Vectors Repository is available on http://wiki.nlpl.eu/index.php/Vectors/home

as traditional word embeddings, we used AraVec (Soliman et al., 2017) [10] for Arabic and Spanish 3B words Word2Vec Embeddings (Bilbao-Jayo and Almeida, 2018)[11] for Spanish.

Similarly to the English datasets, from the unsupervised STS methods we considered, Smooth Inverse Frequency with ELMo and BERT stacked embeddings gave the best results for both Arabic and Spanish datasets. For Arabic our approach had 0.624 Pearson correlation whilst the best result (Wu et al., 2017) in the competition had 0.754 Pearson correlation (Cer et al., 2017). Our approach would rank eighteenth out of 49 teams in the final results. As with English datasets, our approach has the best result for an unsupervised method and surpasses other complex supervised models. For example, Kohail et al. (2017) proposed a supervised approach, combining dependency graph similarity and coverage features with lexical similarity measures using regression methods and scored only 0.610 Pearson correlation. This show that the proposed unsupervised STS method can outperform supervised STS methods too.

For Spanish, our approach had 0.712 Pearson correlation whilst the best result (Tian et al., 2017) in the competition had 0.855 Pearson correlation (Cer et al., 2017). Our approach would rank sixteenth out of 46 teams in the final results, which is the best result for an unsupervised approach. As with the English model, this one also surpasses other complex supervised models. For example,

[10] AraVec has been trained on Arabic Wikipedia articles. The models are available on https://github.com/bakrianoo/aravec

[11] Spanish 3B words Word2Vec Embeddings have been trained on Spanish news articles, Wikipedia articles and Spanish Boletín Oficial del Estado (BOE; English: Official State Gazette). The model is available on https://github.com/aitoralmeida/spanish_word2vec

Barrow and Peskov (2017) used a supervised machine learning algorithm with word embeddings and scored only 0.516 Pearson correlation. Our fairly simple unsupervised approach outperforms this supervised method by a large margin.

These findings answer our **RQ3**; the proposed unsupervised STS method can be successfully applied to other languages, and it is very competitive even with the supervised methods.

2.4 Portability to Other Domains

To answer our *RQ4*; how well the proposed unsupervised STS method can be applied in different domains, we evaluated our method on Bio-medical STS dataset as explained in 1. As we mentioned Bio-medical STS dataset does not have a training set. Therefore, only the unsupervised approaches can be applied to this dataset which provides an ideal opportunity for the STS method we introduced in this chapter.

For the experiments, as for the contextual word embedding models, we used BioELMo (Jin et al., 2019)[12], BioBERT (Lee et al., 2019)[13] and BioFLAIR (Sharma and Daniel Jr, 2019)[14]. Additionally, to compare the performance with standard word embeddings, we used BioWordVec (Zhang et al., 2019c)[15]. As with the English and multilingual experiments, Smooth Inverse Frequency with

[12]BioELMo is the biomedical version of ELMo, pre-trained on PubMed abstracts. The model is available on https://github.com/Andy-jqa/bioelmo

[13]BioBERT has trained BERT on PubMed abstracts. The model is available on https://github.com/dmis-lab/biobert

[14]BioFLAIR is FLAIR embeddings trained on PubMed abstracts. The model is available on https://github.com/shreyashub/BioFLAIR

[15]BioWordVec has trained word2vec on a combination of PubMed and PMC texts. The model is available on https://bio.nlplab.org/

ELMo and BERT stacked embeddings performed best with this dataset. It had 0.708 Pearson correlation, whilst the best performing method had 0.836 Pearson correlation. This would rank our approach seventh out of 22 teams in the final results of the task (Soğancıoğlu et al., 2017).

It should also be noted that it outperforms many complex methods that sometimes use external tools too. As an example, the UBSM-Path approach is based on ontology-based similarity, which uses METAMAP (Aronson, 2001) for extracting medical concepts from the text, and our simple unsupervised approach outperforms them by a large margin. UBSM-Path only has a 0.651 Pearson correlation, and compared to that, our simple STS method based on contextual embeddings outperforms them.

This answers our fourth and the final *RQ*; the proposed unsupervised STS method can be successfully applied to other domains, and it is very competitive with the available STS methods.

2.5 Conclusions

This chapter experimented with three unsupervised STS methods: cosine similarity using average vectors, Word Mover's Distance and cosine similarity using Smooth Inverse Frequency, with contextualised word embeddings to calculate semantic similarity between pairs of texts and compared them with other unsupervised/ supervised approaches. Contextualised word embeddings could not improve cosine similarity using average vectors and Word Mover's Distance methods, but the results when using the Smooth Inverse Frequency

method were improved with contextualised word embeddings instead of standard word embeddings. Furthermore we report that stacking ELMo and BERT provides a stronger word representation than the individual representations of ELMo and BERT. The results indicated that calculating cosine similarity using Smooth Inverse Frequency with stacked embeddings of ELMo and BERT is the best unsupervised method from the available approaches. Also, the performance of our approach was ranked in the top half of the final results list in the SICK dataset, surpassing many complex and supervised approaches.

Our approach was also applied in Arabic, Spanish and Bio-medical STS tasks. In all the cases, our simple unsupervised method finished in the top half of the final results list outperforming many supervised/ unsupervised STS methods. Therefore, given our results, we can safely assume that regardless of the language or the domain, cosine similarity, using Smooth Inverse Frequency with stacked embeddings of ELMo and BERT will provide a simple but strong and unsupervised method for STS tasks.

Contextual word embedding models are rapidly increasingly in popularity due to their superior performance compared to standard word embedding models. Contextual word embedding models are available even in low resource languages like Assamese (Kakwani et al., 2020), Hebrew (Chriqui and Yahav, 2021), Odia (Kakwani et al., 2020), Yoruba (Alabi et al., 2020), Twi (Alabi et al., 2020) etc. Contextual word embedding models will soon be available in all languages where standard word embedding models are available. Therefore, we can conclude that the unsupervised STS method we introduced in this chapter

will be beneficial to many languages and domains.

As future work, the experiments can be extended in to other BERT like contextual word embedding models such as XLNet (Yang et al., 2019b), RoBERTa (Liu et al., 2019), SpanBERT (Joshi et al., 2020) etc. One drawback of using contextual word embedding models is that the majority of the pre-trained models only support maximum number of 512 tokens which would be problematic when encoding longer sequences. Therefore, STS with longer sequences could be explored with recently released contextual word embedding models like Longformer (Beltagy et al., 2020) and Big Bird (Zaheer et al., 2021) that support encoding sequences longer than the 512 maximum number of tokens. Benefiting from the fact that this method is unsupervised and does not need a training dataset, it could be further be expanded into many languages and domains as future work.

Chapter 3

STS with Neural Sentence Encoders

The main goal of a sentence encoder is to map a variable-length text to a fixed-length vector representation. In basic terms, a sentence encoder takes a sentence or text as an input and outputs a vector. This vector encodes the meaning of the sentence and can be used for downstream tasks such as text classification, text similarity etc. In these downstream tasks, the sentence encoder is often considered a black box, where the users employ it to produce sentence embeddings without knowing exactly what happens in the encoder itself.

Ideally, the approaches we experimented with in Chapter 2 such as Vector Averaging (Mitchell and Lapata, 2008) and Smooth Inverse Frequency (Arora et al., 2017) can be considered as sentence encoders given that in thsse approaches, the input is a variable-length text and the output is a fixed-length vector. However, these approaches have major drawbacks when representing sentences. One such drawback is these approaches do not care about word order. Consider the following two sentences: *"The food is good, but the service is bad"* and *"The food is bad, but the service is good"*. The sentences would have the same embeddings using these approaches even though the meaning of these two sentences is

completely different. Another drawback is these approaches lose information during the vector aggregation process. Consider the following two sentences *"It is great"* and *"It is not great"*, Vector Averaging and Smooth Inverse Frequency will give similar sentence embeddings as there is only a one word difference in the two sentences. Even though this effect can be minimised using techniques like TF/IDF weighting, as we explored in Chapter 2, a different approach would be to train end-to-end models to get sentence embeddings. These models are commonly known as *sentence encoders* in the NLP community.

Over the years, various sentence encoders like Sent2vec (Pagliardini et al., 2018), Infersent (Cer et al., 2018) and Universal Sentence Encoder (Conneau et al., 2017) have been proposed. Even though the majority of these sentence encoders have sophisticated architectures, many are only using them as a black box to get the sentence embeddings. Therefore, the sentence encoders have been prevalent in the NLP community. As the sentence encoders provide good quality sentence embeddings efficiently, word embedding aggregation methods such as Vector Averaging (Mitchell and Lapata, 2008) and Smooth Inverse Frequency (Mitchell and Lapata, 2008) have often been overlooked by the NLP community in favour of sentence encoders (Logeswaran and Lee, 2018).

Once the sentence embeddings are obtained from a sentence encoder, using them to calculate STS is an easy task (Pagliardini et al., 2018). Since these sentence embeddings are already semantically powerful, a simple vector comparison technique such as cosine similarity between the embeddings can be used to calculate the STS of the two sentences (Conneau and Kiela, 2018).

Therefore, a pre-trained sentence encoder can be used as an unsupervised STS method (Ranasinghe et al., 2019a). Even though sentence encoders have been commonly used in STS tasks, there has not been a comprehensive study done on them. Since most researchers use sentence encoders as a black box in many applications, they do not understand the limitations of using them. This chapter addresses this gap by exploring sentence encoders in different STS datasets adapting them for various languages and domains. This study seeks to identify the limitations of sentence encoders and to provide clarity when not to use them.

We address three research questions in this chapter:

RQ1: How well do the sentence encoders perform in English STS datasets?

RQ2: Can the sentence encoders be easily adapted in different languages?

RQ3: How well do these sentence encoders perform in different domains?

The main contributions of this chapter are as follows.

1. In the Related Work Section (Section 3.1), we discuss three sentence encoders that are popular in the NLP community.

2. We evaluate these three sentence encoders on three English STS datasets, two non-English STS datasets and a bio-medical STS dataset which were introduced in Chapter 1.

3. The code used to conduct the experiments is publicly available to the community[1].

[1]The public GitHub repository is available on https://github.com/tharindudr/simple-sentence-similarity

The rest of this chapter is organised as follows. Section 3.1 describes the three sentence encoders we experimented with in this section. In Section 3.2 we present the experiments we conducted with the three sentence encoders in English STS datasets followed by the comparative results from the other unsupervised STS methods. Sections 3.3 and 3.4 show how sentence encoders can be applied to different languages and domains and the results from this. The chapter finishes with conclusions and ideas for future research directions in sentence encoders.

3.1 Related Work

As aforementioned, sentence encoders are popular with the NLP community. The three sentence encoders explored in this chapter; Sent2vec (Pagliardini et al., 2018), Infersent (Conneau et al., 2017) and Universal Sentence Encoder (Cer et al., 2018), are the most popular sentence encoders. There is also Doc2vec (Le and Mikolov, 2014) which can be considered as a sentence encoder. However, due to the various upgrades in different python libraries, the official pre-trained Doc2vec models are no longer working. Therefore, we only used the following sentence encoders in our experiments.

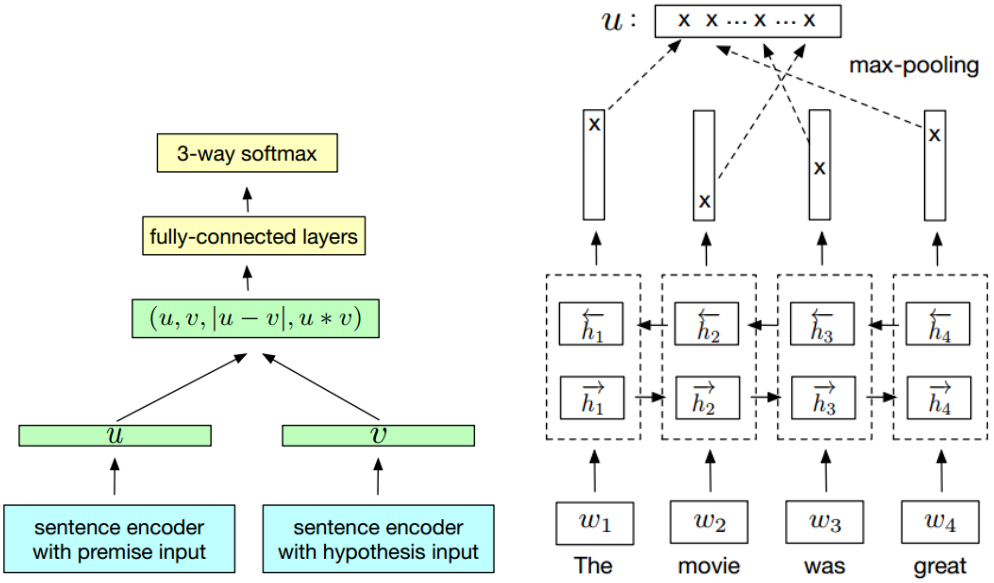
Sent2vec Sent2vec presents a simple but efficient unsupervised objective to train distributed representations of sentences (Pagliardini et al., 2018). It can be considered as an extension of Word2vec (CBOW) to sentences. The key differences between CBOW and Sent2Vec are the removal of the input

subsampling, considering the entire sentence as context, and the addition of word n-grams. With Sent2vec, sentence embedding is defined as the average of the word embeddings of its constituent words. The objective of Sent2vec is similar to CBOW; predict the missing word given the context (Pagliardini et al., 2018).

Sent2vec has officially released several pre-trained models to derive the sentence embeddings[2]. Because the approach is unsupervised and the objective function is simple, Sent2vec has also been adapted in different languages and domains (Heo et al., 2021; Zhu et al., 2018; Allot et al., 2019).

Infersent InferSent is an NLP technique developed by Facebook for universal sentence representation, which uses supervised training to produce high-quality sentence vectors (Conneau et al., 2017). The authors explore seven different architectures for sentence encoding, including LSTM (Hochreiter and Schmidhuber, 1997), GRU (Chung et al., 2014), Concatenation of last hidden states of forward and backward GRU, Bi-directional LSTM (Schuster and Paliwal, 1997) with mean/max pooling, Self-attentive network and Hierarchical Deep Convolutional Neural Network (Conneau et al., 2017). All of these models were trained for the natural language inference (textual entailment) task using the architecture in Figure 3.1a. They evaluate the quality of the sentence representation by using sentence vectors as features in 12 different transfer tasks including Binary and multi-class text classification, semantic textual similarity, paraphrase detection etc. The results indicate that the BiLSTM with the max-

[2]The code and the pre-trained models are available on https://github.com/epfml/sent2vec



(a) General NLI Training Scheme. (b) Bi directional LSTM with max pooling

Figure 3.1: General NLI training scheme in Infersent with the best architecture; Bi directional LSTM with max pooling (Conneau et al., 2017).

pooling operation performs best on these tasks (Conneau et al., 2017). The architecture of BiLSTM with the max-pooling model is shown in Figure 3.1b.

Facebook released two models to derive the sentence embeddings. One model is trained with GloVe (Pennington et al., 2014) which in turn has been trained on text preprocessed with the PTB tokeniser. The other model is trained with fastText (Mikolov et al., 2018) which has been trained on text preprocessed with the MOSES tokeniser. We used both models in our experiments[3].

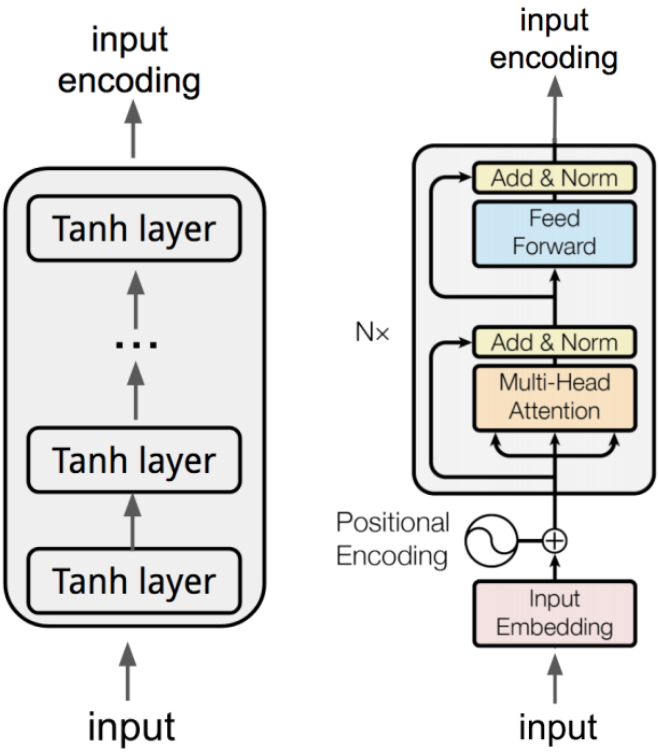
Universal Sentence Encoder The Universal Sentence Encoder (Cer et al., 2018) released by Google is the third and final sentence encoder we employed

[3]The code and the pre-trained models are available on https://github.com/facebookresearch/InferSent

in our study. This is again an unsupervised sentence encoder. It comes with two versions, i.e. one trained with the Transformer encoder and the other trained with Deep Averaging Network (DAN). Both architectures are outlined briefly below. The two have a trade-off between accuracy and computational resource requirement. While the one with the Transformer encoder has higher accuracy, it is computationally more expensive. The one with DAN encoding is computationally less expensive but with slightly lower accuracy.

The original Transformer encoder model is comprised of an encoder and decoder. Since our research is focussed on encoding sentences to vectors, we only use its encoder part. The encoder is composed of a stack of six identical layers (Cer et al., 2018). Each layer has two sub-layers. The first is a multi-head self-attention mechanism, and the second is a simple, position-wise, fully connected feedforward network. Cer et al. (2018) employed a residual connection around each of the two sub-layers, followed by layer normalisation. Since the model contains no recurrence and no convolution, for the model to make use of the order of the sequence, it must inject some information about the relative or absolute position of the tokens in the sequence, this is what the "positional encodings" do. The transformer-based encoder achieves the best overall transfer task performance. However, this comes at the cost of computing time and memory usage, scaling dramatically with sentence length.

Deep Averaging Network (DAN) is much simpler where input embeddings for words and bi-grams are first averaged together and then passed through a feedforward deep neural network to produce sentence embeddings. The primary



(a) DAN Architecture (b) Transformer Architecture

Figure 3.2: Two architectures in Universal Sentence Encoders (Cer et al., 2018).

advantage of the DAN encoder is that computation time is linear in relation to the length of the input sequence. With this sentence encoder, we used both architectures in our experiments[4]. Unlike the other sentence encoders, Google officially released two multilingual models for Universal Sentence Encoder.

[4]Pre-trained sentence encoder for transformer model is available on https://tfhub.dev/google/universal-sentence-encoder-large and pre-trained sentence encoder for DAN model is available on https://tfhub.dev/google/universal-sentence-encoder.

3.2 Exploring Sentence Encoders in English STS

Adapting sentence encoders for STS is an easy task. If embeddings from the two sentences are closer, the sentences are said to be semantically similar. For the approach, first, the two sentences are passed through the sentence encoders to get the embeddings. Then we calculate the cosine similarity between the resulting embeddings, that represents the textual similarity of the two input sentences. Suppose the two vectors for two sentences X and Y are a and b correspondingly. In that case, we calculate the cosine similarity between a and b as of equation 3.1 and use that value to represent the similarity between the two sentences.

$$\begin{aligned} \cos(\mathbf{a}, \mathbf{b}) &= \frac{\mathbf{ab}}{\|\mathbf{a}\|\|\mathbf{b}\|} \\ &= \frac{\sum_{i=1}^{n} \mathbf{a}_i \mathbf{b}_i}{\sqrt{\sum_{i=1}^{n} (\mathbf{a}_i)^2} \sqrt{\sum_{i=1}^{n} (\mathbf{b}_i)^2}} \end{aligned} \tag{3.1}$$

First, we experimented with English STS datasets as explained in Section 1.1. For the experiments, we used all the sentence encoders discussed in Section 3.1. For **Sent2vec**, we used the pre-trained Sent2vec model, *sent2vec_wiki_bigrams* trained on English Wikipedia articles. Using this, we were able to represent a sentence from a 700 dimensional vector. For **Infersent**, as we mentioned previously, there are two pre-trained models available; *infersent1* which was trained using GloVe (Pennington et al., 2014), and *infersent2* which was trained using fastText (Mikolov et al., 2018). Both models have been trained on the SNLI dataset, which consists of 570k human-generated English sentence pairs,

manually labelled with one of three categories: entailment, contradiction and neutral (Bowman et al., 2015). Using this, we could represent a sentence from a 512 dimensional vector. For **Universal Sentence Encoder**, we used the 'universal-sentence-encoder' (DAN architecture) and the 'universal-sentence-encoder-large' (Transformer architecture), which were trained on text resources including Wikipedia and news articles. Using this method, we could also represent a sentence from a 512 dimensional vector.

We evaluated these three sentence encoders against the three English STS datasets we explained in Section 1.1; SICK, STS2017 and QUORA. Table 3.1 shows the results for the SICK dataset, Table 3.2 shows the results for the STS 2017 dataset and Table 3.3 shows the results for the Quora Questions Pairs dataset.

Model	ρ	τ
ELMo $\bigoplus$ *BERT*	0.753	0.669
Sent2vec	0.759	0.672
Infersent1	0.763	0.679
Infersent2	0.769	0.684
USE (DAN)	0.772	0.695
USE (Transformer)	$0.780^{\dagger}$	$0.721^{\dagger}$

Table 3.1: Results for SICK dataset with sentence encoders. For each sentence encoder, Pearson Correlation (ρ) and Spearman Correlation (τ) are reported between the predicted values and the gold labels of the test set. USE denotes Universal Sentence Encoder. Additionally, we report the results of the best model from Chapter 2; *ELMo* $\bigoplus$ *BERT*. The best result from all the methods is marked with †.

As can be seen in the results, the Universal Sentence Encoder outperformed all other sentence encoders in all of the English STS datasets. From the two architectures available in the Universal Sentence Encoder, the Transformer

Model	ρ	τ
ELMo $\bigoplus$ *BERT*	0.654	0.616
Sent2vec	0.673	0.645
Infersent1	0.703	0.696
Infersent2	0.711	0.701
USE(DAN)	0.725	0.703
USE(Transformer)	$0.744^{\dagger}$	$0.721^{\dagger}$

Table 3.2: Results for STS 2017 dataset with sentence encoders. For each sentence encoder, Pearson Correlation (ρ) and Spearman Correlation (τ) are reported between the predicted values and the gold labels of the test set. USE denotes Universal Sentence Encoder. Additionally, we report the results of the best model from Chapter 2; *ELMo* $\bigoplus$ *BERT*. The best result from all the methods is marked with †.

Model	RMSE
ELMo $\bigoplus$ *BERT*	0.566
Sent2vec	0.632
Infersent1	0.642
Infersent2	0.653
USE(DAN)	0.666
USE(Transformer)	$0.686^{\dagger}$

Table 3.3: Results for QUORA dataset with sentence encoders. For each sentence encoder model, Root Mean Squared Error (RMSE) is reported. USE denotes Universal Sentence Encoder. Additionally, we report the results of the best model from Chapter 2; *ELMo* $\bigoplus$ *BERT*. The best result is marked with †.

architecture outperforms the DAN architecture as explained in their paper (Cer et al., 2018). Furthermore, it should be noted that in all three datasets, sentence encoders outperform the embedding aggregation based Smooth Inverse Frequency method that performed best in Chapter 2. This concludes that sentence encoders generally perform better than embedding aggregation techniques in STS.

With these results, we can answer our **RQ1**, sentence encoders can be easily adapted and perform well in English STS tasks. However, most of these models

are complex in nature, resulting in more processing time/resources, which can be problematic in some situations.

3.3 Portability to Other Languages

Our **RQ2** targets the multilingual aspect of sentence encoders; *How well the sentence encoders perform in different languages?*. To answer this, we evaluated our method in the Arabic STS and Spanish STS datasets that were introduced in Chapter 1. With these experiments, we identified a major weakness in sentence encoders; sentence encoders pre-trained in different languages are not easy to find.

Consider **Infersent**, it was pre-trained using the SNLI dataset, which consists of 570k human-generated English sentence pairs, manually labelled with one of three categories: entailment, contradiction and neutral (Bowman et al., 2015). If someone is adapting **Infersent** to a language other than English, they need to have a corpus comparative to SNLI with a similar size. Annotating such a corpus for a different language would be challenging. Even though there are some attempts to create such a corpus including XNLI (Conneau et al., 2018), the number of annotated instances are very limited. This makes it challenging to adapt **Infersent** to other languages, which is an explicit limitation of the Infersent architecture.

The other two sentence encoders we experimented within this Chapter, Sent2vec and Universal Sentence Encoder, are in a better position in multilingualism, compared to Infersent as they don't require a large annotated

corpus like SNLI. Both of these sentence encoders have been trained on unsupervised textual data that will be easy to find in most languages. However still, they need powerful computing resources to train the models, which is a challenge when adapting these sentence encoders to different languages.

For **Sent2Vec**, there was no Arabic pre-trained model available. However, there is a Spanish Sent2vec model[5] available which is pre-trained on Spanish Unannotated Corpora[6]. Using that, we could represent a Spanish sentence with a 700 dimensional vector. For **Universal Sentence Encoder**, there is a multilingual version which supports 16 languages[7] including Arabic and Spanish (Yang et al., 2020). This multilingual model is available in both architecture in Universal Sentence Encoder; DAN and Transformer[8]. Using that, we could represent the Arabic and Spanish sentences with a 512 dimensional vector. As mentioned before, for **Infersent**, we could not find any pre-trained models that support Arabic or Spanish. Therefore, we did not use Infersent in our multilingual experiments. The Arabic and Spanish STS results with the above mentioned sentence encoders are available in Table 3.4.

As can be seen in results, similarly to the English datasets, from the sentence

[5]The pre-trained model is available on https://github.com/BotCenter/spanish-sent2vec

[6]Spanish Unannotated Corpora is available on https://github.com/josecannete/spanish-corpora

[7]The model currently supports Arabic, Chinese-simplified, Chinese-traditional, English, French, German, Italian, Japanese, Korean, Dutch, Polish, Portuguese, Spanish, Thai, Turkish and Russian.

[8]The multilingual Universal Sentence Encoder with DAN architecture is available on https://tfhub.dev/google/universal-sentence-encoder-multilingual/3 and the multilingual Universal Sentence Encoder with Transformer architecture is available on https://tfhub.dev/google/universal-sentence-encoder-multilingual-large/3.

Language	Sentence Encoder	ρ	τ
Arabic	*ELMo* $\bigoplus$ *BERT*	0.624	0.589
	USE (DAN)	0.654	0.612
	USE (Transformer)	0.668†	0.635†
Spanish	*ELMo* $\bigoplus$ *BERT*	0.712	0.663
	USE (DAN)	0.723	0.6.682
	Sent2vec	0.725	0.688
	USE (Transformer)	0.741†	0.702†

Table 3.4: Results for Arabic and Spanish STS with different sentence encoders. For each sentence encoder, Pearson Correlation (ρ) and Spearman Correlation (τ) are reported between the predicted values and the gold labels of the test set. USE denotes Universal Sentence Encoder. Additionally, we report the results of the best models from Chapter 2; *ELMo* $\bigoplus$ *BERT*. The best result for each language is marked with †.

encoders we considered, Universal Sentence Encoder with the Transformer architecture gave the best results for the Arabic and Spanish datasets. From the two architectures available in the Universal Sentence Encoder, the Transformer architecture outperforms the DAN architecture in both languages. Furthermore, it should be noted that in both languages, sentence encoders outperform the word embedding based Smooth Inverse Frequency method that performed best in Chapter 2.

From these experiments, we can answer our **RQ2**: *How well the sentence encoders can be adapted in different languages?*. Adapting sentence encoders to different languages is challenging since there are no pre-trained sentence encoder models available for many languages. However, in the cases where they are available, it is straightforward to use them in STS tasks, and they provide better results than other unsupervised STS methods.

3.4 Portability to Other Domains

To answer our **RQ3**: *how well the sentence encoders can be applied in different domains*, we evaluated the sentence encoders previously explained in this chapter against the Bio-medical STS dataset explained in Chapter 1. As aforementioned, the Bio-medical STS dataset does not have a training set. Therefore, only the unsupervised approaches can be applied to this dataset, providing an ideal opportunity for the sentence encoders we experimented with in this chapter.

However, we faced the same issue with the Bio-medical STS experiments that we encountered with the Arabic and Spanish STS experiments. There are not many options when it comes to sentence encoders that were trained in the Bio-medical domain (Tawfik and Spruit, 2020). For **Sent2vec**, we could find a pre-trained model in the Bio-medical domain, which was trained using PubMed data (Chen et al., 2019)[9]. However, for the other two sentence encoders, we could not find any available pre-trained sentence encoder models in the Bio-medical domain. Therefore, for Universal Sentence Encoder and Infersent, we used pre-trained sentence encoder models trained on general English texts for these experiment. The results are shown in Table 3.5.

As can be observed in the results, the sentence encoder trained on the Bio-medical domain; BioSentVec (Chen et al., 2019) outperformed the other approaches. In fact, when compared to the best results in the BIOSSES dataset,

[9]The code and the pre-trained model is available on https://github.com/ncbi-nlp/BioSentVec

Model	$\boldsymbol{\rho}$
Infersent2	0.294
Infersent1	0.301
USE(DAN)	o.321
USE(Transformer)	0.345
ELMo $\bigoplus$ *BERT*	0.708
BioSentVec (Chen et al., 2019)	$0.810^{\dagger}$

Table 3.5: Results for BIOSSES dataset with different sentence encoders compared with top results reported for BIOSSES. Additionally, we report the results of the best model from Chapter 2; *ELMo* $\bigoplus$ *BERT*. For each variant, Pearson Correlation ($\boldsymbol{\rho}$) is reported between the predicted values and the gold labels of the test set. The best result is marked with †.

BioSentVec outperforms all of them, meaning that BioSentVec delivers the best result for BIOSSES. It should be noted that out of domain sentence encoders like Universal Sentence Encoder and Infersent that were not trained on the Bio-medical domain, performed very poorly in these experiments. The simple *ELMo* $\bigoplus$ *BERT* embedding aggregation based approach we experimented with in Chapter 2 outperforms Universal Sentence Encoder and Infersent by a large margin. This may be due to the fact that there is a large number of out-of-vocabulary words for these general sentence coders in the Bio-medical domain, unlike *ELMo* $\bigoplus$ *BERT* where we used the ELMo and BERT models trained on Bio-medical domain. We can conclude that sentence encoders can be successfully adapted for STS in different domains. However, they won't succeed unless they are pre-trained in that particular domain.

With these findings, we answer our **RQ3**: *Is it possible to adapt sentence encoders in different domains?*. We showed that it is possible to adapt sentence encoders to a different domain. However, it is difficult since pre-trained sentence

encoder models are not common in most domains. The sentence encoders pre-trained on a general domain perform poorly on a specific domain such as Biomedical.

3.5 Conclusions

In this chapter, we experimented with another unsupervised STS method; sentence encoders. We evaluated three popular sentence encoders; Sent2vec, Infersent and Universal Sentence Encoder, in three English STS datasets. The results show that the sentence encoders outperform other unsupervised STS methods. From the sentence encoders, the Universal Sentence Encoder with the Transformer architecture performs best in all three datasets. Furthermore, we evaluated sentence encoders in different languages and domains. We faced a challenge when adapting sentence encoders to different languages and domains; the pre-trained sentence encoders that support different languages and domains are not common when compared to the word embedding/ contextual word embedding models available on those languages and domains. This is because the sentence encoders take a lot of time to train, and some of the sentence encoders such as Infersent require specific training data, which is difficult to compile in most of the languages and domains. Also, since the applications of sentence encoders are limited compared to word embeddings/ contextual word embeddings, people have not dedicated time to creating language-specific and domain-specific sentence encoders. However, when available, they perform very well in the relevant tasks. We experimented with using sentence encoders that

were trained on a general domain in a different but specific domain like Biomedical. However, the results obtained were unsatisfactory. Therefore, we can conclude that using sentence encoders on a different domain to their pre-trained domain do not perform well.

Being an unsupervised STS method, the sentence encoders have the obvious advantage of not needing a training set for STS. However, they still need a pre-trained sentence encoding model, which is not available in most of the languages and domains. This is a major drawback for sentence encoders compared to the embedding aggregation methods since the word embedding/ contextual embedding models are commonly available in many languages and domains. This is further aggravated by the fact that sentence encoders perform poorly on the domains they did not see in the training process. This suggests that we should only use sentence encoders when available in a certain domain/ language; otherwise, embedding aggregation methods like Smooth Inverse Frequency would fare better.

As future work, the experiments could be extended into recently released sentence encoders such as LASER (Artetxe and Schwenk, 2019). LASER supports 93 languages, including low resource languages such as Kabyle, Malagasy, Sinhala etc. This should solve some of the multilingual issues we experienced with sentence encoders in this chapter. Very recently, sentence encoders have been explored with Siamese neural architectures and Transformers. We discuss these in Chapters 4 and 5.

Chapter 4

Siamese Neural Networks for STS

A *Siamese Neural Network* refers to a class of neural network architecture that contains two or more identical subnetworks. This is usually employed in applications that require comparisons between two or more inputs (signature verification, image similarity etc.). The network can take two or more inputs at the same time and they will be processed by different subnetworks simultaneously. The subnetworks have the same configuration with identical parameters and weights. The training process is mirrored across all of the subnetworks, meaning that the parameters remain constant across all of the subnetworks. Each subnetwork contains a traditional perceptron model such as LSTM, CNN etc. The neural network compares the outputs of the two subnetworks through a distance metric such as cosine distance, and the error is propagated back (Mueller and Thyagarajan, 2016). In the testing phase, the neural network predicts whether the two inputs are different through this similarity measure (Neculoiu et al., 2016).

Siamese neural networks have been employed in many applications in different areas such as signal processing (Thiolliere et al., 2015; Manocha et al., 2018; Shon et al., 2017; Zhang et al., 2019b; Švec et al., 2017; Gündoğdu et al., 2017;

Siddhant et al., 2017; Zeghidour et al., 2016), biology (Zheng et al., 2018; Szubert et al., 2019), chemistry and pharmacology (Jeon et al., 2019), geometry (Sun et al., 2020b), computer vision (Baldi and Chauvin, 1993; Chopra et al., 2005; He et al., 2018; Paisios et al., 2012; Yi et al., 2014; Lefebvre and Garcia, 2013; Taigman et al., 2014; Berlemont et al., 2015; Kassis et al., 2017; Hanif, 2019), physics (Zou et al., 2018; De Baets et al., 2019), robotics (Utkin et al., 2017a; Utkin et al., 2017b; Zeng et al., 2018), video processing (Ryoo et al., 2018; Liu et al., 2018b; Liu et al., 2018a; Lee and Kim, 2019) etc. They have also been used in NLP tasks (Yih et al., 2011; Kumar et al., 2018; González et al., 2019) including STS (Das et al., 2016; Neculoiu et al., 2016; Mueller and Thyagarajan, 2016). In fact, the best result from the SICK dataset (Marelli et al., 2014) was provided by a Siamese neural network (Mueller and Thyagarajan, 2016) which shows that state-of-the-art in STS are Siamese neural networks.

In addition to providing state-of-the-art results in STS, there are additional advantages of using Siamese neural networks. As previously mentioned, in Siamese neural networks the weights are shared across subnetworks resulting in fewer parameters to train, this in turn means that less training data is required and there is a lower likelihood of overfitting (Ranasinghe et al., 2019b). Given the amount of human labour required to produce datasets for STS, Siamese neural networks can provide an ideal solution. Another advantage is that when trained on a STS task the Siamese neural network architecture can be adapted as a sentence encoder. The output vector of the subnetwork is a semantically rich vector representation of the input sentence (Mueller and Thyagarajan, 2016).

These advantages have motivated us to further explore Siamese neural network architectures in STS.

We address four research questions in this chapter:

RQ1: Can an existing state-of-the-art Siamese neural network architecture be modified to provide better STS results?

RQ2: Can the method be further improved with transfer learning and data augmentation techniques?

RQ3: Can the proposed Siamese neural network be easily adapted for different languages?

RQ4: How well does the proposed Siamese neural network perform in a different domain?

The main contributions of this chapter are as follows.

1. We propose a GRU (Gated Recurrent Unit) based Siamese neural network that outperformed the state-of-the-art LSTM based Siamese neural network in small STS datasets.

2. We propose an LSTM (Long Short Term Memory) and self-attention based Siamese neural network that outperformed the state-of-the-art LSTM based Siamese neural network in large STS datasets.

3. We propose further enhancements to the architecture using transfer learning and data augmentation.

4. We evaluate how well the proposed Siamese neural network architecture performs in different languages and domains.

5. The code and the pre-trained models are publicly available to the community[1].

The rest of this chapter is organised as follows. Section 4.1 describes the existing research done on Siamese neural networks. Section 4.2 discusses the methodology and the experiments undertaken with three English STS datasets. Sections 4.2.1 and 4.2.2 provide more experiments which improve the results. Experiments conducted with other languages and domains are shown in Sections 4.3 and 4.4. The chapter finishes with conclusions and ideas for future research directions in Siamese neural networks.

4.1 Related Work

The Siamese neural networks are prevalent in the machine learning community. The first appearance of Siamese neural networks date back to 1993 when they were first introduced by Bromley et al. (1993) to detect forged signatures. By comparing two handwritten signatures, this Siamese neural network could predict if the two signatures were both original or if one was a forgery. Preceding this, Baldi and Chauvin (1993) introduced a similar artificial neural network able to recognise fingerprints, though by a different name.

Siamese neural networks have been employed for various uses since these initial applications. In the audio and speech signal processing field, Thiolliere et al. (2015) merged a dynamic-time warping based spoken term discovery (STD)

[1]The public GitHub repository is available on https://github.com/tharindudr/Siamese-recurrent-rrchitectures

system with a Siamese deep neural network for the automatic discovery of linguistic units from raw speech. Manocha et al. (2018) used a Siamese model to detect all semantically similar audio clips from an input audio recording, and Shon et al. (2017) employed a Siamese model to recognise Arabic dialects from the Arabic speech content found in media broadcasts. In biology, Zheng et al. (2018) implemented a Siamese neural network to compare DNA sequences and recently Szubert et al. (2019) presented a Siamese neural network-based technique for a visualisation and interpretation of single-cell datasets. Image analysis is the field with the highest number of applications for the Siamese neural networks. Recognising fingerprints (Baldi and Chauvin, 1993), similar image detection (Chopra et al., 2005; He et al., 2018; Paisios et al., 2012; Yi et al., 2014; Lefebvre and Garcia, 2013), face verification (Taigman et al., 2014), gesture recognition (Berlemont et al., 2015), handwriting analysis (Kassis et al., 2017) and patch matching (Hanif, 2019) are some examples of them. All of these tasks involve the comparison of two or more things.

Recently, Siamese neural networks have also been employed in NLP. Yih et al. (2011) proposed similarity Learning via Siamese Neural Network (S2Net), a technique able to discriminatingly learn the concept vector representations of text words. Kumar et al. (2018) used a Siamese neural network to recognise clickbaits in online media outlets. González et al. (2019) proposed a natural language processing application of the Siamese neural network for extractive summarisation, which means that their technique can extrapolate the most relevant sentences in a document.

Siamese neural networks are not limited to these applications and have also been implemented in STS tasks in NLP. Das et al. (2016) used a CNN based Siamese neural network to detect similar questions on question and answer websites such as Yahoo Answers, Baidu, Zhidao, Quora, and Stack Overflow. Neculoiu et al. (2016) employed a Siamese neural network based on Bidirectional LSTMs to identify similar job titles. The baseline we used for this chapter, MALSTM (Mueller and Thyagarajan, 2016) uses LSTM based Siamese neural network to produce semantic textual similarity, and it provides the best results for the SICK dataset outperforming other STS methods such as Tree-LSTMs (Tai et al., 2015). They use the exponent of the negative Manhattan distance between two outputs from the two subnetworks as the similarity function. Due to the performance, this can be considered as the state-of-the-art Siamese neural network for STS. However, this architecture leaves considerable room for variation, which we exploit in this chapter as we explain in Section 4.2.

4.2 Exploring Siamese Neural Networks for STS

The basic structure of the Siamese neural network architecture used in our experiments is shown in Figure 4.1. It consists of an embedding layer that represents each sentence as a sequence of word vectors. This sequence of word vectors is then fed into a Recurrent Neural Network (RNN) cell, which learns a mapping from the space of variable-length sequences of 300-dimensional vectors into a 50 dimensional vector. The sole error signal backpropagated during training stems from the similarity between these 50 dimensional vectors, which

can also be used as a sentence representation. Initially, the similarity function we used was based on Manhattan distance. To ensure that the prediction is between 0 and 1, we took the exponent of the negative Manhattan distance between two sentence representations. The similarity function was adapted from Mueller and Thyagarajan (2016). The proposed variants of our architecture are:

1. LSTM - Block A in Figure 4.1 contains a single LSTM cell. This is the architecture suggested by Mueller and Thyagarajan (2016)

2. Bi-directional LSTM - Block A in Figure 4.1 contains a single Bi-directional LSTM cell. Bi-directional LSTM tends to understand the context better than Unidirectional LSTM (Schuster and Paliwal, 1997).

3. GRU - Block A in Figure 4.1 contains a single GRU cell. GRUs have been shown to exhibit a better performance on smaller datasets (Chung et al., 2014).

4. Bi-directional GRU - Block A in Figure 4.1 contains a single Bi-directional GRU cell. Bi-directional GRUs tend to understand the context better than Unidirectional GRUs (Vukotić et al., 2016).

5. LSTM + Attention - Block A in Figure 4.1 contains a single LSTM cell with self attention (Vaswani et al., 2017a).

6. GRU + Attention - Block A in Figure 4.1 contains a single GRU cell with self attention (Vaswani et al., 2017a).

7. GRU + Capsule + Flatten - Block A in Figure 4.1 contains a GRU followed by a capsule layer and a flatten layer. Dynamic routing used between capsules performs better than a traditional max-pooling layer (Sabour et al., 2017).

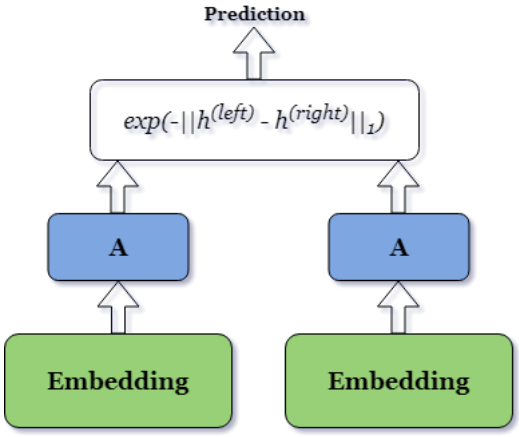


Figure 4.1: Basic structure of the Siamese neural network. Unit A is changed over the architectures.

As the word embedding model, we used Word2vec embeddings (Mikolov et al., 2013a) pre-trained on the Google news corpus[2]. Using this model, we represented each word as a 300 length vector. For the words that do not appear in the model, we used a random vector. We evaluated all of the above variations in the three English STS datasets we introduced in 1; SICK, STS 2017 and QUORA. We trained the Siamese models on the training sets in these datasets and evaluated them on the testing sets. The results are shown in Table 4.1, Table 4.2 and Table 4.3 respectively.

As can be seen in Tables 4.1 and 4.2, for the SICK and STS2017 datasets,

[2]Pre-trained Word2vec can be downloaded from https://code.google.com/archive/p/word2vec/

Model	ρ	τ
LSTM	0.802	0.733
Bi-LSTM	0.784	0.708
GRU	0.838†	0.780†
Bi-GRU	0.832	0.773
LSTM + Attention	0.827	0.765
GRU + Attention	0.818	0.751
GRU + Capsule + Flatten	0.806	0.733

Table 4.1: Results for SICK dataset with different variants of Siamese Neural Network. For each variant, Pearson Correlation (ρ) and Spearman Correlation (τ) are reported between the predicted values and the gold labels of the test set. The best result from all the variations is marked with †.

Model	ρ	τ
LSTM	0.831	0.762
Bi-LSTM	0.784	0.708
GRU	0.853†	0.811†
Bi-GRU	0.844	0.804
LSTM + Attention	0.830	0.791
GRU + Attention	0.825	0.782
GRU + Capsule + Flatten	0.806	0.765

Table 4.2: Results for STS 2017 dataset with different variants of Siamese Neural Network. For each variant, Pearson Correlation (ρ) and Spearman Correlation (τ) are reported between the predicted values and the gold labels of the test set. The best result from all the variations is marked with †.

the GRU based Siamese neural network model outperformed the LSTM based Siamese neural network model, which we used as a baseline, and this provided the best result. It can be seen that complex architectures that involve Bi-directional RNNs, Attention and Capsule mechanisms did not perform well when compared to simple architectures like GRU. We can conclude that for the smaller datasets like STS 2017 and SICK, the GRU based architecture performs better because GRU has fewer parameters than LSTM (Chung et al., 2014). With fewer parameters, the architecture does not need many training instances to optimise

Model	RMSE
LSTM	0.412
Bi-LSTM	0.402
GRU	0.415
Bi-GRU	0.408
LSTM + Attention	0.382†
GRU + Attention	0.398
GRU + Capsule + Flatten	0.421

Table 4.3: Results for QUORA dataset with different variants of Siamese Neural Network. For each variant, Root Mean Squared Error (RMSE) reported between the predicted values and the gold labels of the test set. The best result from all the variations is marked with †.

the weights during the training process.

However, when it comes to the big STS dataset, QUORA, the way the variants of the Siamese neural network behaves is different. As we introduced in Chapter 1, QUORA was the biggest STS dataset we experimented with, and it has 320,000 training instances. As a result, even complex architectures like RNNs with Attention get the opportunity to optimise their parameters and deliver good results. This can be seen in Table 4.3. For the QUORA dataset, the LSTM + Attention based Siamese neural network model outperformed the LSTM based Siamese neural network model, which we used as a baseline, and provided the best result. For bigger datasets, we can conclude that Siamese neural networks based on LSTM with Attention would outperform Siamese neural networks only with LSTMs.

From the variants we examined, one notable observation is the poor performance of capsules in Siamese architectures. Despite providing good results in many NLP tasks such as text classification (Sabour et al., 2017; Hettiarachchi

and Ranasinghe, 2019; Xia et al., 2018; Srivastava et al., 2018) and relation extraction (Zhang et al., 2019a; Zhang et al., 2018; Zhang and Geng, 2020), the capsule-based variant failed to outperform the simple LSTM based variant even in the bigger STS dataset. This observation implies that capsule-based Siamese neural networks will not be a good fit for STS tasks.

With these findings, we answer our **RQ1** in this chapter. We have improved the state-of-the-art Siamese neural network architecture and propose a GRU based Siamese neural network architecture for the smaller STS datasets and LSTM + Attention based Siamese neural network for larger STS datasets.

4.2.1 Impact of Transfer Learning

Transfer learning is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task. It is a popular approach in deep learning where pre-trained models are used as the starting point for a new task. This is usually done in scenarios where there is not enough data to train a neural network, so that starting from already finetuned weights would be advantageous (Houlsby et al., 2019; Ruder et al., 2019). Transfer learning has often provided good results for smaller datasets. Therefore, we explored the impact of transfer learning with Siamese neural networks in STS.

We saved the weights of the models that were trained on each STS dataset; SICK, STS 2017 and QUORA. We specifically used the two models that performed best in these datasets; Siamese neural network with GRU and the Siamese neural network with LSTM + Attention. We initiated training for each dataset, however

rather than training from scratch, we used the weights of the models trained on another STS dataset. We compared these transfer learning results to the results we got from training the model from scratch. We conducted this transfer learning experiment only on the STS2017 and SICK datasets since the QUORA dataset is already large and transfer learning from a smaller dataset to a larger dataset is nonsensical.

Start Model	STS2017	SICK
$STS2017_{GRU}$	0.853	(+0.01)
$STS2017_{LSTM+Aten}$	0.830	(+0.01)
$SICK_{GRU}$	(+0.01)	0.838
$SICK_{LSTM+Aten}$	(+0.01)	0.827
$QUORA_{GRU}$	(-0.02)	(-0.02)
$QUORA_{LSTM+Aten}$	(-0.04)	(-0.04)

Table 4.4: Results for transfer learning with different variants of Siamese Neural Network. For each transfer learning experiment we show the difference between with and without transfer learning. Non-grey values are the results of the experiments without transfer learning which we showed in the previous section. For ease of visualisation we only report the Pearson correlation ($\boldsymbol{\rho}$).

As can be seen in Table 4.4 some of the transfer learning experiments improved the results for the STS2017 and SICK datasets with both architectures. The results improved when we performed transfer learning from STS2017 $\Rightarrow$ SICK and SICK $\Rightarrow$ STS2017. This shows that transfer learning can improve the results in Siamese neural networks. However, when we performed transfer learning from QUORA $\Rightarrow$ STS2017 and QUORA $\Rightarrow$ SICK, the results did not improve. In fact, they decrease, despite QUORA being the largest STS dataset we experimented with. This finding is somewhat controversial as the general belief

in the community is that transfer learning from a larger dataset will improve the result. In this case, we believe that this happened because the QUORA dataset is very different to the other two datasets, as discussed in Chapter 1. Despite QUORA having a large number of training instances, when performing transfer learning, the neural network finds it difficult to optimise the weights for STS2017 and SICK as they were already optimised for a very different dataset; QUORA. This leads to a decrease in the results. On the other hand, transfer learning between STS2017 and SICK improved the results for both datasets since they are similar in nature, as we discussed in Chapter 1.

Therefore, we can conclude that transfer learning can improve the results for Siamese neural networks in STS. However, the transfer learning dataset should be picked carefully taking the similarity of the two datasets into consideration, rather than only considering the size of the dataset.

4.2.2 Impact of Data Augmentation

As we observed earlier, the neural networks perform better when there are large number of training instances. Therefore, many approaches have been taken to increase the number of training instances. Usually, this has resulted in better performance with neural networks (Wei and Zou, 2019). Therefore, we examined the impact of data augmentation on the Siamese neural network architectures proposed previously. We only conducted this experiment with the STS 2017 and SICK datasets as QUORA already has a large number of training instances.

We employed thesaurus-based augmentation in which 10,000 additional

training examples are generated by replacing random words with one of their synonyms from Wordnet (Miller, 1995). A similar approach has also been successfully adapted by Mueller and Thyagarajan (2016), and Zhang et al. (2015). We specifically used the two models that performed best with the bigger dataset and smaller dataset; Siamese neural network with GRU and Siamese neural network with LSTM + Attention. Since using transfer learning improved the results in the previous experiment, we trained the augmented training set on the transferred models; models trained on STS2017 for the SICK experiments and models trained on SICK for the STS2017 experiments. The results are shown in Table **??**.

Dataset	**Start Model**	ρ
SICK	$STS2017_{GRU}$	(+0.01)
	$STS2017_{LSTM+Aten}$	(+0.01)
STS2017	$SICK_{GRU}$	(+0.01)
	$SICK_{LSTM+Aten}$	(+0.01)

Table 4.5: Results for data augmentation with different variants of Siamese neural networks. For each data augmentation experiment, we show the difference between performing the data augmentation and without performing data augmentation. For ease of visualisation we only report the Pearson correlation (ρ).

As can be seen in Table **??**, data augmentation improved the results of all the experiments. However, even with the additional 10,000 training instances, the GRU based Siamese neural network outperformed the LSTM + Attention based Siamese neural network. We can conclude that simple data augmentation techniques improve the performance of Siamese neural networks in STS tasks. From the Siamese neural network experiments we conducted, our best results

for both the STS2017 and SICK datasets were provided by GRU based Siamese neural network when combined with transfer learning and data augmentation.

These observations answer our *RQ2* in this Chapter; we can use transfer learning and simple data augmentation techniques to improve the results of Siamese neural networks in STS.

Model	ρ
Jimenez et al. (2014)	0.807
Bjerva et al. (2014)	0.827
Zhao et al. (2014)	0.841
Siamese LSTM	0.863
Siamese GRU	0.882

Table 4.6: Results for SICK dataset with different variants of Siamese neural networks. For each variant, Pearson Correlation (ρ) is reported between the predicted values and the gold labels of the test set.

Model	ρ
Tian et al. (2017)	0.851
Siamese LSTM	0.852
Maharjan et al. (2017)	0.854
Cer et al. (2017)	0.855
Siamese GRU	0.862

Table 4.7: Results for STS2017 dataset with different variants of Siamese neural networks. For each variant, Pearson Correlation (ρ) is reported between the predicted values and the gold labels of the test set.

Furthermore, we compared the results of the best Siamese neural network variant with the best results submitted to the competitions (Cer et al., 2017; Marelli et al., 2014), and with the unsupervised STS methods we have experimented with so far in the thesis. As can be seen in Tables 4.6 and 4.7, the

GRU based Siamese neural network architecture outperforms the best systems submitted to both competitions. It also outperforms the unsupervised STS methods we have explored so far in the thesis. Therefore, we can conclude that Siamese architecture is currently the best system we have experimented with for English STS.

4.3 Portability to Other Languages

Our *RQ3* targets the multilingual aspect of the proposed approach; *Can the proposed Siamese neural network be easily adopted into different languages?*. To answer this, we evaluated our method in the Arabic STS and Spanish STS datasets that were introduced in Chapter 1. Our approach has the advantage that it does not rely on language-dependent features. As a result, the approach is easily portable to other languages, given the availability of pre-trained word embedding models in that particular language. The word embedding models, we used are AraVec (Soliman et al., 2017) [3] for Arabic and Spanish 3B words Word2Vec Embeddings (Bilbao-Jayo and Almeida, 2018)[4] for Spanish.

As can be seen in Tables 4.8 and 4.9, the GRU based Siamese neural network outperformed all other variants we experimented with in both Arabic and Spanish. As we discussed in Chapter 1, both of the Arabic and Spanish STS datasets we considered are small in size. Therefore, similarly to the STS2017

[3] AraVec has been trained on Arabic Wikipedia articles. The models are available on https://github.com/bakrianoo/aravec

[4] Spanish 3B words Word2Vec Embeddings have been trained on Spanish news articles, Wikipedia articles and Spanish Boletín Oficial del Estado (BOE; English: Official State Gazette). The model is available on https://github.com/aitoralmeida/spanish_word2vec

Model	ρ	τ
LSTM	0.746	0.690
Bi-LSTM	0.725	0.683
GRU	0.763†	0.723†
Bi-GRU	0.752	0.717
LSTM + Attention	0.741	0.703
GRU + Attention	0.739	0.691
GRU + Capsule + Flatten	0.712	0.679

Table 4.8: Results for Arabic STS dataset with different variants of Siamese Neural Network. For each variant, Pearson Correlation (ρ) and Spearman Correlation (τ) are reported between the predicted values and the gold labels of the test set. The best result from all the variations is marked with †.

Model	ρ	τ
LSTM	0.842	0.773
Bi-LSTM	0.814	0.782
GRU	0.863†	0.822†
Bi-GRU	0.851	0.813
LSTM + Attention	0.845	0.801
GRU + Attention	0.832	0.790
GRU + Capsule + Flatten	0.795	0.773

Table 4.9: Results for Spanish STS dataset with different variants of Siamese Neural Network. For each variant, Pearson Correlation (ρ) and Spearman Correlation (τ) are reported between the predicted values and the gold labels of the test set. The best result from all the variations is marked with †.

and SICK datasets, the GRU based Siamese neural network outperforms other architectures as GRU does not need a lot of training instances to optimise its weights. It should be noted that it is very easy to adapt this STS method in a different language. We only changed the embeddings to the new language and then performed the training.

Furthermore, we compared the results of the best Siamese neural network variant with the best results submitted to the competition (Cer et al., 2017), and with the unsupervised STS methods we have experimented with so far in the

thesis.

Model	ρ
Tian et al. (2017)	0.744
Nagoudi et al. (2017)	0.746
Siamese LSTM	0.746
Wu et al. (2017)	0.754
Siamese GRU	0.763

Table 4.10: Results for Arabic STS dataset with different variants of Siamese Neural Network. For each variant, Pearson Correlation (ρ) is reported between the predicted values and the gold labels of the test set.

Model	ρ
Siamese LSTM	0.842
Hassan et al. (2017)	0.848
Wu et al. (2017)	0.850
Tian et al. (2017)	0.855
Siamese GRU	0.863

Table 4.11: Results for Spanish STS dataset with different variants of Siamese Neural Network. For each variant, Pearson Correlation (ρ) is reported between the predicted values and the gold labels of the test set.

As can be seen in the results, transformer-based STS methods outperformed all the other supervised and unsupervised STS models in both languages. They outperformed the top systems from the competition in both languages. From the experimented pre-trained transformer models, language-specific models such as BETO and AraBERT outperformed the general multilingual models. With these observations, we can conclude that transformers are currently state-of-the-art in Arabic and Spanish STS. Furthermore, it should be noted that it is straightforward to adapt transformers to a different language. We only changed

the pre-trained model to the new language and performed the training.

These observations answer our **RQ3**; the Siamese architectures that we propose in this chapter can be successfully adapted in different languages by changing the word embeddings and the training dataset.

4.4 Portability to Other Domains

To answer our *RQ4*; how well the proposed Siamese neural network architecture can be applied in different domains, we evaluated our method on the Bio-medical STS dataset explained in Chapter 1 (BIOSSES). As we mentioned previously, the Bio-medical STS dataset does not have a training set. Therefore, we had to follow a transfer learning strategy to evaluate Siamese neural networks on the Bio-medical STS dataset. We used the pre-trained English STS models and performed inference on the Bio-medical STS dataset. We can refer to this as a *"zero-shot transfer learning"* since the pre-trained English STS models did not see any Bio-medical data.

For this transfer learning strategy, we considered two word embedding models; the general Word2vec model we used before Mikolov et al. (2013a) that was pre-trained on Google news corpus, and BioWordVec Zhang et al. (2019c), which has trained Word2vec on a combination of PubMed and PMC texts[5]. With each word embedding model, we trained a Siamese neural network based on GRU and a Siamese neural network based on LSTM + Attention (the two best models we had from the English STS experiments) and evaluated them on the BIOSSES

[5]The model is availble on https://bio.nlplab.org/

dataset.

Data	Model	Word2vec	BioWordVec
STS2017	*Siamese GRU*	0.651	0.721
	Siamese LSTM+Atten	0.612	0.701
SICK	*Siamese GRU*	0.642	0.719
	Siamese LSTM+Atten	0.608	0.699
QUORA	*Siamese GRU*	0.591	0.622
	Siamese LSTM+Atten	0.603	0.634

Table 4.12: Results for transfer learning with different variants of Siamese neural networks in BIOSSES dataset. **Data** column shows the datasets we performed transfer learning from and **Model** column displays the Siamese variant we employed. Two considered word embedding models are **Word2vec** and **BioWordVec**. For ease of visualisation we only report the Pearson correlation ($\boldsymbol{\rho}$).

As you can see in Table 4.12, Siamese neural architectures provided satisfactory results. We got the best result from the Siamese neural network based on GRU, when trained on STS 2017 using BioWordVec. However, the results from the SICK dataset are not far behind. There was a clear improvement when the English STS model was trained using BioWordVec rather than general Word2vec embeddings. This may be because most of the Bio-medical words that appear in the BIOSSES dataset are out of vocabulary in general Word2vec embeddings, which can cause problems for the neural network when it observes them in the testing phase. It should be noted that in this experiment, when we performed transfer learning from the QUORA dataset, the results are lower than when we performed transfer learning from SICK or STS 2017. This again may be due to the fact that the SICK and STS2017 datasets have a similar annotation strategy to the BIOSSES dataset as discussed in Chapter 1. Even though QUORA

has a large number of training instances, it can't produce good transfer learning results because its annotation strategy is different.

Model	ρ
ELMo $\bigoplus$ *BERT*	0.708
Siamese $GRU_{STS2017}$	0.719
Soğancıoğlu et al. (2017)	0.754
BioSentVec Chen et al. (2019)	0.810

Table 4.13: Results for BIOSSES dataset with different variants of Siamese Neural Network compared with top results reported for BIOSSES. For each variant, Pearson Correlation (ρ) is reported between the predicted values and the gold labels of the test set.

Furthermore, we compared our results with the best results reported for the dataset. The results are shown in Table 4.13. The best model we had in Table 4.12 which is the Siamese GRU model trained on the STS2017 dataset, is represented as *Siamese* $GRU_{STS2017}$. As shown in Table 4.13, our method provides satisfactory results compared with the best approaches submitted to the BIOESS dataset. However, the unsupervised method we experimented with in the previous chapter with BioSentVec (Chen et al., 2019), comfortably outperformed the Siamese neural network approaches we explored in this chapter. We can answer our **RQ4**: *How well does the proposed Siamese neural network perform in a different domain?* with these findings. The Siamese neural network architectures can be adapted to different domains by changing the pre-trained word embeddings. However, without a proper training set, the results are not strong.

4.5 Conclusions

This chapter experimented with using Siamese neural networks for calculating semantic similarity between pairs of texts and compared them with other unsupervised/ supervised approaches. We used an existing Siamese neural network as the baseline; MALSTM (Mueller and Thyagarajan, 2016) and explored six different variants of Siamese neural networks. We experimented with three English STS datasets, SICK, STS2017 and QUORA. For the smaller STS datasets; SICK and STS2017, we show that the Siamese neural network based on GRU outperforms the baseline. For the larger STS dataset; QUORA, we show that Siamese neural network with LSTM and Attention outperforms the baseline. Also, we show that we can further improve the results with transfer learning and data augmentation techniques. However, we experienced that performing transfer learning from a bigger dataset does not always improve the results. The quality of the dataset which was used for transfer learning also matters. We show that Siamese neural network based on GRU outperforms the top submissions in both *SemEval 2017 task 1* (Cer et al., 2017) and *SemEval 2014 task 1* (Marelli et al., 2014). The data augmentation techniques we used in this chapter are language-dependent as they rely on WordNet (Miller, 1995). However, as future work, we can experiment with data augmentation techniques that are not language dependant and relies on word embeddings (Kumar et al., 2020b).

We extended the experiments with the Siamese neural network architectures to the Arabic and Spanish STS datasets (Cer et al., 2017). In these experiments,

the GRU based Siamese neural network architecture again outperformed all the systems submitted to the shared task and outperformed all of the STS methods we have experimented with so far in this part of the thesis. This proves that the Siamese neural network that we propose in this study, can be adapted to different languages. Furthermore, we performed experiments with the BIOSSES dataset. However, since the BIOSSES dataset does not have a training set, we used transfer learning based zero-shot learning when Siamese neural networks are applied. Even though they provided satisfactory results, Siamese neural networks could not outperform the sentence encoder based method we explored in Chapter 3. We can conclude that even though the Siamese neural networks can be adapted into different domains by changing the word embedding model, they do not provide strong results without a proper training set.

Since word embedding models are now available in most languages, including low resource languages such as Urdu (Haider, 2018), Telugu (Kumar et al., 2020a) and domains such as the legal domain (Chalkidis and Kampas, 2019), the method we experimented with in this chapter can be useful for many languages and domains. However, one drawback is the need for STS training data in each language and domain, this can be challenging in many scenarios.

As future work, it would be interesting to experiment transfer learning between languages with cross-lingual embeddings such as fastText (Mikolov et al., 2018) using Siamese neural networks. Such an approache will train an STS model on resource-rich languages like English and project the prediction for other languages using the zero-shot transfer learning. It would be a potential

solution to satisfy the training data requirement for low resource languages.

With the introduction of transformer models such as BERT (Devlin et al., 2019) and XLNet (Yang et al., 2019b), Siamese neural networks have evolved by utilising transformers in their architectures (Reimers and Gurevych, 2019). We will discuss these further in Chapter 5.

Chapter 5

Adapting Transformers for STS

Transformers can be considered as the most significant revolution that happened in NLP in recent years. Similarly to Recurrent Neural Networks (RNNs), transformers handle sequential input data, such as natural language, for tasks including machine translation and text summarisation. However, unlike RNNs, transformers do not necessarily process the sequences in order. Instead, the attention mechanism in transformers handles the context for any position in the input sequence. For example, if the input data is a sentence, the transformer does not need to process the beginning of the sentence before the end. Since RNNs process the sequence in order, they often forget the content of distant positions in the sequence. This nature of the RNNs causes problems when processing long sequences. Furthermore, since RNNs process word by word, it is hard to parallelise the work for processing sentences. The attention mechanism in transformers can address both of these issues (Vaswani et al., 2017b).

Transformers were first introduced in Vaswani et al. (2017b), where the authors used a transformer architecture for the sequence to sequence tasks such as machine translation. They show that an architecture with only attention mechanisms and without any RNNs can improve the results in the sequence to

sequence tasks. The first notable research in using transformers for language modelling was OpenAI GPT (Radford et al., 2018). OpenAI GPT adapts a pre-training followed by a fine-tuning scheme which means that once pre-trained, it can be fine-tuned to a large number of downstream NLP tasks such as text classification, named entity recognition etc. However, the first breakthrough in using transformer models for language modelling happened with the introduction of BERT (Devlin et al., 2019). As we explain in Section 5.1, BERT employs a masked language modelling (MLM) objective, which brings improvements over OpenAI GPT. Later, different variants of BERT were proposed by the NLP community. We explain some of these models that we used in this chapter in Section 5.1. All of these transformer models follow the same fine-tuning scheme of BERT.

Transformer models, which we have experimented with in this chapter, use special tokens to obtain a single contiguous sequence for each input sequence. Specifically, the first token is always a special classification token ([CLS]), and sentence pairs are separated using a special token ([SEP]). The final hidden state of [CLS] is used for the sentence-level fine-tuning tasks such as text classification, and the final hidden state of each token is used for the token-level fine-tuning tasks such as named entity recognition. The fine-tuning scheme in transformers is usually simple. For example, transformer models can be adapted to text classification tasks by adding a softmax layer on top of [CLS] token. Furthermore, the fine-tuning scheme is very efficient as the parameters in the transformer model are already optimised in the pre-training process. Therefore,

transformer models have been prevalent and successful in many NLP tasks (Devlin et al., 2019).

In this chapter, we experiment with different transformers models in a variety of STS datasets. We address four research questions in this chapter:

RQ1: How well do the existing state-of-the-art transformer models perform in the STS task?

RQ2: Can the method improve with the transfer learning and data augmentation techniques?

RQ3: Can the transformer model be easily adapted to different languages?

RQ4: How well do the proposed transformer models perform in a different domain?

The main contributions of this chapter are as follows.

1. We evaluate five popular transformer models in three English STS datasets. We compare the results with the previous STS methods and show that transformer-based STS methods outperform all the other STS methods we have experimented with in this thesis.

2. We propose further enhancements to the architecture using transfer learning and data augmentation.

3. We evaluate how well the transformer models perform on STS datasets in different languages and domains.

4. The code and the pre-trained models are publicly available to the

community[1]. We have published the code as a python library [2] and by the time of writing this chapter, it has more than 3,000 downloads from the community.

The rest of this chapter is organised as follows. Section 5.1 describes the pre-trained transformer models used in this chapter. Section 5.2 discusses the architecture and 5.3 shows the experiments conducted with three English STS datasets. Sections 5.3.1 and 5.3.2 provide more experiments to improve the results. Experiments done with other languages and domains are shown in Sections 5.4 and 5.5. Section 5.6 discusses the recent developments carried out with integrating transformers into Siamese architectures, addressing a key issue in using transformers in STS. The chapter finishes with conclusions and ideas for future research directions in transformers.

5.1 Related Work

As we mentioned before, after the introduction of BERT (Devlin et al., 2019), many variants of different transformer models have been proposed by adding minor modifications to the original BERT transformer. Usually, these modifications have resulted in improvements in the fine-tuning scheme for the downstream NLP tasks. Expecting a similar behaviour for the STS task, we evaluated the following transformer models for the experiments in this chapter.

[1]The public GitHub repository is available on https://github.com/tharindudr/STS-Transformers.

[2]The developed python library is available on https://pypi.org/project/ststransformers/.

BERT (Devlin et al., 2019) proposes an MLM objective, where some of the tokens of the input sequence are randomly masked, and the objective is to predict these masked positions, taking the corrupted sequence as input. BERT applies a Transformer encoder to attend to bi-directional contexts during pre-training. In addition, BERT uses a next-sentence-prediction (NSP) objective. Given two input sentences, NSP predicts whether the second sentence is the next sentence of the first sentence. The NSP objective aims to improve the tasks, such as question answering and natural language inference, which require reasoning over sentence pairs.

RoBERTa (Liu et al., 2019) makes a few changes to the BERT architecture and achieves substantial improvements. These changes include: (1) Training the model longer with larger batches and more data; (2) Removing the NSP objective; (3) Training on longer sequences; (4) Dynamically changing the masked positions during pre-training. The authors show that these changes lead to significant improvements in the downstream NLP tasks.

ALBERT (Lan et al., 2020) proposes two parameter-reduction techniques (factorised embedding parameterisation and cross-layer parameter sharing) to lower memory consumption and speed up training. Furthermore, ALBERT (Lan et al., 2020) shows that the NSP objective in BERT lacks difficulty, as the negative examples are created by pairing segments from different documents, which mixes topic prediction and coherence prediction into a single task. Instead of that, ALBERT uses a sentence-order prediction (SOP) objective. SOP

obtains positive examples by taking out two consecutive segments and negative examples by reversing the order of two consecutive segments from the same document. The results show that ALBERT provides better results than BERT in many downstream NLP tasks.

ELECTRA Compared to BERT, ELECTRA (Clark et al., 2020) proposes an effective pre-training method. Instead of corrupting some positions of inputs with [MASK], ELECTRA replaces some tokens of the inputs with their plausible alternatives sampled from a small generator network. ELECTRA trains a discriminator to predict whether the generator replaced each token in the corrupted input or not. The pre-trained discriminator can then be used in downstream tasks for fine-tuning, improving upon the pre-trained representation learned by the generator.ed in downstream tasks for fine-tuning, improving upon the pre-trained representation learned by the generator.

XLNET (Yang et al., 2019b) identifies a key weakness in BERT pre-training. Yang et al. (2019b) show that the symbols such as [MASK] that BERT introduces during pre-training cause a discrepancy between pre-training and fine-tuning as they never occur in real data. Therefore, XLNET proposes a new auto-regressive method based on permutation language modelling (PLM) (Uria et al., 2016) without introducing new symbols.

Upon the introduction, these transformer models are evaluated in many downstream NLP tasks, including STS. However, there is no comprehensive study on STS using the transformers in large and small STS datasets, transfer

learning, data augmentation, multilingual STS, etc., which we do in this chapter.

5.2 Transformer Architecture for STS

The transformer architecture for STS is shown in Figure 5.1. The input of this model is a concatenation of the two sentences, separated by the [SEP] token. Then the output of the [CLS] token is used as the input of a softmax layer that predicts the similarity of the two sentences. We used the mean-squared-error loss as the objective function. For the configurations, we used a batch-size of eight, Adam optimiser with a learning rate $2e^{-5}$, and a linear learning rate warm-up over 10% of the training data. During the training process, the parameters of the transformer and the parameters of the subsequent layers were updated. The models were trained using only training data. Furthermore, they were evaluated while training after every 100 batches, using an evaluation set that had one-fifth of the instances in training data. We performed early stopping if the evaluation loss did not improve over ten evaluation steps. All the models were trained for three epochs. As these transformer models are computationally expensive, we used an Nvidia Tesla T4 GPU for the training process. We have kept these configurations the same for all the experiments to ensure consistency between all the experiments. The implementation is based on PyTorch (Paszke et al., 2019) and HuggingFace (Wolf et al., 2020).

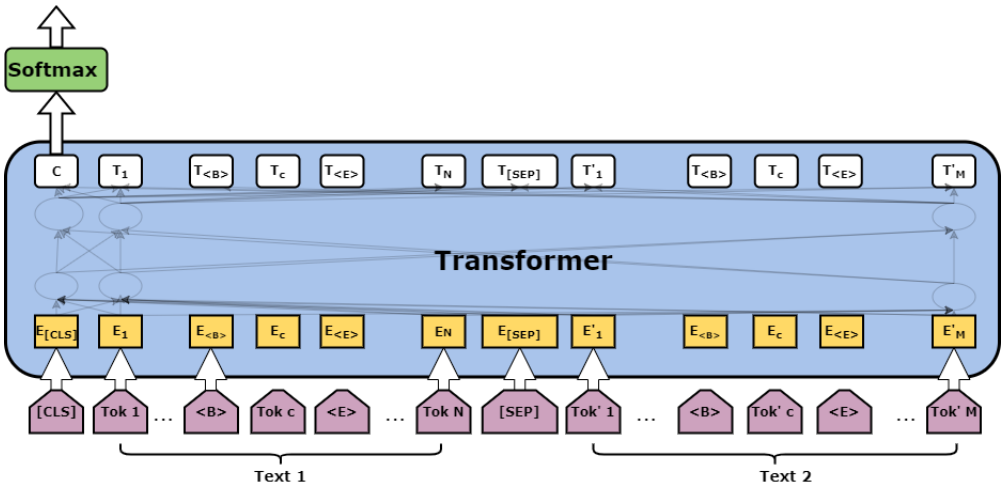


Figure 5.1: Architecture for using Transformers in STS.

5.3 Exploring Transformers in English STS

We evaluated all the transformer variations mentioned before, in three English STS datasets we introduced in 1; SICK, STS 2017 and QUORA. All of the transformer models we experimented have several models that supports English (e.g. *bert-large-cased* & *bert-base-cased* for BERT, *albert-xxlarge-v2* & *albert-base-v2* for ALBERT), and usually the large models outperform the smaller models in downstream tasks. Therefore, we used the largest possible model that our GPU setup can load with each transformer type; *bert-large-cased* for BERT (Devlin et al., 2019), *albert-large-v2* for ALBERT (Lan et al., 2020), *roberta-large* for RoBERTa (Liu et al., 2019), *google/electra-large-discriminator* for ELECTRA (Clark et al., 2020) and *xlnet-large-cased* (Yang et al., 2019b) for XLNET. All of these models are available in HuggingFace (Wolf et al., 2020) model hub[3].

We trained the transformer models on the training sets of these datasets and

[3]Models are available on https://huggingface.co/models

evaluated on the testing sets. The results for the SICK, STS2017 and QUORA are shown in Tables 5.1, 5.2 and 5.3, respectively.

Model	ρ	τ
BERT	0.881	0.826
ALBERT	0.886	0.829
RoBERTa	$0.892^{\dagger}$	$0.834^{\dagger}$
ELECTRA	0.872	0.819
XLNET	0.879	0.821

Table 5.1: Results for SICK dataset with different variants of transformer models. For each variant, Pearson Correlation (ρ) and Spearman Correlation (τ) are reported between the predicted values and the gold labels of the test set. The best result from all of the variations is marked with †.

Model	ρ	τ
BERT	0.889	0.858
ALBERT	0.874	0.852
RoBERTa	$0.895^{\dagger}$	$0.861^{\dagger}$
ELECTRA	0.873	0.849
XLNET	0.868	0.843

Table 5.2: Results for STS 2017 dataset with different variants of Transformers. For each variant, Pearson Correlation (ρ) and Spearman Correlation (τ) are reported between the predicted values and the gold labels of the test set. The best result from all of the variations is marked with †.

As can be seen in Tables 5.1 and 5.2, for the SICK and STS 2017 datasets, RoBERTa outperformed other transformer models. Since SICK and STS 2017 are smaller datasets, the optimised nature of RoBERTa is beneficial. The QUORA dataset, which is larger than the SICK and STS 2017, *XLNET* outperforms other transformer models. However, from the results, there is no clear indication that which transformer model would perform best in a particular dataset other than

Model	RMSE
BERT	0.349
ALBERT	0.354
RoBERTa	0.359
ELECTRA	0.353
XLNET	0.346†

Table 5.3: Results for QUORA dataset with different variants of Transformers. For each variant, Root Mean Squared Error (RMSE) reported between the predicted values and the gold labels of the test set. The best result from all of the variations is marked with †.

the fact that the *RoBERTa* performs slightly better in smaller datasets. It should be noted that all of the transformers perform on par with each other.

In the initial experiments, we noticed that the transformer models are susceptible to the random seed[4] of the experiments (Zhang et al., 2021). Changing the random seed led to different results. To minimise this effect from the random seed, we conducted experiments for five different random seeds. We took the mean of these experiments as the final results, which is the value reported in Tables 5.1, 5.2 and 5.3. We noticed that doing many experiments with different random seeds reduced the variance of the final result.

With this, we answer our **RQ1**: transformers can be successfully adapted in the STS task, and they produce good results in all the datasets. For the smaller datasets, *RoBERTa* performed slightly better than other transformer models. Furthermore, we recommend conducting more experiments with different random seeds to minimise the variance.

[4]A random seed is used to configure the starting weights in a neural network. Keeping the random seed constant in the experiments removes the variation due to this randomness, making it easier to interpret the effects of other design changes such as hyperparameter values.

5.3.1 Impact of Transfer Learning

Similar to the *Siamese neural networks* in Chapter 4, we explored the impact of transfer learning in STS, with transformers. We saved the weights of the transformers models trained on each STS dataset; SICK, STS 2017 and QUORA. We specifically used the two models that performed best in these datasets; RoberTa and XLNET. We again initiated training for each dataset; however, rather than training the transformer models from scratch, we used the weights of the models trained on a different STS dataset. We compared these transfer learning results to the results we got from training the model from scratch. Similar to the *Siamese neural network* experiments, we conducted this transfer learning experiment only on the STS2017 and SICK datasets since the QUORA dataset is already large and transfer learning from a smaller dataset to a larger dataset is nonsensical.

Start Model	STS2017	SICK
$STS2017_{RoBERTa}$	0.895	(+0.009)
$STS2017_{XLNET}$	0.868	(+0.011)
$SICK_{RoBERTa}$	(+0.008)	0.892
$SICK_{XLNET}$	(+0.013)	0.879
$QUORA_{RoBERTa}$	(-0.025)	(-0.021)
$QUORA_{XLNET}$	(-0.039)	(-0.043)

Table 5.4: Results for transfer learning with different Transformers. For each transfer learning experiment we show the difference between with and without transfer learning. Non-grey values are the results of the experiments without transfer learning which we showed in the previous section. For ease of visualisation we only report the Pearson correlation (ρ).

As can be seen in Table 5.4, when we performed transfer learning from

STS2017 ⇒ SICK and SICK ⇒ STS2017 the results improve. This shows that transfer learning can improve the results of transformers. However, similar to the *Siamese neural networks*, when we performed transfer learning from QUORA ⇒ STS2017 and QUORA ⇒ SICK, the results did not improve, in fact, they decrease. Therefore, we can assume that performing transfer learning from a very different dataset, is not beneficial in transformers.

Therefore, we can conclude that transfer learning can improve the results for transformers in STS. However, the transfer learning dataset should be picked carefully, taking the similarity of the two datasets into consideration, rather than only considering the size of the dataset.

5.3.2 Impact of Data Augmentation

Since the transformer models have provided better results with more training data, we experimented with the impact of data augmentation on the transformer models. Similar to the *Siamese neural networks* in Chapter 4, we employed thesaurus-based augmentation in which 10,000 additional training examples are generated by replacing random words with one of their synonyms in Wordnet (Miller, 1995). We specifically used the two models that performed best with the bigger dataset and smaller dataset; *RoBERTa* and *XLNET*. Since using transfer learning improved the results in the previous experiment, we trained the augmented training set on the transferred models; models trained on STS2017 for the SICK experiments and models trained on SICK for the STS2017 experiments. The results are shown in Table 5.5.

Dataset	Start Model	ρ
SICK	$STS2017_{RoBERTa}$	(+0.012)
	$STS2017_{XLNET}$	(+0.011)
STS2017	$SICK_{RoBERTa}$	(+0.014)
	$SICK_{XLNET}$	(+0.013)

Table 5.5: Results for data augmentation with different transformers. For each data augmentation experiment we show the difference between with dat augmentation and without data augmentation. For ease of visualisation we only report the Pearson correlation (ρ).

As can be seen in Table 5.5, data augmentation improved the results of all the experiments with transformers. However, even with the additional 10,000 training instances, *RoBERTa* outperformed *XLNET*. We can conclude that simple data augmentation techniques can improve the performance of transformers in the STS task. From the experiments we conducted, our best results for both the STS2017 and SICK datasets were produced by *RoBERTa* when combined with transfer learning and data augmentation.

These observations answer our *RQ2* in this Chapter; we can use transfer learning and simple data augmentation techniques to improve the results of transformers in STS.

Model	ρ
Jimenez et al. (2014)	0.807
Bjerva et al. (2014)	0.827
Zhao et al. (2014)	0.841
Siamese GRU	0.882
RoBERTa	0.920

Table 5.6: Results for the SICK dataset with different transformer models. For each variant, Pearson Correlation (ρ) is reported between the predicted values and the gold labels of the test set.

Model	ρ
Tian et al. (2017)	0.851
Maharjan et al. (2017)	0.854
Cer et al. (2017)	0.855
Siamese GRU	0.862
RoBERTa	0.915

Table 5.7: Results for the STS2017 dataset with different variants of Siamese Neural Network. For each variant, Pearson Correlation (ρ) is reported between the predicted values and the gold labels of the test set.

Furthermore, we compared the results of the best transformer model with the best results submitted to the competitions (Cer et al., 2017; Marelli et al., 2014), and with the supervised and unsupervised STS methods, we have experimented in this thesis. As can be seen in Tables 5.6 and 5.7, *RoBERTa* outperforms the best system submitted to both competitions. It also outperforms the unsupervised and supervised STS methods we have explored in this thesis, including *Siamese neural networks*. Therefore, we can conclude that transformers are the current state-of-the-art for English STS.

5.4 Portability to Other Languages

Similar to the other STS methods we have experimented with in this thesis, we evaluated transformers in Arabic STS and Spanish STS datasets introduced in Chapter 1. Transformers have the advantage that they do not rely on language-dependent features. As a result, the approach is easily portable to other languages, given the availability of the pre-trained transformer models in that particular language. The transformer models, we used are AraBERT and AraELECTRA for Arabic which were trained on Arabic Wikipedia dump,

the 1.5B words Arabic Corpus (El-khair, 2016), the OSCAR corpus (Ortiz Suárez et al., 2020) and the OSIAN Corpus (Zeroual et al., 2019) using original BERT and ELECTRA architectures explained in Section 5.1[5]. Both of these models use Farasa segmentation as a pre-processing step (Abdelali et al., 2016). For Spanish, we used BETO; a Spanish BERT model (Cañete et al., 2020) trained on Spanish Unannotated Corpora[6] using the original BERT architecture[7]. Additionally, for both languages, we used the "BERT-Base, Multilingual Cased" model (Devlin et al., 2019), which is trained on the top 100 languages with the largest Wikipedias that includes Arabic and Spanish languages.

The results are shown in Tables 5.8 and 5.9 for the Arabic and Spanish datasets. We also compared the results of the transformer models with the best methods submitted to the competition (Cer et al., 2017), and with the supervised/unsupervised STS methods, we have experimented in this thesis.

Model	ρ
Tian et al. (2017)	0.744
Nagoudi et al. (2017)	0.746
Wu et al. (2017)	0.754
Siamese GRU	0.763
mBERT	0.778
AraElectra	0.791
AraBERT	0.794

Table 5.8: Results for the Arabic STS dataset with different transformers. For each variant, Pearson Correlation (ρ) is reported between the predicted values and the gold labels of the test set.

[5]More details about the models and download links are available on https://github.com/aub-mind/arabert

[6]Corpora is available on https://github.com/josecannete/spanish-corpora

[7]More details about BETO is available on https://github.com/dccuchile/beto

Model	ρ
Hassan et al. (2017)	0.848
Wu et al. (2017)	0.850
Tian et al. (2017)	0.855
Siamese GRU	0.863
mBERT	0.884
BETO	0.890

Table 5.9: Results for the Spanish STS dataset with different variants of Siamese Neural Network. For each variant, Pearson Correlation (ρ) is reported between the predicted values and the gold labels of the test set.

As can be seen in the results transformer based STS method outperformed all the other supervised and unsupervised STS models in both languages and outperforms the top systems of the competition in both languages. From the experimented pre-trained transformer models, language specific models like BETO, AraBERT outperformed general multilingual models. Therefore, we can conclude that transformers are currently the state-of-the-art for Arabic and Spanish STS too. Furthermore, it should be noted that it is very easy to adapt transformers in a different language. We only changed the pre-trained model to the new language and performed the training.

This answers our **RQ3:**, the transformers can be successfully adapted in different languages by changing the pre-trained model and the training dataset. They produce state-of-the-art results in STS.

5.5 Potability to Other Domains

To answer our *RQ4*; how well the proposed transformer models can be applied in STS tasks in different domains, we evaluated our method on the Bio-medical STS

dataset explained in Chapter 1 (BIOSSES). As we mentioned previously, the Bio-medical STS dataset does not have a training set. Therefore, we had to follow a transfer learning strategy to evaluate transformers on the Bio-medical STS dataset. Similar to *Siamese neural network* experiments in Chapter 4, we used the pre-trained English STS transformer models and performed inference on the Bio-medical STS dataset.

For this transfer learning strategy, we considered two pre-trained transformer models; *bert-large-cased* (Devlin et al., 2019) (We refer to this as *BERT*) which we used in the English STS experiments and, BioBERT (Lee et al., 2019) which has trained BERT on PubMed abstracts[8].

Data	**BERT**	**BioBERT**
STS2017	0.663	0.763
SICK	0.658	0.751
QUORA	0.612	0.682

Table 5.10: Results for transfer learning with transformers in the BIOSSES dataset. Two considered pre-trained transformer models are textbfBERT and **BioBERT**. For ease of visualisation we only report the Pearson correlation ($\boldsymbol{\rho}$).

As can be seen in the Table 5.10, transformers provided satisfactory results in Bio-medical STS. We got the best result from transformers when trained on STS 2017 using BioBERT. Furthermore, there was a clear improvement when the English STS model was trained using BioBERT rather than general BERT. This may be because most of the Bio-medical words that appear in the BIOSSES dataset are out of vocabulary in the general BERT model, which can cause

[8]More details and the model are available on https://github.com/dmis-lab/biobert

problems to the neural network when it observes them in the testing phase. Furthermore, it should be noted that in this experiment, when we performed transfer learning from the QUORA dataset, the results are lower than when we performed transfer learning from SICK or STS 2017. This again can be due to the reason that the SICK and STS 2017 datasets have a similar annotation strategy to the BIOSSES dataset as discussed in Chapter 1. These results are similar to what we observed with *Siamese neural networks* in Chapter 4.

Model	$\boldsymbol{\rho}$
ELMo $\bigoplus$ *BERT*	0.708
Siamese $GRU_{STS2017}$	0.719
Soğancıoğlu et al. (2017)	0.754
$BioBERT_{STS2017}$	0.763
BioSentVec (Chen et al., 2019)	0.810

Table 5.11: Results for the BIOSSES dataset with transformers compared with top results reported for BIOSSES. For each variant, Pearson Correlation ($\boldsymbol{\rho}$) is reported between the predicted values and the gold labels of the test set.

Furthermore, we compared our results with the best results reported for the dataset. The results are shown in Table 5.11. The best model we had in Table 5.10 which is based on BioBERT when trained on STS2017, is represented as $BioBERT_{STS2017}$. As shown in the results, our method provides satisfactory results when compared to the best approaches. Also, it outperforms the GRU based *Siamese neural network* architecture we experimented in Chapter 4 using the same transfer learning strategy. However, the unsupervised method we experimented with in Chapter 3 with BioSentVec (Chen et al., 2019) comfortably outperformed the transformer-based STS method we experimented with in this

Chapter.

With these observations, we can answer our **RQ4**: *How well do the proposed transformer models perform in a different domain?*. The transformers can be adapted to STS tasks in different domains by changing the pre-trained transformer model. However, without a proper training set, the results are not strong. This is a common observation we had for supervised STS methods in this thesis.

5.6 Recent Developments: Siamese Transformers

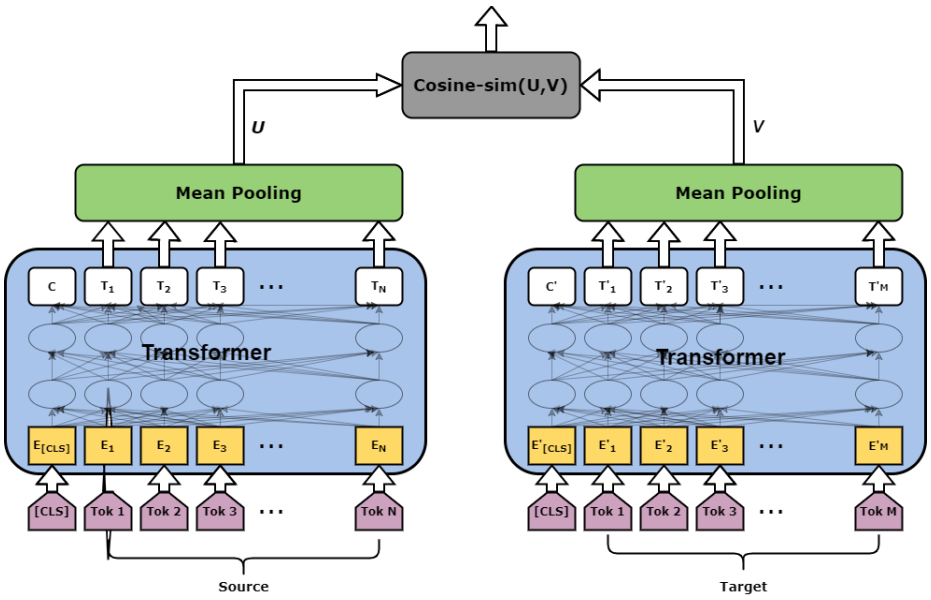


Figure 5.2: Architecture for using Siamese Transformers in STS.

As we observed in previous experiments, transformers are state-of-the-art in supervised STS. However, to predict the similarity in test time, both sentences

must be fed into the network. Two sentences are passed to the transformer network, and the similarity is predicted. However, this setup is unsuitable for various text similarity tasks. For example, finding the sentence pair with the highest similarity in a collection of n = 10 000 sentences requires $n \times \frac{(n-1)}{2} = 49,995,000$ inference computations with the default transformer architecture experimented in this chapter. On a modern V100 GPU, this requires about 65 hours. Similarly, finding the most similar question for a new question from over 40 million existing questions in Quora would require over 50 hours. This massive computational overload is not suitable for many real-world applications (Reimers and Gurevych, 2019).

The obvious solution to this would be to get sentence embeddings from the transformer network. A common approach is to use vector aggregation methods that we already experimented in Chapter 2. However, we previously showed that these simple, unsupervised vector aggregation-based STS methods are outperformed by sentence encoders that use traditional word embeddings. Therefore, Reimers and Gurevych (2019) propose a Siamese neural architecture based on transformers. The architecture is shown in Figure 5.2, which is similar to the Siamese architectures we experimented in Chapter 4.

This architecture can be trained on an STS dataset. Since this is a Siamese architecture, it can produce sentence embeddings that can be used at the inference time without having both sentences in the network. Reimers and Gurevych (2019) show that this architecture provides less accuracy than the default transformer architecture in STS tasks. Yet, the results are very compatible

and outperform other sentence encoders such as Universal Sentence Encoder and Infersent. Furthermore, it outperforms the word embedding based Siamese neural network architectures experimented in Chapter 4. Therefore, this architecture is the current state-of-the-art Siamese neural network in STS tasks. Reimers and Gurevych (2019) calculate that the complexity for finding the most similar sentence pair in a collection of 10,000 sentences is reduced from 65 hours with the default architecture to 5 seconds with the Siamese transformer architecture. We can conclude that this architecture is very efficient in many NLP applications where it is required to find similar sentences from a large set of sentences, such as Translation Memories[9].

5.7 Conclusions

In this chapter, we experimented with utilising state-of-the-art transformers in the STS task. We evaluated the default sentence pair regression architecture on transformers in three English datasets, two non-English datasets and a bio-medical STS dataset. For the smaller STS datasets, we showed that *RoBERTa* outperformed other transformer models. For the larger STS dataset, *XLNET* provided the best result. We showed that we could improve the results with transfer learning and data augmentation techniques. For the three English and two non-English datasets, the transformer-based STS method outperformed all the other supervised and unsupervised STS methods we experimented with in this part of the thesis. Furthermore, they outperformed the best systems

[9]More details and the pre-trained models on Siamese transformers are available on https://www.sbert.net/

submitted for each competition. This shows that the transformers are the current state-of-the-art in supervised STS.

However, in the BIOSSES dataset where it does not have a training set, we used a transfer learning based zero-shot learning when transformers are applied. Even though transformers outperformed other STS methods we experimented such as Siamese neural networks, they could not outperform the sentence vector-based method we experimented with in Chapter 3. We can conclude that although the transformers can be adapted in different domains by changing the pre-trained model, they do not provide strong results without a proper training set.

One major limitation in the transformer-based STS method is that it requires to have both sentences in the network at the inference time, which can cause a massive computational overhead for some NLP applications. To overcome this, Reimers and Gurevych (2019) proposed a Siamese transformer architecture that is capable of proving sentence vectors that can reduce the inference time. Another limitation in the transformers is the pre-trained transformer models are large and can cause problems in real-world applications. As a solution to this, we hope to explore knowledge distillation (Gou et al., 2021) in STS tasks. With knowledge distillation, transformer-based methods can be used as teacher models to train simple student models such as Siamese GRU, which would provide competitive results to transformer-based STS methods but with smaller disk space.

With this, we conclude Part I of the thesis. We explored numerous STS

methods based on word vectors which are easy to adapt in different languages and domains. We evaluated them on different STS datasets and discussed their benefits and limitations in each chapter. We can conclude that transformers are state-of-the-art in supervised STS methods and sentence encoders are state-of-the-art in unsupervised STS methods. Keeping in mind the benefits and limitations of these STS methods, in the following two parts of the thesis, we will employ them in two applications in translation technology; translation memories in Part II and translation quality estimation in Part III.

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" (Simard, 2020). 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 (1979) 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 translators' performance. This storage 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 (Bowker, 2006). Since then, the use of this particular technology has kept growing, and recent studies show that it is used on a regular basis by a large proportion of translators (Zaretskaya et al., 2018).

TM systems help translators by continuously providing them with so-called matches, which are translation proposals retrieved from its database. These

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

matches are identified automatically by comparing the segment that must 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 similar segment to the one to be translated. 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 when it is impossible to identify a fuzzy match (i.e. there is no segment similar enough to the one to be translated).

TMs distinguish between fuzzy 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 the edit distance between two texts can be easily calculated, is fast and is largely language-independent. However, edit distance cannot 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 identify it in most cases. To make this clearer, consider the following three sentences.

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

[2]It is unclear the origin for these values, but translators widely use them. 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.

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.

Assume sentences 2 and 3 already had their translations in the TM database, and now sentence 1 has to be translated. The majority of the commercial TM systems based on edit distance return sentence 2 as a fuzzy match to the incoming sentence since the edit distance between sentences 1 and 2 are lower than sentences 1 and 3. However, sentence 3 is semantically closer to sentence 1 than sentence 2 and does not need many edits in the post-editing process. This nature of the edit distance based TM systems of not providing semantically close matches hinders the translators' efficiency (Ranasinghe et al., 2020a).

Researchers address this shortcoming of the edit distance metric by employing similarity metrics to identify semantically similar segments even when they are different at the token level. Section 6.1 discusses some of the approaches proposed so far. These approaches incorporate simple operations like paraphrasing to the TM matching process to provide semantically similar matches. As we observed in Part I of the thesis, deep learning based architectures are 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 that can 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 easily be adapted in other languages and domains, which is beneficial to TMs as they are employed in a

wide range of domains and languages.

Utilising deep learning in TM matching methods bring obvious challenges regarding efficiency and storage. In Chapter 7, we discuss these challenges and carefully pick STS methods that are efficient in the TM matching process. We evaluate these 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 deep learning based STS methods 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-world TM using English-Spanish pairs.

3. We compare the results of the proposed method with an existing TM system and show that our approach improves the TM matching and retrieving process.

The remainder of this chapter is structured as follows. Section 6.1 discusses the various TM matching algorithms and their shortcomings. 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. The chapter finishes with the conclusions.

6.1 Related Work

As discussed before, even though TM systems have revolutionised the translation industry, these tools are far from being perfect. A serious shortcoming is that most commercial TM systems' (fuzzy) matching algorithm is based on edit distance, and no language processing is employed. Among the first ones to discuss the shortcomings were Macklovitch and Russell (2000) who showed 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 for developing second-generation TM tools, which had some language processing capabilities such as grammatical pattern recognition and performed limited segmentation at the sub-sentence level. However, there are only a few commercially available second-generation TM systems such as *Similis* (Planas, 2005), *Translation Intelligence* (Grönroos and Becks, 2005) and Meta Morpho TM system, *Morphologic* (Hodász and Pohl, 2005). *Similis* (Planas, 2005) performs linguistic analysis 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 to improve matching, especially for morphologically rich languages such as Hungarian (Hodász and Pohl, 2005). Even though the second-generation TM tools solved some of the issues in first-generation TM tools, Mitkov and Corpas (2008) discuss that they

still can not provide strong matches in most of the cases. Mitkov and Corpas (2008) 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 the acquisition of shares*.

To overcome this shortcoming, Pekar and Mitkov (2007) 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 (2007) perform linguistic processing over tree graphs (Szpektor et al., 2004; Knight and Graehl, 2005) followed by lexicosyntactic normalisation. Then similarity between syntactic-semantic tree graphs is computed, and matches at the 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 that performs matching at the level of syntactic trees was proposed by Vanallemeersch and Vandeghinste (2014). The results presented in their paper are preliminary, and the authors notice that the tree matching method is "prohibitively slow".

Further work towards the development of third-generation TM systems included paraphrasing and clause splitting. Raisa Timonera and Mitkov (2015) experimented with clause splitting and paraphrasing, seeking to establish whether these NLP tasks can improve the performance of TM systems in terms

of matching. Furthermore, Gupta et al. (2016b) experimented with incorporating paraphrasing to the TM matching algorithm to secure more matches. The authors sought to embed information from PPDB[3], a database of paraphrases (Ganitkevitch et al., 2013), in the edit distance metric by employing dynamic programming (DP) (Gupta et al., 2016b) as well as dynamic programming and greedy approximation (DPGA) (Gupta et al., 2016a). In more recent work, Gupta et al. (2014a) 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 (Gupta et al., 2014b).

With this analysis, we identified two key limitations in current third-generation TM systems. First, most of them rely on external knowledge bases, including WordNet and PPDB, which are challenging to use in many languages and domains. Secondly, the majority of these approaches are slow to be used in real-world applications. To address these limitations, we propose to use deep learning based STS metrics we experimented in Part I of the thesis in TM matching. As aforementioned, these methods do not depend on external knowledge bases, and most of them are optimised to use effectively in real-world scenarios. Therefore, in Chapter 7 we evaluate these STS metrics in TM matching

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

and retrieval. To the best of our knowledge, this is the first study to employ deep learning in translation memories.

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 Commission's (EC) Directorate General for Translation (DGT) and the EC's Joint Research Centre. DGT-TM contains official legal acts. It consists of sentences and their professional translations covering twenty-two official European Union (EU) languages and their 231 language pair combinations. 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 (Aker et al., 2013; Besacier and Schwartz, 2015).

While we chose English-Spanish sentence pairs for the experiments of this study, our approach is easily extendable to any language pair. In this study, 2018 Volume 1 was used as the experimental translation memory and 2018 Volume 3 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

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

used as input sentences.

6.3 Evaluation

TM systems are typically evaluated by measuring the *quality* of the retrieved segments from the matching algorithm (Gupta et al., 2015b). This *quality* is often considered to be the correspondence between the retrieved segment and the reference translation: *"the closer a retrieved segment is to a reference translation, the better it is"*. First, the quality scores are calculated for individual segments by comparing them with the relevant reference translations. These scores are then averaged over the whole corpus to estimate the quality of the TM system. Such quality evaluation techniques between the retrieved segment and the reference translation are called automatic metrics for machine translation evaluation.

Over the years, researchers have produced many automatic metrics for MT evaluations. BLEU (bilingual evaluation understudy) (Papineni et al., 2002) is 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 (Gupta et al., 2015a). However, using BLEU has drawbacks. The main drawback in BLEU is it does not consider the meaning and does not directly consider sentence structure (Sellam et al., 2020). Since this study aims 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 that was designed to explicitly address several observed weaknesses in BLEU (Banerjee

and Lavie, 2005). Similar to BLEU, METEOR is also based on explicit word-to-word matching. However, unlike BLEU, it not only supports matching between identical words in the two strings 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 (Sellam et al., 2020). These metrics have their own limitations, which can affect the evaluations of this study. Whatever the automatic evaluation metric we use, we would not be able to avoid these weaknesses completely. Therefore, in addition to the automatic evaluation, we 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 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 shown dramatic changes in the translation workflow. One of the essential functions of TM systems is their 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. 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 these limitations, the 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 adaptable 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 with in Part I of the thesis. In addition to providing state-of-the-art STS results, these algorithms are fast and easily adaptable to other languages and domains, which is 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 will also incorporate a human evaluation 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. This matching and retrieving process in most commercial TM systems are still limited to algorithms based on edit distance. 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 (Mitkov and Corpas, 2008). As a result, even if the TM contains a semantically similar segment, the retrieval algorithm will not identify it in most cases. In Chapter 6, we identified this as a major drawback in TMs.

Researchers 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 (Pekar and Mitkov, 2007). 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 multiple times, they can be easily adapted to different languages and domains. Therefore, having a deep learning based STS metric would benefit TMs in many ways (Ranasinghe et al., 2021a). In this

chapter, we will 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, the transformers have set a new state-of-the-art performance on semantic textual similarity. However, to predict the similarity in test time, both sentences must be fed into the transformer network, which causes a massive computational overhead (Reimers and Gurevych, 2019). Finding the most similar sentence to the incoming sentence in a collection of 100,000 sentences take 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 STS solutions.

As discussed in Part I of the thesis, the next best STS method we experimented with was Siamese architectures 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 the 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 the 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 (Reimers and Gurevych, 2019) 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 (Conneau et al., 2017) and

Universal Sentence Encoder (Cer et al., 2018). As far as we know, this is 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 tasks?

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 in the 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 sentence encoders outperform Okapi in certain fuzzy match ranges.

3. We further perform a detailed human evaluation of 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 than Okapi.

[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.

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

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 these 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 (Liu et al., 2019).

With the SICK dataset, edit distance achieve only 0.361 Pearson correlation

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

while Infersent, Universal Sentence Encoder and SBERT achieve 0.769, 0.780 and 0.892 Pearson correlation, respectively. Similarly, with the STS 2017 dataset, sentence encoders outperform edit distance by a large margin. This is a clear indication that the sentence encoders can calculate the text similarity better than edit distance. 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 the same. The 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 mentioned before; Infersent, Universal Sentence Encoder and SBERT. We used the same pre-trained models mentioned 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 decentralised 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 three fields: source segment, target segment and the 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. We retrieve the embedding with the highest cosine similarity and retrieve the corresponding target segment for the embedding as the translation memory match. We used the *'getNearest'* functionality provided by AquilaDB for this step. This step is visualised in Figure 7.1.

The efficiency of the TM matching and retrieval is a key factor for the

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

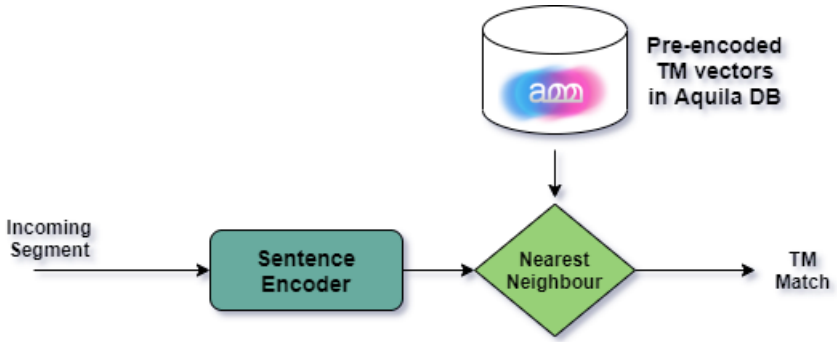


Figure 7.1: TM matching process for an incoming segment.

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.

translators using them. As discussed in Chapter 6, most of the proposed third-generation TM systems were not efficient enough to be used in real-world scenarios. This was the reason why they are not popular in the community. We wanted to avoid making the same mistake with our proposed approach to make it more useful for the community. Therefore, as the first step, we calculated the efficiency of the proposed method.

Table 7.2, discusses the efficiency of each sentence encoder. The experiments were done in an 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 a CPU (Central Processing Unit) since the translators using translation memory tools would not have access to a GPU on a daily basis.

The translation memory was processed in batches of 256 sentences to obtain 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 Table 7.2 reports the time needed for each sentence 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 the translators translate sentences one by one. Interestingly, while the Universal Sentence Encoder was the most efficient in generating sentence embeddings in batches, it was the least efficient encoder to derive the embedding for a single sentence where it took 1.23 seconds to do so. InferSent was the fastest sentence encoder for a single sentence.

The third column of Table 7.2 reports the time needed to retrieve the best match from the translation memory. This includes the time taken to compute the cosine similarity between the embeddings of TM sentences and the embeddings of the input sentence. It also consists of the time to sort the similarities, get the index of the highest similarity, and retrieve the TM sentence considered a match for the input sentence. As shown in Table 7.2, all sentence encoders needed approximately 0.5 seconds 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 regarding efficiency.

7.3 Results and Evaluation

In this section, we report the results of the three selected sentence encoders in TM matching. 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. To measure 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.

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

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. The best result for each partition is shown in bold.

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 the TM matching and retrieval task. The first partition contained the matches derived from Okapi with 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 in 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 the 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 the 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 improving the translator's productivity. According to the annotation guidelines of STS tasks which we explained in Chapter 1, an STS score of 0.8 means *"The two sentences are mostly equivalent, but some unimportant details differ"* and an STS 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 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 is very efficient.

7.4 Error Analysis

As mentioned in Chapter 6, automatic machine translation evaluation metrics such as METEOR are far from being perfect. As METEOR relies largely on string overlap, it cannot properly capture the semantic equivalence of the segments retrieved using the sentence encoders. Therefore, a human evaluation is required for this study. In this section, we carried out a human evaluation in the form of 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 that contained information that 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 that 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 that 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 can retrieve a segment that conveys the meaning correctly but has the incorrect date (7). From the point of view of the effort required to produce an 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 only a part of the English sentences. Therefore, the match suggested is only partially correct. As an illustration, for segment (8), Okapi retrieved (9) as a match. 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 (10) is correct. This pattern can also be seen when comparing Okapi with SBERT. For example, while the proposed match for (11) by SBERT is correct (12), Okapi only recognises one word of the segment, as the retrieved match 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 Okapi's 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. (Meat of bovine animals, frozen.)

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. Okapi retrieves the translation of the multiword expression (23) 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 several 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).

(33) None of the above

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

(35) Ninguna (None)

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 (36) is (37) instead of (38); SBERT retrieves (39) when the original segment is (40), and the proposal by InferSent for (41) is (42).

(36) Japan

(37) Israel

(38) Japón

(39) Singapur (Singapore)

(40) Philippines

(41) within municipality of Sitovo

(42) 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 original English segment. The match InferSent returns for (43) is (44), and in this case, Okapi retrieves a correct translation (45). On another isolated occasion, SBERT also adds a redundant word "mixto" (joint) by proposing (46) as the translation for (47). In this particular instance, the retrieved match by Okapi is correct (48).

(43) Requirements

(44) Requisitos del Eurosistema (Eurosystem requirements)

(45) Requisitos

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

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

(48) El Comité dispondrá su reglamento interno

With this analysis, it is clear that sentence encoders provide better matches than Okapi in most 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 of 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 evaluated 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 the sentence encoders return better matches than simple edit distance for sentences with a fuzzy match score less than 0.8 in Okapi. Of the sentence encoders, InferSent fares best. We also present an error analysis, where three native Spanish speakers analysed the matches proposed by the sentence encoders and Okapi. This error analysis further confirms that the sentence encoders can be used to improve the matching and retrieval process in TMs (Ranasinghe et al., 2020a).

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 a 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 make the cosine similarity calculation between vectors a computationally less intensive process. Faster algorithms generating sentence embeddings such as 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 limitations that might have affected the evaluation of this study. Very recently, new automatic MT evaluation metrics such as BLEURT (Sellam et al., 2020) 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 this study. Therefore, as future work, we will incorporate these new automatic MT evaluation metrics.

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 the 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 Part II of the thesis would pave a new direction in third-generation TM systems.

Part III

Applications - Translation Quality Estimation

Chapter 8

Introduction to Translation Quality Estimation

The goal of quality estimation (QE) is to evaluate the quality of a translation without having access to a reference translation (Specia et al., 2018). High-accuracy QE that can be easily deployed for a number of language pairs is the missing piece in many commercial translation workflows as they have numerous potential uses. They can be employed to select the best translation when several translation engines are available or can inform the end-user about the reliability of automatically translated content. In addition, QE systems can be used to decide whether a translation can be published as it is in a given context, or whether it requires human post-editing before publishing or even translation from scratch by a human (Kepler et al., 2019).

Quality estimation task is different from automatic machine translation evaluation (Barrault et al., 2020). Automatic machine translation evaluation approaches such as BLEU (Papineni et al., 2002) need the reference translation in order to perform the machine translation evaluation. As mentioned before, quality estimation, on the other hand, does not need the reference translation. Therefore, automatic MT evaluation approaches such as BLEU (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005), LEPOR (Han et al., 2012) and others

can not be directly applied in the QE task. As a result, the solutions proposed for QE are completely different from that of automatic machine translation evaluation.

The estimation of translation quality can be done at different levels: word/phrase-level sentence-level and sentence-level (Ive et al., 2018). Word-level QE aims to spot words that need to be reviewed during the post-editing process. It indicates which words from the source have been incorrectly translated in the target and whether the words inserted between these words are correct. Sentence-level QE models provide a single score for each pair of source and target sentences. Sentence-level QE scores help to rank translations that are worth post-editing. Document-level QE, on the other hand, scores or ranks documents according to their quality for fully automated MT usage scenarios (Ive et al., 2018). In recent years, word-level QE and sentence-level QE have been more popular among the community (Specia et al., 2018).

During the past decade, there has been tremendous progress in the field of quality estimation, largely as a result of the QE shared tasks organised annually by the Workshops on Statistical Machine Translation (WMT), more recently called as the Conferences on Machine Translation, since 2012 (Callison-Burch et al., 2012; Bojar et al., 2013; Bojar et al., 2014; Bojar et al., 2015; Bojar et al., 2016; Bojar et al., 2017; Specia et al., 2018; Fonseca et al., 2019; Specia et al., 2020). First, they have provided annotated datasets which can be used to train QE models and to evaluate them. Second, The annotated datasets these shared tasks released each year have led to the development of many open-source QE

systems (Specia et al., 2015; Ive et al., 2018; Kepler et al., 2019).

At present neural-based QE methods constitute state-of-the-art in quality estimation. However, these approaches are based on complex neural networks and require resource-intensive training. This resource-intensive nature of these deep learning based frameworks makes it expensive to train QE models. Furthermore, these architectures require a large number of annotated instances for training, making the quality estimation task very difficult for the low-resource language pairs. This nature of the current state-of-the-art QE systems has hindered their popularity in real-world applications.

This part of the thesis addresses these problems by employing simple STS architectures we experimented with in Part I of the thesis in quality estimation. We redefine the QE task as a cross-lingual STS task and show that state-of-the-art STS architectures can be applied in the QE task by changing the input embeddings. As far as we know, this is the first study done on applying neural STS models directly to the QE task.

In Chapter 9, we explore sentence-level QE with STS architectures. We evaluate their performance in recent sentence-level QE datasets, comparing them with open-source QE tools such as OpenKiwi (Kepler et al., 2019) and QuEst++ (Specia et al., 2015). Finally, we propose a new state-of-the-art QE method for sentence-level QE.

In Chapter 10, we expand this idea to word-level QE. We modify the output of the STS architecture to predict word-level translation qualities. We evaluated their performance in recent word-level QE datasets, comparing the results with

open-source word-level QE tools such as OpenKiwi (Kepler et al., 2019) and Marmot (Logacheva et al., 2016). We show that in most of the datasets, our simple word-level architecture outperforms other QE tools.

In Chapter 11, for the first time, we explore multilingual QE with state-of-the-art word-level and sentence-level QE methods. Furthermore, we evaluate multilingual QE in different training environments, including zero-shot and few-shot. Our findings in Chapter 11 would be beneficial for low resource languages.

The main contributions of this part of the thesis are as follows.

1. We propose two STS architectures based on transformers to perform sentence-level QE. These architectures are simpler than the architectures available in OpenKiwi (Kepler et al., 2019) and DeepQuest (Ive et al., 2018). We evaluate them on 15 language pairs in which sentence-level QE data was available, and we show that the two architectures outperform the current state-of-the-art sentence-level QE frameworks such as OpenKiwi (Kepler et al., 2019) and DeepQuest (Ive et al., 2018).

2. We introduce a simple architecture to perform word-level quality estimation that predicts the quality of the words in the source sentence, target sentence and the gaps in the target sentence. We evaluate it on eight different language pairs in which the word-level QE data was available, and we show that the proposed architecture outperforms the current state-of-the-art word-level QE frameworks like Marmot (Logacheva et al., 2016) and OpenKiwi (Kepler et al., 2019).

3. We propose multilingual learning for QE with the proposed architectures for sentence-level and word-level. We show that multilingual models are helpful in low-resource languages where the training data is difficult to find.

4. We provide important resources to the community. The code of each chapter is bundled to an open-source QE framework, and the pre-trained sentence-level and word-level QE models will be freely available to the community. The link to the relevant code and the models will be unveiled in the introduction section of each chapter.

The remainder of this chapter is structured as follows. Section 8.1 explores the various methods that have been employed in sentence-level and word-level QE, including previous research done incorporating STS in the QE task. Section 8.2 discuss the various datasets we used in this part of the thesis. In Section 8.3, we show the main evaluation metrics used for the sentence-level and word-level QE experiments in the following Chapters in Part III of the thesis. The chapter finishes with the conclusions.

8.1 Related Work

Before the neural network era, most of the quality estimation systems such as QuEst (Specia et al., 2013) and QuEst++ (Specia et al., 2015) were heavily dependent on linguistic processing and feature engineering to train traditional machine-learning algorithms including support vector regression and

randomised decision trees (Specia et al., 2013). These features can be either extracted from the machine translation system (*glass-box* features) or obtained from the source and translated sentences, as well as external resources, such as monolingual or parallel corpora (*black-box* features) (Specia et al., 2009). For example, QuEst (Specia et al., 2013) has 17 manually crafted features fed in to a support vector regression algorithm. These 17 features consist of glass-box features such as *ratio of number of tokens in source and target segments*, *ratio of percentage of nouns/verbs in the source and target* etc. as well as black-box features such as *global score and relevant features of the SMT system*, *proportion of pruned search graph nodes* etc. In QuEst++, the number of features varies from 80 to 123 depending on the language pair (Specia et al., 2015). QuEst (Specia et al., 2013), QuEst++ (Specia et al., 2015) and Marmot (Logacheva et al., 2016) can be considered as the most popular traditional QE tools. QuEst (Specia et al., 2013) only supports sentence-level QE, Marmot (Logacheva et al., 2016) only supports word-level QE while QuEst++ (Specia et al., 2015) can support both word-level and sentence-level QE. Even though, they provided good results in early days, these traditional approaches are no longer the state-of-the-art. In recent years, neural-based QE systems have consistently topped the leader boards in WMT quality estimation shared tasks (Kepler et al., 2019).

With the increasing popularity of word embeddings (Mikolov et al., 2013a), neural networks based on word embeddings got popular in the QE field. They outperformed traditional QE systems and provided state-of-the-art results. For example, the best-performing system at the WMT 2017 shared task on

QE was POSTECH, which is purely neural and does not rely on feature engineering at all (Kim et al., 2017). POSTECH revolves around an encoder-decoder Recurrent Neural Network (RNN) (referred to as the 'predictor'), stacked with a bidirectional RNN (the 'estimator') that produces quality estimates. In the predictor, an encoder-decoder RNN model predicts words based on their context representations and in the estimator step, there is a bidirectional RNN model to produce quality estimates for words, phrases and sentences based on representations from the predictor. To be effective, POSTECH requires extensive predictor pre-training, which means it depends on large parallel data and is computationally intensive (Ive et al., 2018). The POSTECH architecture was later re-implemented in DeepQuest (Ive et al., 2018). DeepQuest supports both word-level and sentence-level QE (Ive et al., 2018).

OpenKiwi (Kepler et al., 2019) is another open-source neural QE framework developed by Unbabel. It implements four different neural network architectures QUETCH (Kreutzer et al., 2015), NuQE (Martins et al., 2016), Predictor-Estimator (Kim et al., 2017) and a stacked model of these architectures. Both the QUETCH and NuQE architectures have simple neural network models that do not rely on additional parallel data but do not perform that well. The Predictor-Estimator model is similar to the POSTECH architecture and relies on additional parallel data. In OpenKiwi, the best performance for sentence-level quality estimation was given by the stacked model that used the Predictor-Estimator model, meaning that the best model requires extensive predictor pre-training and relies on large parallel data and computational resources.

As discussed before, these models' complex and resource-intensive nature creates many limitations when deployed in real-world scenarios. Therefore, in this study, we propose to use simple STS architectures in QE. Over the years, there have been a few attempts to integrate semantic similarity into QE. Specia et al. (2011) employed semantic information into the QE task to address the problem of meaning preservation in translation. The authors integrated semantic similarity features to the QE model and improved the results of the QE task (Specia et al., 2011). Biçici and Way (2014) introduced the use of referential translation machines (RTM) for QE. RTM is a computational model for judging monolingual and bilingual similarity that achieves state-of-the-art results. This approach provided the best result in both sentence-level and word-level tasks in WMT 2013 (Bojar et al., 2013). Furthermore, Kaljahi et al. (2014) and Souza et al. (2014) used syntactic and semantic information in quality estimation and were able to improve over the baseline when combining these features with the features of the baseline. Finally, in a different approach, Bechara et al. (2016) used semantically similar sentences and their quality scores as features to estimate the quality of machine translated sentences. This method improved the prediction of machine translation quality for semantically similar sentences (Bechara et al., 2016).

Even though there are several studies done to integrate semantic similarity into QE, as far as we know, this is the first study to employ state-of-the-art neural STS models directly in the QE task.

8.2 Datasets

All the datasets that we used in this part of the thesis are publicly available and were released in WMT quality estimation tasks in recent years (Specia et al., 2018; Fonseca et al., 2019; Specia et al., 2020). This was done to ensure the replicability of our experiments and to allow us to compare our results with state-of-the-art methods. The following sections will describe the sentence-level and word-level QE datasets we experimented, separately.

8.2.1 Sentence-level QE

Sentence-level QE datasets that we used in this part of the thesis can be categorised into two main areas depending on the aspect they have been annotated; Human-mediated Translation Edit Rate (HTER) and Direct Assessment (DA). Most of the early datasets have been annotated on HTER. Very recently DA aspect is getting popular in the QE community. We describe each of them in detail in the following sections.

Predicting HTER The performance of sentence-level QE systems has typically been assessed using the semiautomatic HTER. HTER is an edit-distance-based measure that captures the distance between the automatic translation and a reference translation in terms of the number of modifications required to transform one into another. In light of this, a QE system should be able to predict the percentage of edits required in the translation. We used several language pairs for which HTER information was available: English-Chinese (En-

Language Pair	Source	MT system	Competition	train, dev, test size
De-En	Pharmaceutical	Phrase-based SMT	WMT 2018 (Specia et al., 2018)	25,963, 1,000, 1,000
En-Zh	Wiki	fairseq based NMT	WMT 2020 (Specia et al., 2020)	7,000, 1,000, 1,000
En-Cs	IT	Phrase-based SMT	WMT 2018 (Specia et al., 2018)	40,254, 1,000, 1,000
En-De	Wiki	fairseq based NMT	WMT 2020 (Specia et al., 2020)	7,000, 1,000, 1,000
En-De	IT	Phrase-based SMT	WMT 2018 (Specia et al., 2018)	26,273, 1,000, 1,000
En-Ru	Tech	Online NMT	WMT 2019 (Fonseca et al., 2019)	15,089, 1,000, 1,000
En-Lv	Pharmaceutical	Attention-based NMT	WMT 2018 (Specia et al., 2018)	12,936, 1,000, 1,000
En-Lv	Pharmaceutical	Phrase-based SMT	WMT 2018 (Specia et al., 2018)	11,251, 1,000, 1,000

Table 8.1: Information about language pairs used to predict HTER. The **Language Pair** column shows the language pairs we used in ISO 639-1 codes[1]. **Source** expresses the domain of the sentence and **MT system** is the Machine Translation system used to translate the sentences. In that column NMT indicates Neural Machine Translation and SMT indicates Statistical Machine Translation. **Competition** shows the quality estimation competition in which the data was released and the last column indicates the number of instances the train, development and test dataset had in each language pair respectively.

Zh), English-Czech (En-Cs), English-German (En-De), English-Russian (En-Ru), English-Latvian (En-Lv) and German-English (De-En). The texts are from a variety of domains, and the translations were produced using both neural and statistical machine translation systems. More details about these datasets can be found in Table 8.1 and in (Specia et al., 2018; Fonseca et al., 2019; Specia et al., 2020). Several examples from WMT 2020, En-De dataset is shown in Table 8.2.

Predicting DA Even though HTER has been typically used to assess quality in machine translations, the reliability of this metric for evaluating the performance of quality estimation systems has been questioned by researchers (Graham et al., 2016). The current practice in MT evaluation is the so-called Direct Assessment (DA) of MT quality (Graham et al., 2017), where raters evaluate the machine translation on a continuous 1-100 scale. This method has been shown to improve

[1]Language codes are available in ISO 639-1 Registration Authority Website Online - https://www.loc.gov/standards/iso639-2/php/code_list.php

Source	Target	HTER
José Ortega y Gasset visited Husserl at Freiburg in 1934 .	1934 besuchte José Ortega y Gasset Husserl in Freiburg .	0.3333
however , a disappointing ninth in China meant that he dropped back to sixth in the standings .	eine enttäuschende Neunte in China bedeutete jedoch , dass er in der Gesamtwertung auf den sechsten Platz zurückfiel .	0.2000
" Renaissance Humanism and the Future of the Humanities . "	" Renaissance Humanism and the Future of the Humanities " .	1.0000
sophomore Jacory Harris directed the newly implemented offense .	Sophomore Jacory Harris leitete die neu umgesetzte Straftat .	0.0000

Table 8.2: Examples source/target pairs from WMT 2020 En-De HTER dataset (Specia et al., 2020). **Source** column shows the source sentence in English and **Target** shows the target sentence in German. **HTER** column shows the annotated HTER value for the translation.

the reproducibility of manual evaluation and to provide a more reliable gold standard for automatic evaluation metrics (Graham et al., 2015).

We used a recently created dataset to predict DA in machine translations which was released for the WMT 2020 quality estimation shared task 1 (Specia et al., 2020). The dataset is composed of data extracted from Wikipedia for six language pairs, consisting of high-resource English-German (En-De) and English-Chinese (En-Zh), medium-resource Romanian-English (Ro-En) and Estonian-English (Et-En), and low-resource Sinhala-English (Si-En) and Nepalese-English (Ne-En), as well as a Russian-English (En-Ru) dataset which combines articles from Wikipedia and Reddit (Fomicheva et al., 2020b). These datasets have been collected by translating sentences sampled from source-language articles using state-of-the-art NMT models built using the fairseq toolkit (Ott et al., 2019) and annotated with DA scores by professional translators.

Source	Target	DA
Tratatul de Aderare fusese semnat la 16 aprilie 2003.	The Accession Treaty had been signed on 16 April 2003.	0.6226
Are o industrie turistică bine dezvoltată, fiind o destinație populară printre turiștii britanici și germani.	It has a well-developed tourist industry as a popular destination among British and German tourists.	0.9386
Numărul mare de avioane de vânătoare trimise în misiuni împotriva avioanelor de recunoaștere nu a fost o greșeală.	There was no mistake in the large number of hunting aeroplanes deployed in missions against recognition aeroplanes.	0.0024
Ar fi fost inutil să încerce obținerea unei o catedre universitare.	It would have been pointless to try to get a university catalogue.	0.5456

Table 8.3: Examples source/target pairs from WMT 2020 Ro-En DA dataset (Specia et al., 2020). **Source** column shows the source sentence in English and **Target** shows the target sentence in German. **DA** column shows the annotated DA value for the translation.

Each translation was rated with a score from 0-100 according to the perceived translation quality by at least three translators (Specia et al., 2020). The DA scores were standardised using the z-score. The quality estimation systems evaluated on these datasets have to predict the mean DA z-scores of test sentence pairs. Each language pair has 7,000 sentence pairs in the training set, 1,000 sentence pairs in the development set and another 1,000 sentence pairs in the testing set.

8.2.2 Word-level QE

Word-level QE annotations are not straightforward as sentence-level QE annotations. They have been annotated for words in the target ('OK' for correct words, 'BAD' for incorrect words), gaps in the target ('OK' for genuine gaps, 'BAD' for gaps indicating missing words) and source words ('BAD' for words

that lead to errors in the target, 'OK' for other words) (Specia et al., 2018). To make it clearer, consider the following example from En-De in WMT 2018 (Specia et al., 2018). The word-level quality estimation labels would be followed. Please note that *green* represents 'OK' and *red* represents 'BAD' labels.

Source - *for example , you could create a document containing a car that moves across the Stage .*

Target - *Sie können beispielsweise ein Dokument erstellen , das ein Auto über die Bühne enthält .*

Source - *for example , you could create a document containing a car that moves across the Stage .*

Target - *<GAP> Sie <GAP> können <GAP> beispielsweise <GAP> ein <GAP> Dokument <GAP> erstellen <GAP> , <GAP> das <GAP> ein <GAP> Auto <GAP> über <GAP> die <GAP> Bühne <GAP> enthält <GAP> . <GAP>*

As can be seen in the example, all the words in the source, all the words in the target and all the *gaps* in the target have been annotated as 'OK' or 'BAD'. Most of the language pairs that are annotated for sentence-level HTER scores also have word-level quality labels. Therefore, for the word-level QE experiments, we used the same language pairs we used for sentence-level HTER, which are shown in Table 8.1. More information about the word-level annotations are available on (Specia et al., 2018; Fonseca et al., 2019; Specia et al., 2020).

8.3 Evaluation

For the evaluation, we used the same approach proposed in the WMT shared tasks so that we can compare our results with the respective baselines and the best systems submitted in each shared task. Obviously, the sentence-level and word-level QE follow two different evaluation criteria, which we explain in the following sections.

Sentence-Level QE Evaluation : Similar to the STS evaluation in Part I of the thesis, Pearson's Correlation Coefficient (ρ) is the most popular evaluation metric in recent WMT sentence-level QE shared tasks (Specia et al., 2018; Fonseca et al., 2019; Specia et al., 2020). Since the gold labels are continuous in both HTER and DA and the models need to predict a continuous value, it makes sense to employ Pearson's Correlation Coefficient as the evaluation metric. A QE model with a Pearson's Correlation Coefficient close to 1 indicates that the predictions of that model and gold labels have a strong positive linear correlation, and therefore, it is a good model to predict sentence-level QE.

Word-level QE Evaluation : The primary evaluation metric for word-level QE is the multiplication of F1-scores for the OK and BAD classes, denoted as $F1_{MULTI}$ (Specia et al., 2018; Fonseca et al., 2019; Specia et al., 2020). The standard equation for the F1 score is shown in Equation 8.1 where TP, TN, FP and FN are True Positive, True Negative, False Positive and False Negative, respectively. This F1 score is calculated for OK and BAD classes separately, and then, they are

multiplied to get the $F1_{MULTI}$ as shown in Equation 8.2.

$$F1 = \frac{2 * TP}{2 * TP + FP + FN} \tag{8.1}$$

$$F1_{MULTI} = F1_{OK} \times F1_{BAD} \tag{8.2}$$

Prior to WMT 2019, $F1_{MULTI}$ score was calculated separately for words in the source ($F1_{MULTI}$ $Source$), words in the target ($F1_{MULTI}$ $Target$) and gaps in the target ($F1_{MULTI}$ $GAPS$) (Specia et al., 2018), while after WMT 2019 (Fonseca et al., 2019; Specia et al., 2020) they produce a single result for target gaps and words named as "$F1_{MULTI}$ $Target$" alongside with "$F1_{MULTI}$ $Source$". We follow the same approach. A QE model with a $F1_{MULTI}$ score close to 1 indicates that the predictions of that model and gold labels are similar and therefore, it is a good model to predict word-level QE.

8.4 Conclusion

Quality estimation is an important component in making machine translation useful in real-world applications. It aims to inform the user of the quality of the MT output at test time. This process can be done on different levels such as word-level, sentence-level and document-level. QE shared tasks organised annually by WMT have increased QE's popularity among the MT community by leading the development of standard datasets covering a variety of languages and domains. These QE shared tasks have further contributed to the development

of evaluation measures in QE. We followed the same evaluation measures; Pearson's Correlation Coefficient for sentence-level and $F1_{MULTI}$ for word-level.

The annotated datasets these shared tasks released each year have led to the development of many open-source QE systems. Similar to other NLP fields, most of the early QE approaches including QuEst (Specia et al., 2013), QuEst++ (Specia et al., 2015) and Marmot (Logacheva et al., 2016) were also based on traditional machine learning and involved heavy feature engineering. However, they no longer provide competitive results. The current state-of-the-art in QE is neural models. Following this, many open-source neural QE frameworks such as OpenKiwi (Kepler et al., 2019) and DeepQuest (Ive et al., 2018) have been created. However, these neural QE architectures are complex and need a lot of computing resources to train a QE model, which we have identified as a major limitation. To address this weakness, we propose to redefine the QE task as a cross-lingual STS task and apply the STS architectures we experimented in Part I of the thesis to QE, which are considerably simpler than the current state-of-the-art QE models.

In the next few chapters, we will be exploring the STS architectures in the QE task. First, in Chapter 9, the STS architectures will be applied in sentence-level QE, and then in Chapter 10, these architectures will be extended to word-level QE. Finally, in Chapter 11, we explore multilingual QE with the proposed architectures.

Chapter 9

TransQuest: STS Architectures for QE

Neural-based QE methods constitute state-of-the-art in quality estimation. This is clear as, in recent years, neural-based QE systems have consistently topped the leaderboards in WMT quality estimation shared tasks (Kepler et al., 2019). For example, the best-performing system at the WMT 2017 shared task on QE was POSTECH, which is purely neural and does not rely on feature engineering at all (Kim et al., 2017). As a result, most of the recent open-source QE frameworks such as OpenKiwi (Kepler et al., 2019) and DeepQuest (Ive et al., 2018) rely on deep learning.

However, despite providing state-of-the-art results in QE, these neural frameworks have a common drawback. They are very complex and need a lot of computing resources. For example, the best performing sentence-level neural architecture in OpenKiwi (Kepler et al., 2019) is the stacked model that used the Predictor-Estimator model we described in Chapter 8 which requires extensive predictor pre-training and relies on a large parallel dataset and computational resources. This is similar to the POSTECH architecture in DeepQuest (Ive et al., 2018) . This complex nature of the state-of-the-art QE frameworks has hindered their popularity in real-life applications.

To overcome this, we propose to remodel the QE task as a cross-lingual STS task. In other words, measuring the quality between a source and a target can be interpreted as computing the cross-lingual textual similarity between the source and the target. With this definition, we can use the STS neural architectures we explored in Part I of the thesis in the QE task. However, since we used monolingual embeddings for STS, we need to change them so that the embeddings can represent both languages in the QE task. For that, we propose to use cross-lingual embeddings. We assume that using the cross-lingual embeddings in the same vector space would ease the learning process for the proposed neural network architectures.

Recalling from Part I of the thesis, state-of-the-art supervised STS methods rely on transformer models. These architectures are considerably simpler than state-of-the-art QE architectures in OpenKiwi (Kepler et al., 2019) and DeepQuest (Ive et al., 2018). Furthermore, cross-lingual embeddings that we intend to use with the STS architectures are fine-tuned to reflect the properties between source and target languages. As a result, this removes the dependency on large parallel data, which in turn means there is no longer need for powerful computational resources. These reasons motivated us to explore STS architectures in the QE task. We believe that simpler and efficient architectures will improve the popularity of QE in real-life applications. As far as we know, this would be the first work to apply STS architectures directly in the QE task.

We address three research questions in this chapter:

RQ1: Can existing state-of-the-art STS architecture be used in sentence-level

QE tasks by modifying the input embeddings?

RQ2: Do cross-lingual embeddings have an advantage over multilingual embeddings in the QE task?

RQ3: Can the models be further improved by data augmentation and ensemble learning?

The main contributions of this chapter are as follows.

1. We propose two architectures based on Transformers to perform sentence-level QE. These architectures are simpler than the architectures available in OpenKiwi and DeepQuest (Lee, 2020; Wang et al., 2018).

2. We evaluate them on both aspects of sentence-level QE on 15 language pairs, and we show that the two architectures outperform the current state-of-the-art sentence-level QE frameworks like DeepQuest (Ive et al., 2018) and OpenKiwi (Kepler et al., 2019).

3. We suggest further improvements to the models by data augmentation and ensemble learning.

4. The two architectures introduced here were released as part of a QE framework; *TransQuest*. TransQuest was independently evaluated in the WMT 2020 QE shared task 1[1] (Specia et al., 2020) and won it in all the language pairs outperforming 50 teams around the globe (Ranasinghe et al., 2020b).

[1]The shared task is available on http://statmt.org/wmt20/quality-estimation-task.html

5. The code and the pre-trained models of *TransQuest* are publicly available to the community[2]. We have published *TransQuest* as a python library[3], and by the time of writing this chapter, it has more than 9,000 downloads from the community[4].

The rest of this chapter is organised as follows. Section 9.1 discusses the methodology and the experiments conducted with 15 language pairs in both aspects of sentence-level QE. Section 9.2 shows the results and Section 9.3 provides further fine-tuning strategies to improve the results. In Section 9.4, we provide a basic error analysis. The chapter finishes with conclusions and ideas for future research directions in QE.

9.1 Methodology

As mentioned before, we considered sentence-level QE as a cross-lingual STS task. Therefore, we used the two best STS architectures we had in part I of the thesis. As we mentioned in Part I, these architectures were based on Transformers, the default sentence pair classification architecture in Transformers and the Siamese Transformer model described in Chapter 5. However, rather than using monolingual Transformers as in STS, we used cross-lingual Transformers for the QE task to represent both languages in the same vector space.

[2]The public GitHub repository is available on https://github.com/tharindudr/TransQuest. The pre-trained QE models for more than 15 language pairs are available on HuggingFace model repository on https://huggingface.co/TransQuest

[3]The developed python library is available on https://pypi.org/project/transquest/

[4]For the latest statistics, please visit https://pepy.tech/project/transquest

Although there are several multilingual transformer models including multilingual BERT (mBERT) (Devlin et al., 2019) and multilingual DistilBERT (mDistilBERT) (Sanh et al., 2019), researchers expressed some reservations about their ability to represent all the languages (Pires et al., 2019). In addition, although mBERT and mDistilBERT showed some cross-lingual characteristics, they do not perform well on cross-lingual benchmarks (K et al., 2020) as they have not been trained using cross-lingual data.

XLM-RoBERTa (XML-R) was released in November 2019 (Conneau et al., 2020) as an update to the XLM-100 model (Conneau and Lample, 2019). XLM-R takes a step back from XLM, eschewing XLM's Translation Language Modeling (TLM) objective since it requires a dataset of parallel sentences, which can be difficult to acquire. Instead, XLM-R trains RoBERTa(Liu et al., 2019) on a huge, multilingual dataset at an enormous scale: unlabelled text in 104 languages, extracted from CommonCrawl datasets, totalling 2.5TB of text. It is trained using only RoBERTa's (Liu et al., 2019) masked language modelling (MLM) objective. Surprisingly, this strategy provided better results in cross-lingual tasks. XLM-R outperforms mBERT on a variety of cross-lingual benchmarks such as cross-lingual natural language inference and cross-lingual question-answering (Conneau et al., 2020). This superior performance of XLM-R in cross-lingual tasks motivated us to use XLM-R in QE.

The *TransQuest* framework that is used to implement the two architectures described here relies on the XLM-R transformer model (Conneau et al., 2020) to derive the representations of the input sentences. The XLM-R transformer

model takes a sequence of no more than 512 tokens as input and outputs the representation of the sequence. The first token of the sequence is always [CLS], which contains the special embedding to represent the whole sequence, followed by embeddings acquired for each word in the sequence. As shown below, our proposed neural network architectures can utilise both the embedding for the [CLS] token and the embeddings generated for each word. The output of the transformer (or transformers for *SiameseTransQuest* described below) is fed into a simple output layer which is used to estimate the quality of a translation. We describe below how the XLM-R transformer is used and the output layer, as they are different in the two instantiations of the framework. The fact that we do not rely on a complex output layer makes training our architectures much less computationally intensive than state-of-the-art QE solutions such as predictor-estimator (Lee, 2020; Wang et al., 2018).

There are two pre-trained XLM-R models released by HuggingFace's Transformers library (Wolf et al., 2020); XLM-R-base and XLM-R-large. We used both of these pre-trained models in our experiments[5]. Both of these pre-trained XLM-R models cover 104 languages (Conneau et al., 2020), making it potentially very useful to estimate the translation quality for a large number of language pairs.

1. **MonoTransQuest** (**MTransQuest**): The first architecture proposed uses a single XLM-R transformer model and is shown in Figure 9.1a. This

[5]XLM-R-large is available on https://huggingface.co/xlm-roberta-large and XLM-R-base is available on https://huggingface.co/xlm-roberta-base

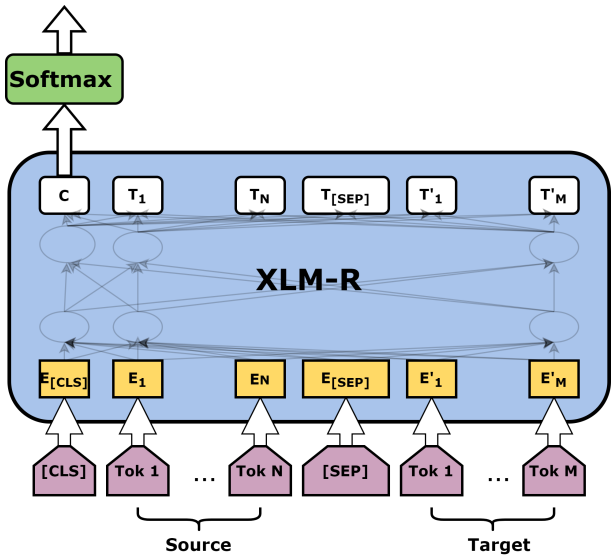
architecture produced the best results for monolingual STS tasks in Part I of the thesis. The input of this model is a concatenation of the original sentence and its translation, separated by the [SEP] token. We experimented with three pooling strategies for the output of the transformer model: using the output of the [CLS] token (`CLS`-strategy), computing the mean of all output vectors of the input words (`MEAN`-strategy), and computing a max-over-time of the output vectors of the input words (`MAX`-strategy). The output of the pooling strategy is used as the input of a softmax layer that predicts the quality score of the translation. We used the mean-squared-error loss as the objective function. Early experiments we carried out demonstrated that the `CLS`-strategy leads to better results than the other two strategies for this architecture. Therefore, we used the embedding of the [CLS] token as the input of a softmax layer.

2. **SiameseTransQuest** (**STransQuest**): The second approach proposed in this chapter relies on the Siamese architecture depicted in Figure 9.1b showed promising results in monolingual STS tasks (Reimers and Gurevych, 2019) in Part I of the thesis. In this case, we feed the original text and the translation into two separate XLM-R transformer models. Similar to the previous architecture, we used the same three pooling strategies for the outputs of the transformer models. We then calculated the cosine similarity between the two outputs of the pooling strategy.

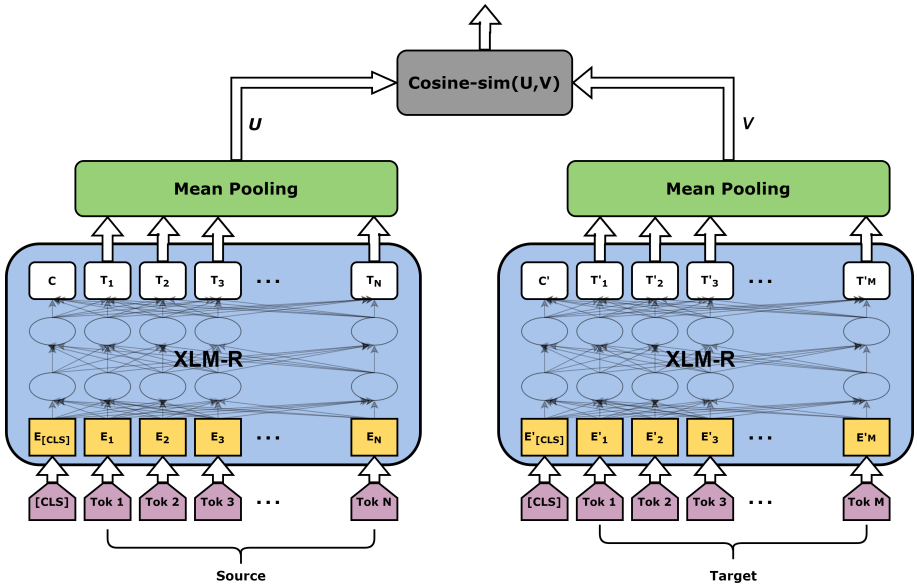
We used the mean-squared-error loss as the objective function. In the initial experiments we carried out with this architecture, the MEAN-strategy showed better results than the other two strategies. For this reason, we used the MEAN-strategy for our experiments. Therefore, cosine similarity is calculated between the the mean of all output vectors of the input words produced by each transformer.

9.1.1 Experimental Setup

We evaluated these two architectures in both aspects of sentence-level quality estimation using the data introduced in Chapter 8. We applied the same set of configurations for all the language pairs in order to ensure consistency between all the experiments. This also provides a good starting configuration for researchers who intend to use TransQuest on a new language pair. In both architectures, we used a batch-size of eight, Adam optimiser with $2e^{-5}$ learning rate, and a linear learning rate warm-up over 10% of the training data. During the training process, the parameters of the XLM-R model and the subsequent layers were updated. The models were trained using only training data. Furthermore, they were evaluated while training once in every 300 training steps using an evaluation set that had one-fifth of the rows in training data. We performed early stopping if the evaluation loss did not improve over ten evaluation steps. All the models were trained for three epochs.



(a) *MonoTransQuest* Architecture



(b) *SiameseTransQuest* Architecture

Figure 9.1: Two architectures employed in TransQuest.

9.2 Results and Discussion

The results for HTER and DA aspects are shown in Tables 9.1 and 9.2. We report the Pearson correlation (ρ) between the predictions and the gold labels

from the test set, which is the most commonly used evaluation metric in recent WMT quality estimation shared tasks (Specia et al., 2018; Fonseca et al., 2019; Specia et al., 2020) as mentioned in Chapter 8. Since we use the same evaluation process, we could compare our results with the baselines and best systems of the respective shared task. Raw I in Tables 9.1 and 9.2 shows the results for *MonoTransQuest* and *SiameseTransQuest* with XLM-R-large and Raw II in Tables 9.1 and 9.2 shows the results for *MonoTransQuest* and *SiameseTransQuest* with XLM-R-base. As can be seen in results, XLM-R-large always outperformed XLM-R-base in both architectures. Therefore, we use the results we got from XLM-R-large for the following analysis.

In the HTER aspect of sentence-level quality estimation, as shown in Table 9.1, *MonoTransQuest* gains ≈ 0.1-0.2 Pearson correlation boost over OpenKiwi in most language pairs. In the language pairs where OpenKiwi results are not available, *MonoTransQuest* gains ≈ 0.3-0.4 Pearson correlation boost over QuEst++ in all language pairs for both NMT and SMT. Table 9.1 also gives the results of the best system submitted for a particular language pair. It is worth noting that the *MonoTransQuest* results surpass the best system in all the language pairs with the exception of the En-De SMT, En-De NMT and En-Zh NMT datasets. *SiameseTransQuest* falls behind *MonoTransQuest* architecture, however it outperforms OpenKiwi and Quest++ in all the language pairs except En-De SMT. This shows that simple STS models with cross-lingual embeddings outperform the complex state-of-the-art QE models in predicting HTER.

As shown in Table 9.2, in the DA aspect of quality estimation,

	Method	Mid-resource				High-resource			
		En-Cs SMT	En-Ru NMT	En-Lv SMT	En-Lv NMT	De-En SMT	En-Zh NMT	En-De SMT	En-De NMT
I	MTransQuest	0.7207	0.7126	0.6592	0.7394	0.7939	0.6119	0.7137	0.5994
	STransQuest	0.6853	0.6723	0.6320	0.7183	0.7524	0.5821	0.6992	0.5875
II	MTransQuest-base	0.7012	0.6982	0.6254	0.7110	0.7653	0.5873	0.6872	0.5762
	STransQuest-base	0.6678	0.6542	0.6132	0.6892	0.7251	0.5551	0.6672	0.5532
III	MTransQuest ⊗	0.7300	0.7226	0.6601	0.7402	0.8021	0.6289	0.7289	0.6091
	STransQuest ⊗	0.6901	0.6892	0.6451	0.7251	0.7649	0.5967	0.7041	0.5991
IV	MTransQuest ⊗ - Aug	**0.7399**	**0.7307**	**0.6792**	**0.7492**	**0.8109**	0.6367	**0.7398**	0.6156
	STransQuest ⊗ - Aug	0.7002	0.6901	0.6498	0.7301	0.7667	0.5991	0.7134	0.6098
V	Quest ++	0.3943	0.2601	0.3528	0.4435	0.3323	NR	0.3653	NR
	OpenKiwi	NR	0.5923	NR	NR	NR	0.5058	0.7108	0.3923
	Best system	0.6918	0.5923	0.6188	0.6819	0.7888	**0.6641**	0.7397	**0.7582**
VI	MTransQuest-B	0.6423	0.6354	0.5772	0.6531	0.7005	0.5483	0.6239	0.5002
	STransQuest-B	0.5987	0.5872	0.5012	0.5901	0.6572	0.5098	0.5762	0.4551

Table 9.1: Pearson correlation (ρ) between *TransQuest* algorithm predictions and human post-editing effort. The best result for each language by any method is highlighted in bold. Row **I** shows the results for XLM-R-large model and Row **II** shows the results for XLM-R-base. Row **III** and Row **IV** present the results for further fine-tuning strategies explained in Section 9.3. Row **V** shows the results of the baseline methods and the best system submitted for the language pair in that competition. **NR** implies that a particular result was *not reported* by the organisers. Row **VI** presents the results of the multilingual BERT (mBERT) model in TransQuest Architectures.

MonoTransQuest gained ≈ 0.2-0.3 Pearson correlation boost over OpenKiwi in all the language pairs. Additionally, *MonoTransQuest* achieves ≈ 0.4 Pearson correlation boost over OpenKiwi in the low-resource language pair Ne-En. Similar to HTER, in DA *SiameseTransQuest* falls behind *MonoTransQuest* architecture, however still *SiameseTransQuest* gained ≈ 0.2-0.3 Pearson correlation boost over OpenKiwi in all the language pairs. This shows that simple STS models with cross-lingual embeddings outperforms the complex state-of-the-art QE models in predicting DA too.

		Low-resource		Mid-resource			High-resource	
	Method	Si-En	Ne-En	Et-En	Ro-En	Ru-En	En-De	En-Zh
I	MTransQuest	0.6525	0.7914	0.7748	0.8982	0.7734	0.4669	0.4779
	STransQuest	0.5957	0.7081	0.6804	0.8501	0.7126	0.3992	0.4067
II	MTransQuest-base	0.6412	0.7823	0.7651	0.8715	0.7593	0.4421	0.4593
	STransQuest-base	0.5773	0.6853	0.6692	0.8321	0.6962	0.3832	0.3975
III	MTransQuest ⊗	0.6661	0.8023	0.7876	0.8988	0.7854	0.4862	0.4853
	STransQuest ⊗	0.6001	0.7132	0.6901	0.8629	0.7248	0.4096	0.4159
IV	MTransQuest ⊗ - Aug	**0.6849**	**0.8222**	**0.8240**	**0.9082**	**0.8082**	**0.5539**	**0.5373**
	STransQuest ⊗ - Aug	0.6241	0.7354	0.7239	0.8621	0.7458	0.4457	0.4658
V	OpenKiwi	0.3737	0.3860	0.4770	0.6845	0.5479	0.1455	0.1902
VI	MTransQuest-B	NS	0.6452	0.6231	0.8351	0.6661	0.3765	0.3982
	STransQuest-B	NS	0.5368	0.5431	0.7652	0.5541	0.3356	0.3462

Table 9.2: Pearson correlation (ρ) between *TransQuest* algorithm predictions and human DA judgments. The best result for each language by any method is highlighted in bold. Row **I** shows the results for XLM-R-large model and Row **II** shows the results for XLM-R-base. Row **III** and Row **IV** present the results for further fine-tuning strategies explained in Section 9.3. Row **V** shows the results of the baseline method; OpenKiwi. Row **VI** presents the results of the multilingual BERT (mBERT) model in TransQuest Architectures.

If we consider the two architectures; *MonoTransQuest* and *SiameseTransQuest*, *MonoTransQuest* outperformed the *SiameseArchitecture* in all the language pairs in both aspects of sentence-level quality estimation. This is similar to the experiments we discussed for monolingual STS in Part I of the thesis. The two architectures have a trade-off between accuracy and efficiency. On an Nvidia Tesla K80 GPU, *MonoTransQuest* takes 4,480s on average to train on 7,000 instances, while *SiameseTransQuest* takes only 3,900s on average for the same experiment. On the same GPU, *MonoTransQuest* takes 35s on average to perform inference on 1,000 instances which takes *SiameseTransQuest*

only 16s to do so. Therefore we recommend using *MonoTransQuest* where accuracy is valued over efficiency, and *SiameseTransQuest* where efficiency is prioritised above accuracy. Furthermore, as we discussed in Part I of the thesis, Siamese architecture outputs sentence vectors. Therefore finding the best quality translation in a collection of translations would take less time with the Siamese architecture. Since there is a growing interest[6] in the NLP community for energy efficient machine learning models, we decided to support both architectures in the TransQuest Framework.

With these results, we can answer our **RQ1:** Can the state-of-the-art STS models be applied in sentence-level QE tasks by changing the embeddings? We show that state-of-the-art STS methods based on cross-lingual transformer models outperform current state-of-the-art QE algorithms based on complex architectures like predictor-estimator in both aspects of sentence-level QE. We support this with the results for more than 15 language pairs with a wide variety from different domains and different MT types.

Additionally, Row VI in both Tables 9.1 and 9.2 shows the results of multilingual BERT (mBERT) in *MonoTransQuest* and *SiameseTransQuest* architectures. We used the same settings similar to XLM-R. The results show that XLM-R model outperforms the mBERT model in all the language pairs of both aspects in quality estimation. As shown in Row II in both Tables 9.1 and 9.2 even XLM-R-base model outperforms mBERT model in all the languages pairs

[6]Several workshops like *EMC*^{2} (Workshop on Energy Efficient Machine Learning and Cognitive Computing), SustaiNLP 2020 (Workshop on Simple and Efficient Natural Language Processing) has been organised on this aspect.

with both architectures. Therefore, we can safely assume that the cross-lingual nature of the XLM-R transformers had a clear impact on the results.

With this, we can answer our **RQ2:** Using cross-lingual embeddings with STS architecture proved advantageous rather than using multilingual embeddings. Therefore, state-of-the-art cross-lingual transformer models such as XLM-R should be utilised more in QE tasks.

9.3 Further Fine-tuning

In this section, we improve the results of TransQuest through two fine-tuning strategies; ensemble learning and data augmentation.

Ensemble Learning learning uses multiple learning algorithms to obtain better predictive performance than any of the constituent learning algorithms alone. We perform a weighted average ensemble for the output of the XLM-R-large and the output of the XLM-R-base for both architectures in TransQuest. We experimented on weights 0.8:0.2, 0.6:0.4, 0.5:0.5 on the output of XLM-R-large and output from XLM-R-base, respectively. Since the results from the XLM-R-base transformer model are slightly worse than those from the XLM-R-large, we did not consider the weight combinations that give higher weight to XLM-R-base transformer model results. We obtained the best results when we used the weights 0.8:0.2. We report the results from the two architectures from this step in Row III of Tables 9.1 and 9.2 as MTransQuest ⊗ and STransQuest ⊗.

As shown in Tables 9.1 and 9.2 both architectures in TransQuest with

ensemble learning gained ≈ 0.01-0.02 Pearson correlation boost over the default settings for all the language pairs in both aspects of sentence-level QE.

Data Augmentation adds additional training instances for the training process. As we already experienced with STS experiments in Part I of the thesis, this often leads to better performance in the machine learning algorithms. Therefore, to experiment with how TransQuest performs with more data, we trained TransQuest on an augmented dataset. Alongside the training, development and testing datasets, the shared task organisers also provided the parallel sentences used to train the neural machine translation system in each language. In the data augmentation setting, we added the sentence pairs from that neural machine translation system training file to the training dataset we used to train TransQuest. In order to find the best setting for the data augmentation, we experimented with randomly adding 1000, 2000, 3000, up to 5000 sentence pairs. Since the ensemble learning performed better than XLM-R-large results of TransQuest, we conducted this data augmentation experiment on the ensemble learning setting. We assumed that the sentence pairs added from the neural machine translation system training file have maximum translation quality.

Up to 2000 sentence pairs, the results continued to get better. However, adding more than 2000 sentence pairs did not improve the results. We did not experiment with adding any further than 5000 sentence pairs to the training set since we were aware that adding more sentence pairs with the maximum

translation quality to the training file will make it imbalanced and negatively affect the machine learning models' performance. We report the results from the two architectures from this step in Row IV of Tables 9.1 and 9.2 as MTransQuest ⊗-Aug and STransQuest ⊗-Aug.

Data augmentation provided the best results for all of the language pairs in both aspects of sentence-level QE. As shown in Tables 9.1 and 9.2, both architectures in TransQuest with the data augmentation setting gained ≈ 0.01-0.09 Pearson correlation boost over the XLM-R-large in all the language pairs. Additionally, for DA, *MTransQuest ⊗-Aug* achieves ≈ 0.09 Pearson correlation boost over TransQuest with XLM-R large in the high-resource language pair En-De, which is the biggest boost got for any language pair.

This answers our **RQ3:** *Can further fine-tuning strategies be used to improve the results?*. We show that ensemble learning and data augmentation can be used to improve the results. In fact, data augmentation combined with ensemble learning provided the best results for all the language pairs in both aspects of the sentence-level QE.

Finally, we submitted the best result we had with TransQuest (MonoTransQuest with ensemble learning and data augmentation) to WMT 2020 QE shared task 1 (Specia et al., 2020). The official results of the competition show that *TransQuest* won first place in all the language pairs in the *Sentence-Level Direct Assessment* task. *TransQuest* is the sole winner in En-Zh, Ne-En and Ru-En language pairs and the multilingual track. For the other language pairs (En-De, Ro-En, Et-En and Si-En), it shares first place with (Fomicheva

et al., 2020a), whose results are not statistically different from ours. Therefore, TransQuest is the new state-of-the-art in sentence-level QE.

9.4 Error analysis

In an attempt to better understand the performance and limitations of *TransQuest* we carried out an error analysis on the results obtained on Romanian - English and Sinhala - English. The availability of native speakers determined the choice of language pairs we analysed to perform this analysis. We focused on cases where the difference between the predicted and expected scores was high. This included instances where the predicted score was underestimated and overestimated.

Analysis of the results does not reveal obvious patterns. The largest number of errors seem to be caused by named entities in the source sentences. In some cases, these entities are mishandled during the translation. The resulting sentences are usually syntactically correct but semantically odd. Typical examples are *RO: În urmă explorărilor Căpitanului James Cook, Australia și Noua Zeelandă au devenit ținte ale colonialismului britanic. (As a result of Captain James Cook's explorations, Australia and New Zealand have become the targets of British colonialism.) - EN: Captain James Cook, Australia and New Zealand have finally become the targets of British colonialism.* (expected: -1.2360, predicted: 0.2560) and *RO: O altă problemă importantă cu care trupele Antantei au fost obligate să se confrunte a fost malaria. (Another important problem that the Triple Entente troops had to face was malaria.)* - EN: *Another important problem that Antarctic troops*

had to face was malaria. (expected: 0.2813, predicted: -0.9050). However, it is debatable whether the expected scores for these two pairs should be so different. Both of them have apparent problems and cannot be clearly understood without reading the source. For this reason, we would expect that both of them have low scores. Instances such as this also occur in the training data. As a result of this, *TransQuest* learns contradictory information, which in turn leads to errors at the testing stage.

A large number of problems are caused by incomplete source sentences or input sentences with noise. For example the pair *RO: thumbright250pxDrapelul cu fâșiile în poziție verticală (The flag with strips in upright position) - EN: ghtghtness 250pxDrapel with strips in upright position* has an expected score of 0.0595, but our method predicts -0.9786. Given that only *ghtghtness 250pxDrapel* is wrong in the translation, the predicted score is far too low. In an attempt to see how much this noise influences the result, we ran the system with the pair *RO: Drapelul cu fâșiile în poziție verticală - EN: Drapel with strips in upright position.* The prediction is 0.42132, which is more in line with our expectations, given that one of the words is not translated.

Similar to Ro-En, in Si-En, most problems seem to be caused by the presence of named entities in the source sentences. For example, in the English translation: *But the disguised Shiv will help them securely establish the statue.* (expected: 1.3618, predicted: -0.008), the correct English translation would be *But the disguised Shividru will help them securely establish the statue..* Only the named entity *Shividru* is translated incorrectly. Therefore the annotators

have annotated the translation with high quality. However, TransQuest fails to identify that. Similar scenarios can be found in English translations *Kamala Devi Chattopadhyay spoke at this meeting, Dr. Ann.* (expected:1.3177, predicted:-0.2999) and *The Warrior Falls are stone's, halting, heraldry and stonework rather than cottages. The cathedral manor is navigable places* (expected:0.1677, predicted:-0.7587). It is clear that the presence of the named entities seem to confuse the algorithm we used. Hence it needs to handle named entities in a proper way.

9.5 Conclusion

In this chapter, we introduced *TransQuest*, a new open-source framework for quality estimation based on cross-lingual transformers. *TransQuest* is implemented using PyTorch (Paszke et al., 2019) and HuggingFace (Wolf et al., 2020) and supports the training of sentence-level quality estimation systems on new data. It outperforms other open-source tools on both aspects of sentence-level quality estimation and yields new state-of-the-art quality estimation results.

We propose two architectures: *MonoTransQuest* and *SiameseTransQuest*. As we showed in Part I of the thesis, they provide state-of-the-art results for STS. However, neither of them have been previously explored in QE tasks. It is the first time that STS architectures have been proposed to the QE task, as far as we know. We further improve the results with data augmentation and ensemble learning. The best results we achieved in this chapter were submitted to WMT 2020 QE task 1, and it won first place in all the language pairs. We can conclude

that TransQuest is state-of-the-art in QE now.

We have open-sourced the framework, and the pre-trained sentence-level QE models for 15 language pairs are freely available to the community in HuggingFace model hub. Upon releasing, TransQuest has caught wide attention from the community resulting in more than 9,000 downloads within the first year. Furthermore, TransQuest has been used as the baselines for many shared tasks, including the most recent edition of WMT QE shared task[7] and *Explainable Quality Estimation shared task* on the second workshop on Evaluation & Comparison of NLP Systems[8]. Not only among researchers, TransQuest is also popular in the industry. ModelFront, which is a leading company in translation risk prediction, lists TransQuest as an option on their website[9]. We believe that the simplicity of the architectures in TransQuest is the reason for this popularity.

Sentence-level QE models are challenging to interpret as they only provide a score for the whole translation without indicating where the errors are. To overcome this with TransQuest, we introduce minor changes to the sentence-level architecture to support word-level quality estimation in Chapter 10. One limitation with TransQuest is that the pre-trained QE models are big in size as they depend on transformer models. Therefore, managing several of them for each language pair would be chaotic in a real-world application. We seek solutions for that with multilingual QE models in Chapter 11.

[7]WMT 2021 QE shared task is available on http://statmt.org/wmt21/quality-estimation-task.html

[8]Explainable Quality Estimation shared task is available on https://eval4nlp.github.io/sharedtask.html

[9]ModelFront options are available on https://modelfront.com/options

As future work, we hope to explore cross-lingual embeddings more with TransQuest for the sentence-level QE task. For example, XeroAlign (Gritta and Iacobacci, 2021) provides a simple method for task-specific alignment of cross-lingual pre-trained transformers, which would be interesting to apply in QE tasks. Furthermore, Facebook has introduced large cross-lingual models recently; XLM-R XL and XLM-R XXL, which have improved results for many cross-lingual NLP tasks (Goyal et al., 2021). These models have the obvious disadvantage of being big in size, yet they would be interesting to apply in QE.

Chapter 10

Extending TransQuest for Word-Level QE

Translation quality can be estimated at different levels of granularity: word, sentence and document level (Ive et al., 2018). So far in this thesis, we have only explored sentence-level QE (Specia et al., 2020), in which QE models provide a score for each pair of source and target sentences. A more challenging task, which is currently receiving a lot of attention from the research community, is the word-level quality estimation which provides more fine-grained information about the quality of a translation, indicating which words from the source have been incorrectly translated in the target (good vs bad words), and whether the words inserted between these words are correct (good vs bad gaps). This information can be useful for post-editors by indicating the parts of a translation on which they have to focus more. Therefore, word-level QE solutions would certainly improve the efficiency of the post-editors.

Furthermore, as mentioned before, sentence-level QE models are difficult to explain as they only output a single score for the translation. The users face the difficulty of interpreting the score and how to make use of the information from the QE method. However, in contrast to that, word-level QE models provide more fine-grained information about the quality of a translation. As a result, they can

improve the usability of the whole QE process. Since there is a growing interest in the NLP community for practical machine learning, we believe that having more usable QE models would improve the popularity of QE.

With these advantages, researchers have provided many solutions for word-level QE. Similar to sentence-level QE, state-of-the-art models in word-level QE are also relying on deep learning. As we explained in Chapter 8, most of the open-source quality estimation frameworks, such as OpenKiwi, uses the same architectures for both word-level and sentence-level QE. For example, the predictor-estimator architecture used in OpenKiwi provides the best results for word-level QE too. Therefore, word-level QE also suffers from the same limitation we mentioned in the last chapter. The architectures are very complex and need a lot of computing resources to train the models, which has seriously hindered the success of word-level QE in real-world applications (Ranasinghe et al., 2021b).

To overcome this, we propose a simple architecture to perform word-level QE. The proposed architecture is very similar to the *MonoTransQuest* architecture we introduced in the last chapter, which in turn was based on a state-of-the-art STS method. We only change the output layer of the *MonoTransQuest* architecture so that it can produce word-level qualities. This architecture is simple and effective when compared with the current state-of-the-art word-level QE architectures such as predictor-estimator. We believe that a simpler architecture can improve the popularity of word-level QE in real-world applications.

As mentioned in Chapter 8, word-level QE systems should have three features in them. *i.* Predict the qualities of the words in the target. *ii.* Predict the qualities of the words in the source. *iii.* Predict the qualities of the gaps in the target. As a result, WMT word-level QE shared task has three separate evaluation metrics focussing on each of the above features. However, most of the open-source word-level QE frameworks ignore predicting the quality of the gaps in the target sentence. For example, Marmot (Logacheva et al., 2016) only supports predicting the quality of the words in the target, and OpenKiwi (Kepler et al., 2019) only supports predicting the quality of the words in the source and the target. They completely ignore predicting the quality of the gaps. Despite that, predicting the quality of the gaps in the target would be important and useful for post-editors. Therefore, when we design the proposed architecture in this chapter, we considered all three features in word-level QE; predicting the quality of the words in the target, predicting the quality of the words in the source and predicting the quality of the gaps in the target. We believe an approach that supports every feature in word-level QE will be stronger than existing solutions.

We address three research questions in this chapter:

RQ1: Can existing state-of-the-art STS architecture be used to predict all the features in the word-level QE task by just modifying the embeddings and output layer?

RQ2: Do cross-lingual embeddings have an advantage over multilingual embeddings in the word-level QE task?

RQ3: Can the proposed model be further improved by performing ensemble

learning?

The main contributions of this chapter are as follows.

1. We introduce a simple architecture to perform word-level quality estimation that predicts the quality of the words in the source sentence, target sentence and the gaps in the target sentence.

2. We evaluate it on eight different language pairs in which the word-level QE data was available, and we show that the proposed architecture outperforms the current state-of-the-art word-level QE frameworks such as Marmot (Logacheva et al., 2016) and OpenKiwi (Kepler et al., 2019).

3. We suggest further improvements to the model by performing ensemble learning.

4. We integrated the architecture with TransQuest, which already had two sentence-level architectures described in Chapter 9[1]. TransQuest framework was already popular with the NLP community and adding a word-level architecture to that boosted its value. Additionally the pre-trained word-level QE models for eight language pairs are freely available to the community[2].

The rest of this chapter is organised as follows. Section 10.1 discusses the methodology and the experiments conducted with eight language pairs in word-

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

[2]Pre-trained word-level QE models are available on https://huggingface.co/models?filter=microtransquest

level QE. Section 10.2 shows the results of three evaluation metrics used in word-level QE. Section 10.3 provides further fine-tuning strategies to improve the results. The chapter finishes with conclusions and ideas for future research directions in word-level QE.

10.1 Methodology

The proposed architecture for the word-level QE is very similar to the *MonoTransQuest* architecture in Chapter 9 where the source and the target are processed through a single transformer model. The difference here is since we need quality for each word, we do not use a pooling mechanism as we did with *MonoTransQuest*. Instead, we keep the state of the individual tokens to get their quality.

As we also need to consider the quality of the GAPs in word-level QE, we first add a new token to the tokeniser of the transformer called <GAP> which is inserted between the words in the target. We then concatenate the source and the target with a [SEP] token as in *MonoTransQuest* architecture and feed them into a single transformer. A simple linear layer is added on top of the word and <GAP> embeddings to predict whether it is "Good" or "Bad" as shown in Figure 10.1. Each of the softmax layers we used on top of the transformer model consists of two neurons. They provide two probabilities for each word or gap, stating the probability of being "Good" or "Bad". We consider the class with the maximum probability as the quality of a particular word or gap. As we integrated this architecture to the *TransQuest* framework and it provides quality for the

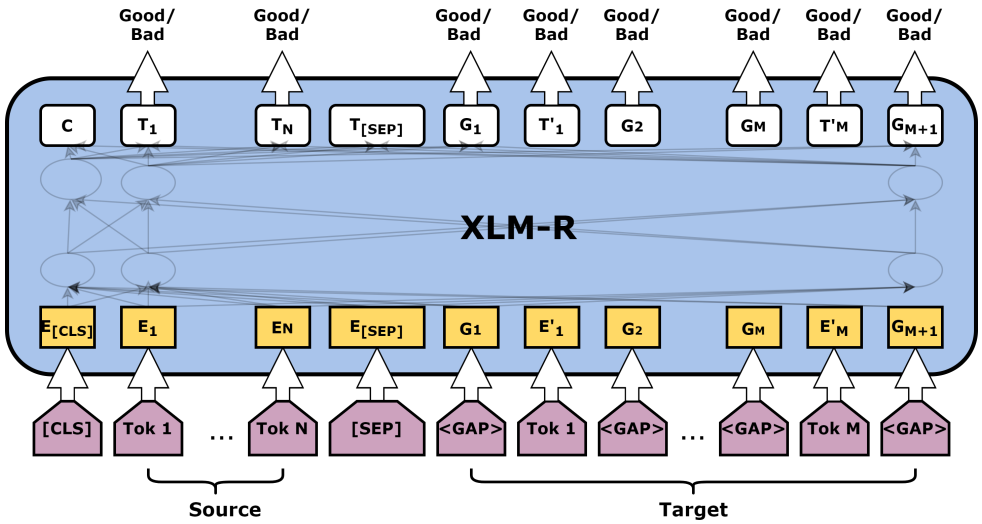


Figure 10.1: MicroTransQuest Architecture

smallest unit possible in QE, we named the proposed word-level architecture as *MicroTransQuest*.

Similar to the sentence-level QE architectures, we used cross-lingual transformer models for this architecture. As explained in Chapter 9, XLM-R provides state-of-the-art cross-lingual transformer models. There are two pre-trained XLM-R models released by HuggingFace's Transformers library (Wolf et al., 2020); XLM-R-base and XLM-R-large. We used both of these pre-trained models in our experiments[3]. Both of these pre-trained XLM-R models cover 104 languages (Conneau et al., 2020), making them potentially very useful to estimate the word-level translation quality for a large number of language pairs.

[3]XLM-R-large is available on https://huggingface.co/xlm-roberta-large and XLM-R-base is available on https://huggingface.co/xlm-roberta-base

10.1.1 Experimental Setup

We evaluated the architecture in word-level quality estimation using the data introduced in Chapter 8. We applied the same set of configurations for all the language pairs to ensure consistency between all the experiments. This also provides a good starting configuration for researchers who intend to use *MicroTransQuest* on a new language pair. We used a batch-size of eight, Adam optimiser with a learning rate $1e^{-5}$, and a linear learning rate warm-up over 10% of the training data. During the training process, the parameters of the XLM-R model and the subsequent layers were updated. The models were trained using only training data. Furthermore, they were evaluated while training once in every 300 training steps using an evaluation set with one-fifth of the training data instances. We performed early stopping if the evaluation loss did not improve over ten evaluation steps. All the models were trained for three epochs.

10.2 Results and Evaluation

For the evaluation, we used the approach proposed in the WMT shared tasks in which the classification performance is calculated using the multiplication of F1-scores ($F1_{MULTI}$) for the 'OK' and 'BAD' classes against the true labels independently: words in the target ('OK' for correct words, 'BAD' for incorrect words), gaps in the target ('OK' for genuine gaps, 'BAD' for gaps indicating missing words) and source words ('BAD' for words that lead to errors in the target, 'OK' for other words) (Specia et al., 2018) as explained in Chapter 8. In

WMT QE shared tasks, Word-level QE systems have been evaluated using all three of these evaluation metrics. Therefore, we follow the same process so that we can compare our results with the baselines and best systems of the respective shared task.

$F1_{MULTI}$ for the words in the target ($F1_{MULTI}$ $Target$), $F1_{MULTI}$ for the gaps in the target ($F1_{MULTI}$ $GAPS$) and $F1_{MULTI}$ for the words in the source ($F1_{MULTI}$ $Source$) are shown in Tables 10.1, 10.2 and 10.3 respectively. Before WMT 2019, organisers provided separate scores for gaps and words in the target, while after WMT 2019, they produce a single result for target gaps and words. Therefore, we report ($F1_{MULTI}$ $GAPS$) only on the language pairs released in WMT 2018 in Table 10.2. For the language pairs in WMT 2019 and 2020, we report a single result for the target gaps and words as ($F1_{MULTI}$ $Target$) in Table 10.1.

Raw I in Tables 10.1, 10.2 and 10.3 shows the results for *MicroTransQuest* with XLM-R-large and Raw II in Tables 10.1, 10.2 and 10.3 shows the results for *MicroTransQuest* with XLM-R-base. As can be seen in results, XLM-R-large always outperformed XLM-R-base. Therefore, we use the results we got from XLM-R-large for the following analysis.

In $F1_{MULTI}$ $Target$ evaluation metric in word-level QE, as shown in Table 10.1, *MicroTransQuest* outperforms OpenKiwi and Marmot in all the language pairs we experimented. Additionally, *MicroTransQuest* achieves ≈ 0.3 $F1_{MULTI}$ score boost over OpenKiwi in the En-Ru NMT. Furthermore, *MicroTransQuest* gained $\approx$ 0.15-0.2 $F1_{MULTI}$ score boost over Marmot in all the language pairs. Table 10.1 also

	Method	Mid-resource				High-resource			
		En-Cs SMT	En-Ru NMT	En-Lv SMT	En-Lv NMT	De-En SMT	En-Zh NMT	En-De SMT	En-De NMT
I	MicroTransQuest	0.6081	0.5592	0.5939	0.5868	0.6485	0.5602	0.6348	0.5013
II	MicroTransQuest-base	0.5642	0.5132	0.5579	0.5431	0.6001	0.5202	0.5985	0.4698
III	MicroTransQuest ⊗	**0.6154**	**0.5672**	**0.6123**	**0.6003**	**0.6668**	0.5678	**0.6451**	0.5121
IV	Marmot	0.4449	NR	0.3445	0.4208	0.4373	NR	0.3653	NR
	OpenKiwi	NR	0.2412	NR	NR	NR	0.5583	NR	0.4111
	Best system	0.4449	0.4780	0.3618	0.4293	0.6012	**0.6415**	0.6246	**0.6186**
V	MicroTransQuest-B	0.5462	0.4987	0.5043	0.5132	0.5643	0.4892	0.5381	0.4371

Table 10.1: $F1_{MULTI}$ *Target* between the algorithm predictions and human annotations. The best result for each language by any method is highlighted in bold. Row **I** shows the results for XLM-R-large model and Row **II** shows the results for XLM-R-base. Row **III** presents the results for further fine-tuning strategies explained in Section 10.3. Row **IV** shows the results of the baseline methods and the best system submitted for the language pair in that competition. **NR** implies that a particular result was *not reported* by the organisers. Row **V** presents the results of the multilingual BERT (mBERT) model in *MicroTransQuest*.

gives the results of the best system submitted for a particular language pair. It is worth noting that the *MicroTransQuest* results surpass the best system in all the language pairs, with the exception of the En-De NMT and En-Zh NMT datasets.

In $F1_{MULTI}$ *GAPS* evaluation metric in word-level QE, as shown in Table 10.2 *MicroTransQuest* outperforms best systems submitted to the respective shared tasks in all the language pairs we experimented. *MicroTransQuest* achieves $\approx$ 0.05-0.2 $F1_{MULTI}$ score boost over best systems. Notably, *MicroTransQuest* achieves $\approx$ 0.2 $F1_{MULTI}$ score boost over the best system in En-De SMT. As we mentioned before, it should be noted that neither of the baselines, Marmot, nor OpenKiwi supports predicting the quality of *gaps* in the target sentence. Since *MicroTransQuest* supports this and produces state-of-the-results in this

	Method	Mid-resource			High-resource	
		En-Cs SMT	En-Lv SMT	En-Lv NMT	De-En SMT	En-De SMT
I	MicroTransQuest	0.2018	0.2356	0.1664	0.4203	0.4927
II	MicroTransQuest-base	0.1876	0.2132	0.1452	0.4098	0.4679
III	MicroTransQuest ⊗	**0.2145**	**0.2437**	**0.1764**	**0.4379**	**0.4982**
IV	Marmot	NS	NS	NS	NS	NS
	Best system	0.1671	0.1386	0.1598	0.3176	0.3161
V	MicroTransQuest-B	0.1778	0.2089	0.1387	0.3892	0.4371

Table 10.2: $F1_{MULTI}$ *GAPS* between the algorithm predictions and human annotations. The best result for each language by any method is highlighted in bold. Row **I** shows the results for XLM-R-large model and Row **II** shows the results for XLM-R-base. Row **III** presents the results for further fine-tuning strategies explained in Section 10.3. Row **IV** shows the results of the baseline methods and the best system submitted for the language pair in that competition. **NS** implies that a particular baseline does *not support* predicting quality of gaps. Row **V** presents the results of the multilingual BERT (mBERT) model in *MicroTransQuest*.

evaluation metric, we believe that using *MicroTransQuest* in word-level QE would be beneficial than using OpenKiwi and Marmot.

Finally, in $F1_{MULTI}$ *Source* evaluation metric, as shown in Table 10.3 *MicroTransQuest* outperforms OpenKiwi in all the language pairs we experimented. It should be noted that Marmot does not support predicting the quality of the words in the source. On the other hand, OpenKiwi supports this, but *MicroTransQuest* achieves ≈ 0.05-0.3 $F1_{MULTI}$ score boost in all the language pairs. Furthermore, *MicroTransQuest* results surpass the best system in all the language pairs with the exception of the En-De NMT and En-Zh NMT datasets.

With these results, we can answer our **RQ1:** *Can existing state-of-the-art*

	Method	Mid-resource				High-resource			
		En-Cs SMT	En-Ru NMT	En-Lv SMT	En-Lv NMT	De-En SMT	En-Zh NMT	En-De SMT	En-De NMT
I	MicroTransQuest	0.5327	0.5543	0.4945	0.4880	0.4824	0.4040	0.5269	0.4456
II	MicroTransQuest-base	0.5134	0.5287	0.4652	0.4571	0.4509	0.3876	0.5012	0.4185
III	MicroTransQuest ⊗	**0.5431**	**0.5640**	**0.5076**	**0.4892**	**0.4965**	0.4145	**0.5387**	0.4578
IV	Marmot	NS	NR	NS	NS	NS	NR	NS	NR
	OpenKiwi	NR	0.2647	NR	NR	NR	0.3729	NR	0.3717
	Best system	0.3937	0.4541	0.4945	0.3614	0.3200	**0.4462**	0.3368	**0.5672**
V	MicroTransQuest-B	0.4987	0.5098	0.4441	0.4256	0.4187	0.3567	0.4672	0.3985

Table 10.3: $F1_{MULTI}$ *Source* between the algorithm predictions and human annotations. The best result for each language by any method is highlighted in bold. Row **I** shows the results for XLM-R-large model and Row **II** shows the results for XLM-R-base. Row **III** presents the results for further fine-tuning strategies explained in Section 10.3. Row **IV** shows the results of the baseline methods and the best system submitted for the language pair in that competition. **NR** implies that a particular result was *not reported* by the organisers. Row **V** presents the results of the multilingual BERT (mBERT) model in *MicroTransQuest*.

STS architecture be used to predict all features in the word-level QE task by just modifying the embeddings and the output layer? We show that the state-of-the-art STS method based on cross-lingual transformer models can be used in word-level QE by changing the output layer. It outperforms current state-of-the-art word-level QE methods such as OpenKiwi that relies on complex neural network architectures. Our proposed architecture is not only simple, but also achieves state-of-the-art results in all the language pairs we experimented. Furthermore, *MicroTransQuest* is the only architecture that supports predicting the quality of the gaps in target. We believe that the way we designed the architecture based on state-of-the-art STS architectures gave us the ability to support predicting the quality of the gaps in target with *MicroTransQuest*.

Row VI in Tables 10.1, 10.2 and 10.3 shows the results of multilingual BERT (mBERT) in *MicroTransQuest* architecture. We used the same settings similar to XLM-R. The results show that the XLM-R model outperforms the mBERT model in all the language pairs. As shown in Row II in Tables 10.1, 10.2 and 10.3 even XLM-R-base model outperform mBERT model in all the languages pairs. Therefore, we can safely assume that the cross-lingual nature of the XLM-R transformers had a clear impact on the results. This observation is similar to what we experienced in Chapter 9 with sentence-level QE experiments.

With this, we can answer our **RQ2:** Using cross-lingual embeddings with STS architecture proved advantageous rather than using multilingual embeddings. As far as we know, this is the first time XLM-R was used in a task related to detecting missing words. With the results we got from predicting the quality of the gaps in target, we show that XLM-R can be used in tasks similar to detecting missing words.

10.3 Further Improvements

In this section, we improve the results of *MicroTransQuest* through ensemble learning as we did with sentence-level quality estimation. As mentioned before in Section 10.1 softmax layer provides two probabilities for each word or gap. We calculated these probabilities for each word using XLM-R-large and XLM-R base. Then we performed a weighted average ensemble on these probabilities, and we consider the class with the highest probability after performing the ensemble as the quality of the word or gap. We experimented on weights

0.8:0.2, 0.6:0.4, 0.5:0.5 on the output of XLM-R-large and output from XLM-R-base, respectively. Since the results we got from XLM-R-base transformer model are slightly worse than the results we got from XLM-R-large we did not consider the weight combinations that gives higher weight to XLM-R-base transformer model results. We obtained the best results when we used the weights 0.6:0.4. We report the results from this step in Row III of Tables 10.1, 10.2 and 10.3 as MicroTransQuest ⊗.

As shown in Tables 10.1, 10.2 and 10.3 ensemble learning improved the results for *MicroTransQuest* in all three evaluation metrics. For all the language pairs $F1_{MULTI}$ *Target*, $F1_{MULTI}$ *GAPS*, and $F1_{MULTI}$ *Source* scores gained ≈ 0.01-0.02 boost with ensemble learning. This answers our **RQ3:** *Can the proposed model be further improved by performing ensemble learning?* We show that ensemble learning can be used to improve the word-level QE results similar to sentence-level QE. In fact, ensemble learning provided the best results for *MicroTransQuest* in all the language pairs we experimented.

10.4 Conclusion

In this chapter, we explored word-level QE with transformers. We introduced a new architecture based on transformers to perform word-level QE. The proposed architecture is very similar to the sentence-level QE architecture; *MonoTransQuest*. However, the output layers are different to *MonoTransQuest* so that they keep the state of the individual tokens to get their quality. We evaluated the proposed architecture; *MicroTransQuest* on eight language pairs where the

word-level QE data was available. It outperforms other open-source tools like OpenKiwi and Marmot on all three evaluations metrics of word-level quality estimation and yields new state-of-the-art word-level QE results. Furthermore, *MicroTransQuest* is the only open-source architecture that supports predicting the quality of the gaps in the target sentence. The architecture we proposed in this chapter is very simple compared to the predictor-estimator architecture in OpenKiwi, yet this architecture produces better results. We further improved the results with ensemble learning showed that *MicroTransQuest* outperforms the majority of the best systems submitted for that language in each shared task. We can conclude that state-of-the-art STS architecture can also be used in word-level QE by doing small modifications to the output layers.

The implementation of the architecture, which is based on PyTorch (Paszke et al., 2019) and Hugging Face (Wolf et al., 2020), has been integrated into the TransQuest framework (Ranasinghe et al., 2020c) which won the WMT 2020 QE task (Specia et al., 2020) on sentence-level direct assessment (Ranasinghe et al., 2020b)[4]. The pre-trained word-level QE models for eight language pairs are available to the public on HuggingFace model hub.

One limitation of the proposed architecture is that it only predicts word-level qualities. Managing two models to predict the word-level and sentence-level qualities separately would be chaotic in some situations as the two models can produce contradictory predictions. Therefore, in the future, we hope to explore

[4]The details about the word-level QE architecture in TransQuest is available on http://tharindu.co.uk/TransQuest/architectures/word_level_architecture/

multi-task learning for word-level and sentence-level QE (Caruana, 1997). The two tasks are related as word-level quality contributes to the sentence-level quality in general. Therefore we believe that a multi-task architecture that can learn both tasks simultaneously can be advantageous.

As discussed at multiple points in this thesis, pre-trained models based on transformers are big in size. Therefore, managing several of them for each language pair would be chaotic in a real-world application for word-level QE. As a solution for that, we explore multilingual word-level QE models in Chapter 11 in Part III of the thesis.

Chapter 11

Multilingual Quality Estimation with TransQuest

Machine translation quality estimation is traditionally framed as a supervised machine learning problem (Kepler et al., 2019; Lee, 2020) where the machine learning models are trained on language specific data for quality estimation. We refer to these models as bilingual QE models. This process would require having annotated QE data for all the language pairs. Furthermore, this language specific supervised machine learning process would result in having machine learning models for each language pair separately.

This traditional approach has obvious drawbacks. As mentioned before, this process requires training data for each language pair. However, the training data publicly available to build QE models is limited to very few language pairs, making it difficult to build QE models for many languages. Furthermore, from an application perspective, even for the languages with resources, it is difficult to maintain separate QE models for each language since the state-of-the-art neural QE models are large in size (Ranasinghe et al., 2020c).

To understand this scale, consider a real-world application where it is required to build a quality estimation solution for the European Parliament. The

European Parliament has 24 languages, resulting in 24*23 language pairs equal to 552 language pairs. A traditional bilingual QE solution would require 552 training datasets to train the models, which is highly challenging and costly to collect and annotate. Furthermore, this would require having 552 machine learning models. State-of-the-art QE models like TransQuest are at least 2GB in size. Having 2GB sized 552 models in the RAM at inferencing time would not be practical. The solution to all these problems is Multilingual QE models.

Multilingual models allow training a single model to perform a task from and/or to multiple languages. Even though multilingual learning has been applied to many tasks (Ranasinghe and Zampieri, 2020; Ranasinghe and Zampieri, 2021) including NMT (Nguyen and Chiang, 2017; Aharoni et al., 2019), multilingual approaches have been rarely used in QE (Sun et al., 2020a). Therefore, in this chapter we explore multilingual models with the *TransQuest* architectures we introduced in Chapters 9 and 10 for sentence-level QE and word-level QE respectively. Since we used a cross-lingual transformer model that supports 104 languages (Conneau et al., 2020) it is possible to explore multilingual learning with the same setup.

Usually, the neural machine learning models are *hungry for data*. They need a lot of annotated data for the training process, which is a challenge for the low resource languages. Recently, researchers are trying to exploit this behaviour with learning paradigms such as few-shot and zero-shot learning. What we define as few-shot learning in this chapter is the process where the QE model only sees a few instances from a certain language pair in the training

process (Wang et al., 2020b) while in zero-shot learning, QE model would not see any instances from a certain language pair (Chang et al., 2008) in the training process. Even though few-shot and zero-shot learning has been popular in machine learning applications, including NLP, they have not been explored with QE. Exploring them would benefit the low resource languages, which the QE training data is difficult to find. Therefore, this chapter inspects how the bilingual and multilingual QE models behave in few-shot and zero-shot learning environments.

As far as we know, this is the first study done on multilingual word-level and sentence-level QE. We address three research questions in this chapter:

RQ1: Do multilingual models based on existing state-of-the-art sentence-level, and word-level QE architectures perform competitively with the related bilingual models?

RQ2: How do the bilingual and multilingual models perform in a zero-shot environment, and what is the impact of source-target direction, domain and MT type for zero-shot learning?

RQ3: Do multilingual QE models perform better with a limited number of training instances (Few-shot learning) for an unseen language pair?

The main contributions of this Chapter are,

1. We explore multilingual, sentence-level and word-level quality estimation with the proposed architectures in *TransQuest*. We show that multilingual models are competitive with bilingual models.

2. We inspect few-shot and zero-shot sentence-level and word-level quality estimation with the bilingual and multilingual models. We report how the source-target direction, domain and MT type affect the predictions for a new language pair.

3. The code and the pre-trained multilingual models of *TransQuest* are publicly available to the community[1].

The rest of this chapter is organised as follows. Section 11.1 discusses the methodology and the multilingual experiments done with 15 language pairs in sentence-level QE and word-level QE. Section 11.2 shows the results and Sections 11.2.2 and 11.2.3 provide further analysis on zero-shot and few-shot learning. The chapter finishes with conclusions and ideas for future research directions in multilingual QE.

11.1 Methodology

As mentioned before, we conducted the experiments with the architectures explored in Chapters 9 and 10. For the multilingual sentence-level experiments, we used the *MonoTransQuest* architecture introduced in Chapter 9 which outperformed other open-source QE frameworks and best systems submitted to the shared tasks in the majority of the language pairs. We experimented with both aspects of sentence-level QE, HTER and DA with all the datasets introduced in Chapter 8. For the word-level experiments, we used the *MicroTransQuest*

[1]The pre-trained multilingual QE models are available on the HuggingFace model repository on https://huggingface.co/TransQuest

architecture which again outperformed other open-source QE frameworks and best systems submitted to the shared tasks in the majority of the language pairs in word-level QE. As the word-level QE data, we considered all the word-level QE datasets introduced in Chapter 8.

For the experiments, we considered XLM-R large model and did not use the ensemble models to keep the multilingual experiments simpler. We applied the same set of configurations for all the training processes to ensure consistency between the experiments. We used a batch size of eight, Adam optimiser and a linear learning rate warm-up over 10% of the training data. During the training process, the parameters of the XLM-R-large model, as well as the parameters of the subsequent layers, were updated. The models were trained using only training data. Furthermore, they were evaluated while training once in every 100 training steps using an evaluation set with one-fifth of the training data rows. We performed early stopping if the evaluation loss did not improve over ten evaluation steps. All the models were trained for three epochs. We used a learning rate of $2e^{-5}$ for the sentence-level experiments while for word-level experiments it was $1e^{-5}$.

11.2 Results

In the following sections, we explore different multilingual QE settings and compare the multilingual results to the results we got with supervised, bilingual QE models.

For the sentence-level evaluations, we used Pearson correlation, the same

evaluation metric we used for bilingual sentence-level QE in Chapter 9, which was used for WMT QE shared tasks. Results for multilingual sentence-level QE experiments with HTER and DA are shown in Tables 11.1 and 11.2. For the word-level evaluations, we used the same evaluation metrics we used for bilingual word-level QE in Chapter 10 which in turn was used for the WMT QE shared tasks. Results for multilingual word-level QE experiments with regards to $F1_{MULTI}$ for words in target ($F1_{MULTI}$ $Target$), $F1_{MULTI}$ for gaps in the target ($F1_{MULTI}$ $GAPS$) and $F1_{MULTI}$ for words in source ($F1_{MULTI}$ $Source$) are shown in Tables 11.3, 11.4 and 11.5 respectively.

The values displayed diagonally across the Row I of Tables 11.1, 11.2, 11.3, 11.4 and 11.5 show the results for supervised, bilingual, QE models where the model was trained on the training set of a particular language pair and tested on the test set of the same language pair. This is the exact result we reported in the Row I of Tables 9.1, 9.2, 10.1, 10.2 and 10.3 in Chapters 9 and 10, which we got with *MonoTransQuest* and *MicroTransQuest* using XLM-R-large model (Conneau et al., 2020) for sentence-level and word-level.

11.2.1 Multilingual QE

First, We combined instances from the training sets of all the language pairs where the sentence-level HTER data was available and built a single QE model with *MonoTransQuest*. We evaluated this multilingual model on the test sets of all the language pairs. We repeat the same for sentence-level DA QE with *MonoTransQuest* and word-level QE with *MicroTransQuest*. Our results, displayed

		IT			Pharmaceutical			Wiki	
	Train Language(s)	En-Cs SMT	En-De SMT	En-Ru NMT	De-En SMT	En-LV NMT	En-Lv SMT	En-De NMT	En-Zh NMT
I	En-Cs SMT	**0.7207**	(-0.06)	(-0.07)	(-0.13)	(-0.02)	(-0.01)	(-0.11)	(-0.10)
	En-De SMT	(-0.01)	0.7137	(-0.04)	(-0.12)	(-0.04)	(-0.05)	(-0.07)	(-0.07)
	En-Ru NMT	(-0.12)	(-0.15)	**0.7126**	(-0.13)	(-0.01)	(-0.02)	(-0.08)	(-0.07)
	De-En SMT	(-0.39)	(-0.29)	(-0.34)	**0.7939**	(-0.27)	(-0.31)	(-0.26)	(-0.27)
	En-LV NMT	(-0.11)	(-0.13)	(-0.02)	(-0.11)	0.7394	(-0.01)	(0.08)	(-0.07)
	En-Lv SMT	(-0.03)	(-0.09)	(-0.08)	(-0.15)	(-0.01)	0.6592	(-0.13)	(-0.13)
	En-De NMT	(-0.11)	(-0.07)	(-0.02)	(-0.12)	(-0.01)	(-0.02)	0.5994	(-0.04)
	En-Zh NMT	(-0.21)	(-0.18)	(-0.02)	(-0.18)	(-0.02)	(-0.07)	(-0.08)	0.6119
II	All	0.7111	**0.7300**	0.7012	0.7878	**0.7450**	**0.7141**	0.5982	0.6092
	All-1	(-0.01)	(-0.04)	(-0.02)	(-0.11)	(-0.01)	(-0.01)	(-0.01)	(-0.03)
III	Domain	0.7001	0.7256	0.6987	0.7754	0.7412	0.7065	0.5764	0.5671
IV	SMT/NMT	0.6998	0.7143	0.6998	0.7642	0.7319	0.6872	0.5671	0.5601
V	Quest++	0.3943	0.3653	NR	0.3323	0.4435	0.3528	NR	NR
	OpenKiwi	NR	NR	0.5923	NR	NR	NR	0.3923	0.5058
	Best system	0.6918	0.7397	0.5923	0.7888	0.6819	0.6188	**0.7582**	**0.6641**

Table 11.1: Pearson correlation (ρ) between the *MonoTransQuest* predictions and human post-editing effort in multilingual experiments. The best result for each language by any method is highlighted in bold. Rows **I**, **II**, **III** and **IV** indicate the different multilingual settings. Row **V** shows the results of the baselines and the best system submitted for the language pair in that competition. **NR** implies that a particular result was *not reported* by the organisers. Zero-shot results are coloured in grey and the value shows the difference between the best result in that Row for that language pair and itself.

in Row II ("All") of Tables 11.1, 11.2, 11.3, 11.4 and 11.5 show that multilingual models perform on par with the bilingual models or even better for some language pairs in all the evaluation metrics with both sentence-level and word-level. For example, in sentence-level HTER experiments as shown in Table 11.1 multilingual sentence-level QE model outperforms the bilingual sentence-level QE model in three language pairs; En-De SMT, En-Lv SMT and En-Lv NMT. In word-level also, as shown in Table 11.3, multilingual word-level QE model outperforms the bilingual word-level QE model in all the language pairs except

		Low-resource		Mid-resource			High-resource	
	Train Language(s)	Si-En	Ne-En	Et-En	Ro-En	Ru-En	En-De	En-Zh
I	Si-En	0.6525	(-0.05)	(-0.08)	(-0.15)	(-0.07)	(-0.13)	(-0.13)
	Ne-En	(-0.10)	0.7914	(-0.06)	(-0.08)	(-0.08)	(-0.10)	(-0.11)
	Et-En	(-0.07)	(-0.10)	0.7748	(-0.20)	(-0.08)	(-0.10)	(-0.08)
	Ro-En	(-0.02)	(-0.04)	(-0.02)	**0.8982**	(-0.08)	(-0.10)	(-0.14)
	Ru-En	(-0.11)	(-0.16)	(-0.19)	(-0.26)	0.7734	(-0.04)	(-0.09)
	En-De	(-0.32)	(-0.51)	(-0.39)	(-0.51)	(-0.35)	**0.4669**	(-0.17)
	En-Zh	(-0.16)	(-0.24)	(-0.19)	(-0.36)	(-0.17)	(-0.02)	**0.4779**
II	All	0.6526	0.7581	0.7574	0.8856	0.7521	0.4420	0.4646
	All-1	(-0.02)	(-0.02)	(-0.02)	(-0.03)	(-0.02)	(-0.02)	(-0.05)
III	OpenKiwi	0.3737	0.3860	0.4770	0.6845	0.5479	0.1455	0.1902

Table 11.2: Pearson correlation (ρ) between the *MonoTransQuest* predictions and human DA judgments in multilingual experiments. The best result for each language by any method is highlighted in bold. Rows **I** and **II** indicate the different multilingual settings. Row **III** shows the results of the baselines and the best system submitted for the language pair in that competition. Zero-shot results are coloured in grey and the value shows the difference between the best result in that Row for that language pair and itself.

En-Zh NMT, En-Ru NMT and De-En SMT with regard to Target F1-Multi. Similar observations can be made in other word-level evaluation metrics in Tables 11.4 and 11.5.

We also investigate whether combining language pairs that share either the same domain or MT type can be more beneficial since it is possible that the learning process is better when language pairs share certain characteristics. We could only conduct this experiment in sentence-level HTER QE and word-level QE as sentence-level DA QE datasets are from the same domain and MT type. However, as shown in Row III and IV of Tables 11.1, 11.3, 11.4 and 11.5, for

		IT			Pharmaceutical			Wiki	
	Train Language(s)	En-Cs SMT	En-De SMT	En-Ru NMT	De-En SMT	En-LV NMT	En-Lv SMT	En-De NMT	En-Zh NMT
I	En-Cs SMT	0.6081	(-0.07)	(-0.09)	(-0.15)	(-0.02)	(-0.01)	(-0.10)	(-0.11)
	En-De SMT	(-0.01)	0.6348	(-0.07)	(-0.14)	(-0.06)	(-0.04)	(-0.06)	(-0.09)
	En-Ru NMT	(-0.14)	(-0.16)	**0.5592**	(-0.12)	(-0.01)	(-0.03)	(-0.09)	(-0.08)
	De-En SMT	(-0.43)	(-0.33)	(-0.31)	**0.6485**	(-0.29)	(-0.32)	(-0.25)	(-0.28)
	En-LV NMT	(-0.12)	(-0.14)	(-0.03)	(-0.12)	0.5868	(-0.01)	(0.09)	(-0.08)
	En-Lv SMT	(-0.04)	(-0.10)	(-0.09)	(-0.16)	(-0.01)	0.5939	(-0.15)	(-0.14)
	En-De NMT	(-0.11)	(-0.08)	(-0.02)	(-0.14)	(-0.02)	(-0.04)	0.5013	(-0.06)
	En-Zh NMT	(-0.19)	(-0.17)	(-0.03)	(-0.16)	(-0.03)	(-0.06)	(-0.07)	0.5402
II	All	**0.6112**	**0.6583**	0.5558	0.6221	**0.5991**	**0.5980**	0.5101	0.5229
	All-1	(-0.01)	(-0.05)	(-0.02)	(-0.12)	(-0.01)	(-0.01)	(-0.01)	(-0.05)
III	Domain	0.6095	0.6421	0.5560	0.6331	0.5892	0.5951	0.5021	0.5210
IV	SMT/NMT	0.6092	0.6410	0.5421	0.6320	0.5885	0.5934	0.5010	0.5205
V	Marmot	0.4449	0.3630	NR	0.4373	0.4208	0.3445	NR	NR
	OpenKiwi	NR	NR	0.2412	NR	NR	NR	0.4111	0.5583
	Best system	0.4449	0.6246	0.4780	0.6012	0.4293	0.3618	**0.6186**	**0.6415**

Table 11.3: $F1_{MULTI}$ *Target* between the *MicroTransQuest* predictions and human annotations in multilingual experiments. The best result for each language by any method is highlighted in bold. Row **I**, **II**, **III** and **IV** indicate the different multilingual settings. Row **V** shows the results of the baselines and the best system submitted for the language pair in that competition. **NR** implies that a particular result was *not reported* by the organisers. Zero-shot results are coloured in grey and the value shows the difference between the best result in that Row for that language pair and itself.

the majority of the language pairs, specialised multilingual QE models built on certain domains or MT types do not perform better than multilingual models which contain all the data.

With these observations, we answer our **RQ1:** Multilingual models based on existing state-of-the-art QE architectures perform competitively with the related bilingual models, and in some of the language pairs, multilingual models even outperformed the related bilingual models.

		IT		Pharmaceutical		
	Train Language(s)	En-Cs SMT	En-De SMT	De-En SMT	En-LV NMT	En-Lv SMT
I	En-Cs SMT	0.2018	(-0.08)	(-0.15)	(-0.02)	(-0.01)
	En-De SMT	(-0.08)	0.4927	(-0.14)	(-0.06)	(-0.04)
	En-Ru NMT	(-0.14)	(-0.15)	(-0.12)	(-0.01)	(-0.03)
	De-En SMT	(-0.18)	(-0.33)	**0.4203**	(-0.29)	(-0.32)
	En-LV NMT	(-0.16)	(-0.15)	(-0.12)	0.1664	(-0.01)
	En-Lv SMT	(-0.11)	(-0.11)	(-0.16)	(-0.01)	0.2356
	En-De NMT	(-0.17)	(-0.09)	(-0.14)	(-0.02)	(-0.04)
	En-Zh NMT	(-0.15)	(-0.16)	(-0.16)	(-0.03)	(-0.06)
II	All	**0.2118**	**0.5028**	0.4189	**0.1772**	**0.2388**
	All-1	(-0.03)	(-0.08)	(-0.14)	(-0.01)	(-0.01)
III	Domain	0.2112	0.4951	0.4132	0.1685	0.2370
IV	SMT/NMT	0.2110	0.4921	0.4026	0.1671	0.2289
V	Marmot	NS	NS	NS	NS	NS
	Best system	0.1671	0.3161	0.3176	0.1598	0.1386

Table 11.4: $F1_{MULTI}$ *GAPS* between *MicroTransQuest* predictions and human annotations in multilingual experiments. The best result for each language by any method is highlighted in bold. Row **I**, **II**, **III** and **IV** indicate the different multilingual settings. Row **V** shows the results of the baselines and the best system submitted for the language pair in that competition. **NS** implies that a particular result was *not supported* by the respective baseline. Zero-shot results are coloured in grey and the value shows the difference between the best result in that Row for that language pair and itself.

11.2.2 Zero-shot QE

We performed zero-shot quality estimation to test whether a QE model trained on a particular language pair can be generalised to other language pairs, different domains and MT types. We used the QE model trained on a particular language pair and evaluated it on the test sets of the other language pairs. Non-diagonal values of Row I in Tables 11.1, 11.2, 11.3, 11.4 and 11.5 show how each QE model

		IT			Pharmaceutical			Wiki	
	Train Language(s)	En-Cs SMT	En-De SMT	En-Ru NMT	De-En SMT	En-LV NMT	En-Lv SMT	En-De NMT	En-Zh NMT
I	En-Cs SMT	0.5327	(-0.07)	(-0.09)	(-0.17)	(-0.02)	(-0.01)	(-0.12)	(-0.13)
	En-De SMT	(-0.01)	0.5269	(-0.08)	(-0.14)	(-0.06)	(-0.05)	(-0.08)	(-0.09)
	En-Ru NMT	(-0.14)	(-0.18)	**0.5543**	(-0.14)	(-0.01)	(-0.03)	(-0.09)	(-0.08)
	De-En SMT	(-0.42)	(-0.33)	(-0.31)	**0.4824**	(-0.29)	(-0.32)	(-0.23)	(-0.28)
	En-LV NMT	(-0.12)	(-0.14)	(-0.03)	(-0.12)	0.4880	(-0.01)	(0.09)	(-0.08)
	En-Lv SMT	(-0.04)	(-0.11)	(-0.09)	(-0.17)	(-0.02)	0.4945	(-0.15)	(-0.14)
	En-De NMT	(-0.11)	(-0.08)	(-0.02)	(-0.15)	(-0.03)	(-0.04)	0.4456	(-0.06)
	En-Zh NMT	(-0.19)	(-0.17)	(-0.03)	(-0.18)	(-0.05)	(-0.06)	(-0.07)	0.4040
II	All	**0.5442**	**0.5445**	0.5535	0.4791	**0.4983**	**0.5005**	0.4483	0.4053
	All-1	(-0.02)	(-0.06)	(-0.03)	(-0.16)	(-0.01)	(-0.01)	(-0.01)	(-0.04)
III	Domain	0.5421	0.5421	0.5259	0.4672	0.4907	0.4991	0.4364	0.4021
IV	SMT/NMT	0.5412	0.5412	0.5230	0.4670	0.4889	0.4932	0.4302	0.4012
V	Marmot	NS	NS	NR	NS	NS	NS	NR	NR
	OpenKiwi	NR	NR	0.2647	NR	NR	NR	0.3717	0.3729
	Best system	0.3937	0.3368	0.4541	0.3200	0.3614	0.4945	**0.5672**	**0.4462**

Table 11.5: $F1_{MULTI}$ *Source* between *MicroTransQuest* predictions and human annotations in multilingual experiments. The best result for each language by any method is highlighted in bold. Row **I**, **II**, **III** and **IV** indicate the different multilingual settings. Row **V** shows the results of the baselines and the best system submitted for the language pair in that competition. **NR** implies that a particular result was *not resported* by the organisers and **NS** implies that a particular result was *not supported* by the respective baseline. Zero-shot results are coloured in grey and the value shows the difference between the best result in that row for that language pair and itself.

performed on other language pairs. For better visualisation, the non-diagonal values of Row I in Tables 11.1, 11.2, 11.3, 11.4 and 11.5 show by how much the score changes when the zero-shot QE model is used instead of the bilingual QE model. As can be seen, the scores decrease, but this decrease is negligible and is to be expected. For most pairs, the QE model that did not see any training instances of that particular language pair outperforms the baselines that were trained extensively on that particular language pair. Further analysis of the

results shows that zero-shot QE performs better when the language pair shares some properties such as domain, MT type or language direction. For example, in word-level QE, En-De SMT ⇒ En-Cs SMT is better than En-De NMT ⇒ En-Cs SMT and En-De SMT ⇒ En-De NMT is better than En-Cs SMT ⇒ En-De NMT in $F1_{MULTI}$ $Target$. Similar observations can be made on other evaluation metrics too.

We also experimented with zero-shot QE with multilingual QE models. For sentence-level HTER QE, sentence-level DA QE and word-level QE separately, we trained a multilingual model in all the language pairs except one and performed prediction on the test set of the language pair left out. In Row II ("All-1") of Tables 11.1, 11.2, 11.3, 11.4 and 11.5, we show its difference to the multilingual QE model. This also provides competitive results for the majority of the languages, proving it is possible to train a single multilingual QE model and extend it to a multitude of languages and domains. This approach provides better results than performing transfer learning from a bilingual model.

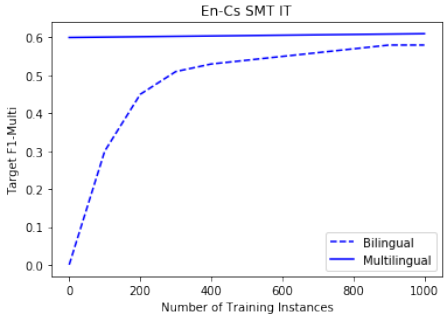
One limitation of the zero-shot QE is its inability to perform when the language direction changes. In the scenario where we performed zero-shot learning from De-En SMT to other language pairs in sentence-level HTER QE and word-level QE, results degraded considerably from the bilingual result. Similarly, the performance is rather poor when we test De-En for the multilingual zero-shot experiment as the direction of all the other pairs used for training differs. These observations are similar in sentence-level DA experiments with En-De and En-Zh.

With these observations, we answer our **RQ2:** zero-shot QE with state-of-the-art QE models provide very competitive results to language pairs which they did not see in the training process. Furthermore, multilingual models provide better zero-shot results than bilingual models.

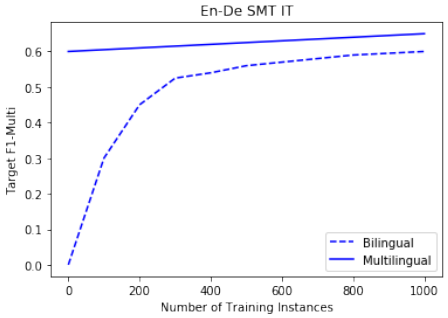
11.2.3 Few-shot QE

We also evaluated how the QE models behave with a limited number of training instances. For each language pair, we initiated the weights of the bilingual model with those of the relevant All-1 QE and trained it on 100, 200, 300 and up to 1000 training instances. We compared the results with those obtained, having trained the QE model from scratch for that language pair. The results in Figure 11.1 show that All-1 or the multilingual model performs well above the QE model trained from scratch (Bilingual) when there is a limited number of training instances available. Even for the De-En language pair in sentence-level HTER QE and word-level QE, for which we had comparatively poor zero-shot results, the multilingual model provided better results with a few training instances. It seems that having the model weights already fine-tuned in the multilingual model provides an additional boost to the training process, which is advantageous in a few-shot scenario.

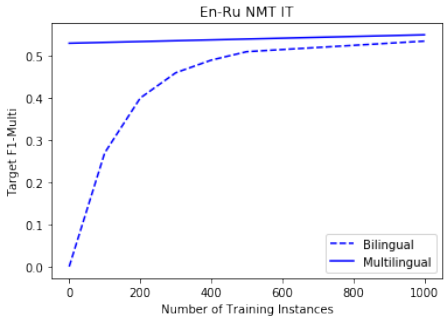
With these findings, we answer our **RQ3:** multilingual QE models perform better with a limited number of training instances (Few-shot learning) for an unseen language pair in both sentence-level and word-level QE. It is always better to transfer the weights from a multilingual QE model than to train the



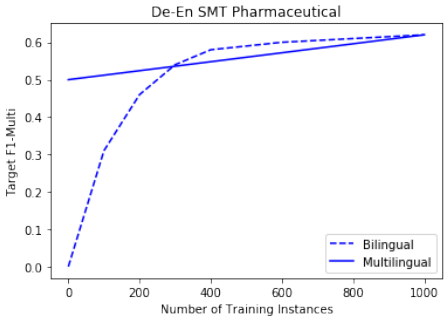
(a) En-Cs SMT IT



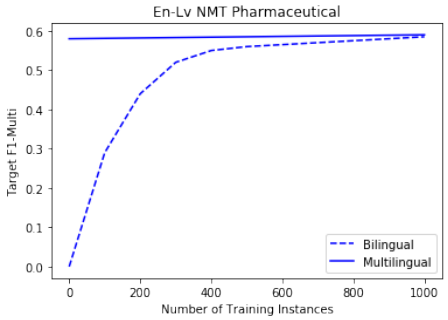
(b) En-De SMT IT



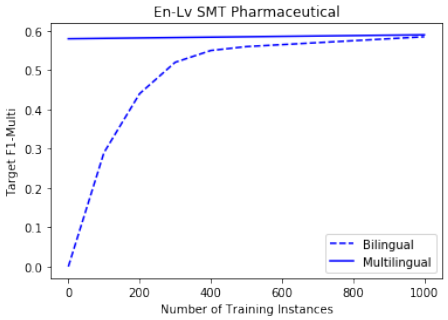
(c) En-Ru NMT IT



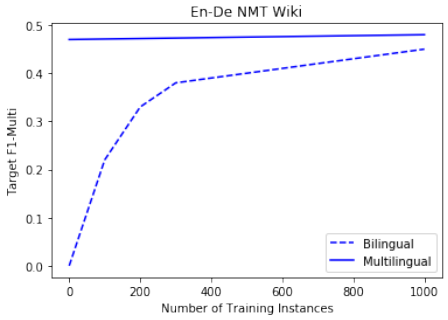
(d) De-En SMT Pharmaceutical



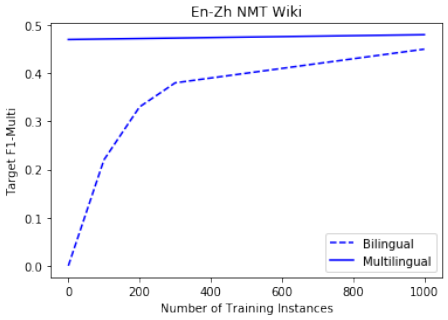
(e) En-Lv NMT Pharmaceutical



(f) En-Lv SMT Pharmaceutical



(g) En-De NMT Wiki



(h) En-Zh NMT Wiki

Figure 11.1: Few-shot learning Results for Word-Level QE. We report $F1_{MULTI}$ *Target* scores against Number of training instances for multilingual and bilingual models.

weights from scratch for a new language pair.

11.3 Conclusion

The traditional way of having a single QE model for each language pair has many limitations; *i.* They need to have annotated training data for each language pair which can be costly, *ii.* Managing several QE models at the same time can be chaotic. These limitations can hinder the ability of the state-of-the-art QE models to be applied in real-world applications. As a solution to that, we explored multilingual QE with state-of-the-art QE models. We used the sentence-level and word-level QE architectures in *TransQuest* and evaluated them in different multilingual settings.

In our experiments, we observed that multilingual QE models deliver excellent results on the language pairs they were trained on. In addition, the multilingual QE models perform well in the majority of the zero-shot scenarios where the multilingual QE model is tested on an unseen language pair. Furthermore, multilingual models perform very well with few-shot learning on an unseen language pair compared to training from scratch for that language pair, proving that multilingual QE models are effective even with a limited number of training instances. This suggests that we can train a single multilingual QE model on as many languages as possible and apply it to other language pairs. These findings can be beneficial to perform QE in low-resource languages for which the training data is scarce and when maintaining several QE models for different language pairs is arduous. Considering the benefits of

multilingual models, we have released several multilingual sentence-level and word-level pre-trained models on HuggingFace model hub.

The main limitation of our multilingual evaluation is that all the languages we used throughout the experiments are supported by the pre-trained XLM-R model we used. XLM-R large model only supports 100 languages at the moment, and there is a lot of low resource but common languages such as Chewa, Tajiki, Tigrinya[2] etc. that XLM-R does not support. A question can arise about how the language pairs that XLM-R does not support perform in our multilingual QE environment. However, as far as we know, until very recently, there were no annotated QE datasets either for the languages outside the 100 languages supported by XLM-R. Therefore, it would not be possible to carry out a proper evaluation. Very recently, in WMT 2021, an annotated QE dataset for Pashto-English and Khmer-English was introduced. XLM-R does not support Pashto and Khmer at the moment, and it would be interesting to experiment with them with our multilingual models, which we hope to do in future work.

Pre-trained multilingual transformer models are rapidly increasing in popularity in the NLP community. From them, one notable multilingual transformer model is mT5 (Xue et al., 2021); multilingual text to text transformer model, which considers every task as a sequence to sequence task. It has provided very good results in a variety of multilingual NLP tasks. As future work, we hope to incorporate mT5 in *TransQuest* framework and evaluate it in a multilingual QE

[2]Chewa, Tajiki and Tigrinya are the official languages of Zimbabwe, Tajikistan and Eritrea respectively that are collectively spoken by more than 30 million people in the world

environment.

With this, we conclude Part III of the thesis, using deep learning based STS metrics in translation quality estimation. We showed that the state-of-the-art STS methods we experimented in Part I of the thesis can be employed successfully in QE. Our method outperform current neural QE models such as OpenKiwi and DeepQuest in word-level and sentence-level QE setting a new state-of-the-art. Furthermore, they are simple compared to the complex neural architectures employed in QE in recent years. We believe that the findings of Part III of the thesis would open a new direction in QE.