Deep learning based Semantic Textual Similarity for Applications in Translation Technology

Tharindu Ranasinghe

A thesis submitted in partial fulfilment of the requirements of the University of Wolverhampton for the degree of Doctor of Philosophy

2021

Signature: . . . . . .

Date: . . . . . .

Abstract

Acknowledgements

Contents

List of Tables

List of Figures

List of Publications

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

-

Introduction

Semantic textual similarity (STS) is a natural language processing (NLP) task to quantitatively assess the semantic similarity between two text snippets. STS is a fundamental NLP task for many text-related applications, including text deduplication, paraphrase detection, semantic searching, and question answering. Measuring STS is a machine learning (ML) problem, where a ML model predicts a value that represents the similarity of the two input texts. These machine learning models can be categorised in to two main areas; supervised and unsupervised. Supervised STS ML models have been trained on an annotated STS dataset while the unsupervised STS ML models predict STS without being trained on annotated STS data. In the first part of the thesis we explore supervised and unsupervised ML models in STS. We explore embedding aggregation based state-of-the-art 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. On the process, we develop new state-of-the-art unsupervised STS method based on contextual word embeddings and new state-of-the-art supervised STS method based on Siamese neural networks.

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

(TM) and translation quality estimation (QE). We identify that the edit distance based matching and retrieval algorithms in TMs are less efficient and we propose a TM matching algorithm based on the STS methods we developed in the first part of the thesis. We empirically show that this algorithm outperforms edit distance based matching algorithms. As the next application, we utilise the STS architectures we developed, in translation quality estimation. We show that STS architectures can be successfully applied in QE by changing the input embeddings in to cross-lingual embeddings. Based on that , we develop TransQuest - a new state-of-the-art QE framework that won the WMT 2020 QE shared task. We release TransQuest as an open-source library and by the time of writing this, TransQuest has more than 8,000 downloads from the community.

Part I

Semantic Textual Similarity

Chapter 1

Introduction

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 and one of the fundamental tasks for many NLP applications and its related areas. To measure STS, the input is always two textual segments and the output is a continues value that represents the degree of similarity of the two input textual segments.

STS is related to both textual entailment (TE) and paraphrasing, but differs in a number of ways. TE can draw three directional relationships between two text fragments while the task considered two text fragments as text (t) and hypothesis (h) respectively. On the other hand, paraphrasing identification is the task of recognising text fragments with approximately the same meaning within a specific context. Therefore, TE and paraphrasing gives a categorical output while STS identifies the degree of equivalence of texts as a continues value.

Measuring STS is an important research problem, having many applications in NLP such as information retrieval (IR) [1, 2], text summarisation [3, 4], question answering [5], relevance feedback [6], text classification [7, 8] and word

sense disambiguation [9]. 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 [4, 10]. Also, STS is widely used in semantic web applications like community extraction [11], Twitter search [12] where it is required the ability to accurately measure semantic relatedness between concepts or entities.

These applications require to measure STS automatically which means that computer programs should be developed to calculate STS between two textual inputs. This can be considered as a Machine Learning (ML) problem. Over the years, researchers have proposed numerous ML solutions for STS. These ML solutions can be categorised in to two main categories; (a) Linguistic feature based (b) Vector/ Embedding based . Most of the early approaches belong to linguistic feature based category. With this, features to the ML algorithm were handcrafted. Such features include edge-distances between nodes in WordNet [13], number of named entities in two input texts, corpus pattern analysis features etc. Then these features would be fed in to a ML algorithm such as Support Vector Machine (SVM), Linear Regression etc [14]. 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 fea-

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

ture 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. However, the biggest limitation of these methods would be, they no longer provide strong results compared to the vector based methods.

With the introduction of word embeddings [15], ML solutions in NLP shifted from feature based methods to vector based methods. Pre-trained word embedding models provide a learned representation for texts where the words that have the same meaning have a similar representation. Since these word embeddings are already semantically powerful, ML solutions no longer requires 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 the state-of-the-art in NLP tasks including STS too providing stronger results than feature based ML solutions. 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 in to two main categories; Supervised and Unupervised. In supervised learning, ML models will be trained using labelled data. It means that data is already annotated by the humans with the correct answer. For unsupervised learning, you do not need an annotated dataset. Unsupervised ML approach

would discover features by itself. Given that annotated STS data is not commonly available in many languages and domains it is important to explore both supervised and unsupervised STS methods. Therefore, first two chapters in this part of the thesis explore unsupervised STS methods while the last two chapters in this part explore supervised STS methods.

Most common unsupervised STS approaches are vector aggregation methods like Word Vector Averaging, Word Mover's Distance [16] and Smooth Inverse Frequency [17]. In Chapter 2 we explore them in detail. We identify the best vector aggregation method empirically by analysing them in different STS datasets and 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 Chapter 4 and 5 we explore most popular supervised STS approaches. Usually, in supervised vector based STS approaches, word embeddings would be fed in to a neural network. There are popular neural network structures when it comes to STS. Tree-structured neural networks [18] and Siamese neural networks are such structures [19]. Among them Siamese neural networks have been

widely used in STS and has 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 asses its performance on different languages and domains.

In the final chapter of the Part I of this thesis, we explore the newly released transformers in STS tasks. Transformers have taken NLP field by storm providing very successful results in variety of NLP tasks. In Chapter 5, we bring together various transformer architectures like BERT [20], XLNet [21], RoBERTa [22] etc and investigate their performance in various STS datasets. We explore the strengths and weaknesses of transformer models regarding the accuracy and efficiency and discover the possible solutions for its limitations.

The remainder of this chapter is structured as follows. Section 1.1 discuss the various datasets we used in *"Semantic Textual Similarity"* part of the thesis. We also briefly analyse the datasets for common properties. In Section 1.3 we discuss the main contributions we have to the community with the *"Semantic Textual Similarity"* part of the thesis. Chapter concludes 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 [23, 24, 25, 26, 27, 28]. First, they have provided annotated datasets which can be used to train STS ML models and to eval-

uate 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.

We experimented with several datasets throughout the experiments in the Semantic Textual Similarity Section that has been released for these shared tasks. In order 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 interesting characteristics. Therefore, with the introduction we also do an exploratory analysis of the dataset focussing on different properties of the dataset. 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 [29]. The dataset has two types of annotations: Semantic Relatedness and Textual Entailment. We only use Semantic Relatedness annotations in our research. SICK was built starting from two existing datasets: the 8K ImageFlickr

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

data set [3] [30] and the SemEval-2012 STS MSR-Video Descriptions dataset [4] [23]. 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 have been randomly chosen from this data set to be used in SICK.

In order to generate SICK data from the 1,500 sentence pairs taken from the source data sets, a 3-step process has been applied to each sentence composing the pair, namely *(i) normalisation, (ii) expansion and (iii) pairing* [29]. 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, in order 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

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

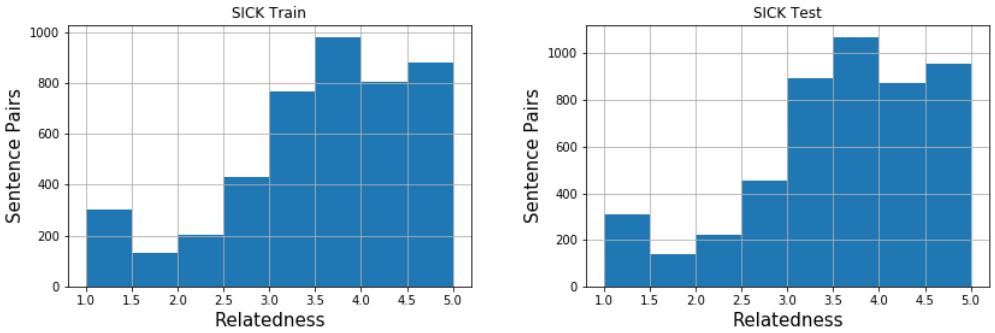
[4]The SemEval-2012 STS MSR-Video Descriptions dataset is available at https://www.cs.york.ac.uk/semeval-2012/task6/index.html

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

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, a number of pairs composed of completely unrelated sentences have been added to the data set by randomly taking two sentences from two different pairs [29].

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 each pair has been evaluated by 10 different annotators. 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 [29]. 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.

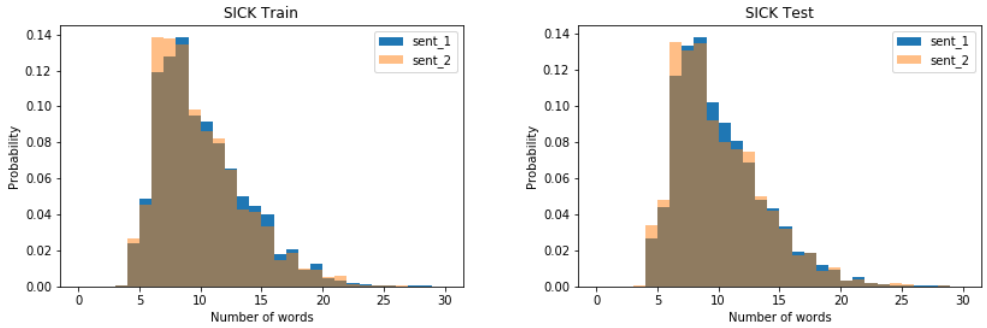
Figure 1.1 shows the distribution of the relatedness value in SICK training and SICK testing set. It is clear that there are more sentence pairs with a high relatedness values compared to low relatedness values. SICK train and SICK test follows a similar distribution.



(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.



(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.

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

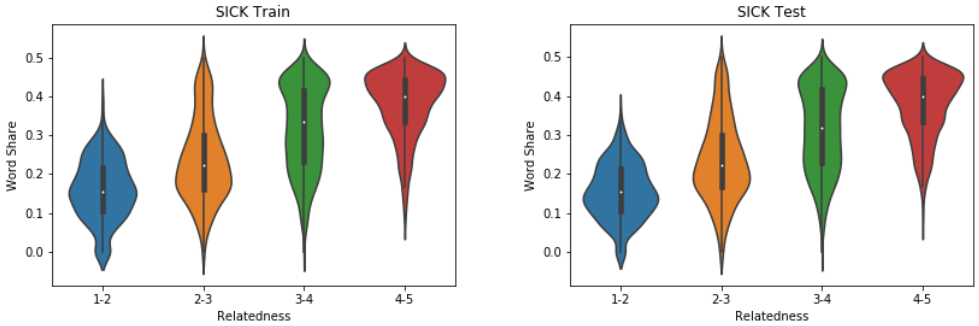
Table 1.1: Example sentence pairs from the SICK dataset with their gold relatedness scores (on a 5-point rating scale). **Sentence Pair** column shows the two sentence and **Relatedness** column denotes the annotated relatedness score.

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

In Figure 1.2 we visualise the normalised distribution of word count for both sentence 1 and sentence 2 in SICK train and SICK test. Both sentences have a similar distribution reaching the maximum around 9 words. SICK train and SICK test follows a similar pattern in word count 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 SICK test too.

The common judgement in STS is that, when two sentences share a large



(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

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 [31, 32, 33, 34]. Systems like Vilariño et al. [31], Lynum et al. [33] use the number of words common in two sentences as a feature directly while systems like Gupta et al. [32], Chávez et al. [34] 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" re-

[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.

latedness score bin, there are some sentence pairs with a high word share. Most common example for such a case would be sentence 2 is the complete negation of the sentence 1. In such cases the two sentences share a large potion of the words and one sentence have the *"not"* word that gives a complete 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 does not contain the same words but will be having synonyms and possess the same overall meaning. Therefore, the STS methods that focusses on word share won't perform well in SICK dataset.

A clear strength in the SICK dataset is that training set and the testing set reflects similar properties so that a properly trained machine learning model on SICK train should give good results to the SICK test set as well.

2. **STS 2017 English Dataset** [7] The second English STS dataset we used to experiment in this section 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 [28]. 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 [23, 24, 25, 26, 27]. Once combined we had 8277 sentence pairs for training. More information about the datasets used to

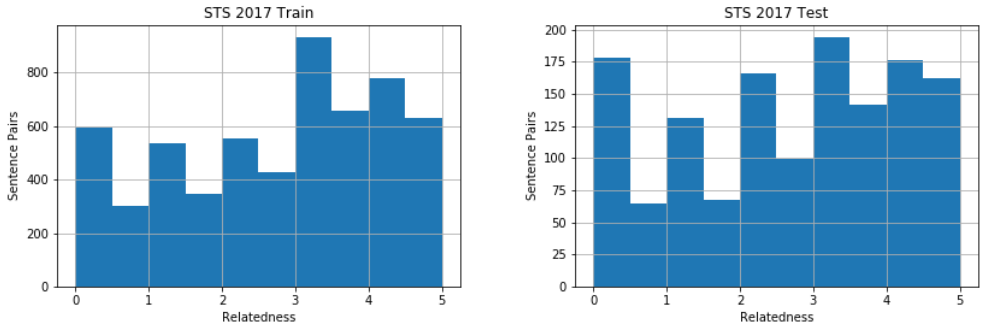
[7] The STS 2017 English Dataset is available to download at http://ixa2.si.ehu.es/stswiki/

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 [28]. The Stanford Natural Language Inference (SNLI) corpus [35] 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 gold score has been computed for each pair as the average of the five ratings assigned by the annotators. However, unlike the SICK dataset, the organisers has a clear explanations 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 to predict a value between 0-5 which reflects the similarity of the given sentence pair.

Similar to the SICK dataset, we calculate some statistics and produce some graphs. 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

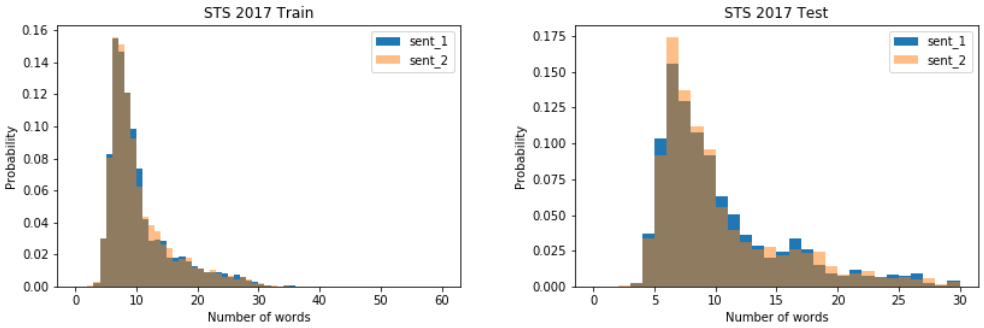
[8] Amazon Mechanical Turk is a crowdsourcing website for businesses to hire remotely located *crowd workers* to perform discrete on-demand tasks. It is available at https://www.mturk.com/



(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.



(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.

Year	Dataset	Pairs	Source
2012 [23]	MSRpar	1500	newswire
	MSRvid	1500	videos
	OnWN	750	glosses
	SMTnews	750	WMT eval.
	SMTeuroparl	750	WMT eval.
2013 [24]	HDL	750	newswire
	FNWN	189	glosses
	OnWN	561	glosses
	SMT	750	MT eval.
2014 [25]	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 [26]	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 [27]	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 [28]	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.

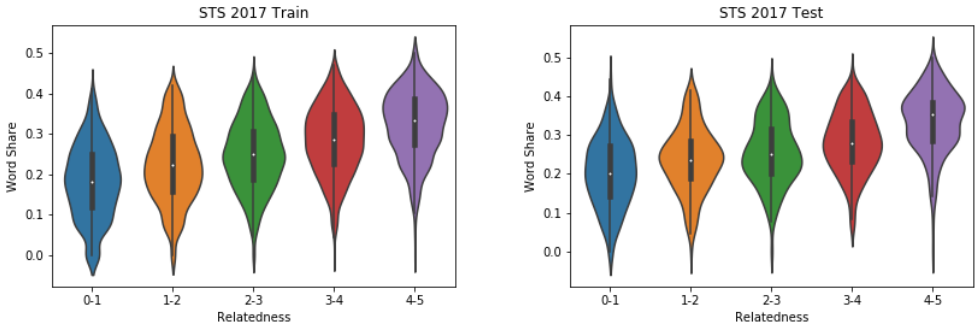
to Table 1.5 while both SICK datasets' and STS 2017 test set's maximum word count is limited to 30. We believe that the reason is STS train is composed with many sources including news articles which can have lengthy sentences. However, the STS algorithm should be able to properly handle

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.

this imbalance nature between STS2017 train and test set.

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 contradicts this which is similar to the observation we had with SICK dataset.



(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

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

Since the statics of SICK and STS 2017 datasets are similar one dataset can be used to augment the training data in the other dataset which can lead to better results as neural networks perform better with more data [36, 37]. We hope to experiment this with supervised machine learning models in Chapters 4 and 5.

3. **Quora Question Pairs** [9] The Quora Question Pairs dataset is a big dataset which was first released for a Kaggle Competition[10]. Quora is a question-and-answer website where questions are asked, answered, followed, and edited by internet users, either factually or in the form of opinions. If a particular 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 have been pairs of *related question* which, although pertaining to similar topics, 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 test set resulting 320,000 questions

[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

pairs in the training set and 80,000 sentence pairs in the testing set. The machine learning models need to predict a value between 0 and 1 that reflects whether it is a duplicate question pair or not. 1 indicates that a certain question pair is a duplicate and 0 indicates it is not a duplicate.

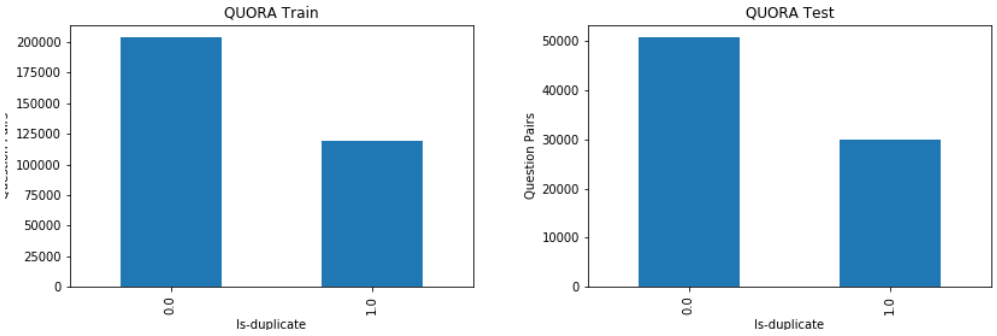
Question Pair	**is-duplicate**
1. What are natural numbers? 2. What is a least natural number?	0
1. Which Pizzas are most popularly ordered in Dominos menu? 2. How many calories does a Dominos Pizza have?	0
1. How do you start a bakery? 2. How can one start a bakery business?	1
1. Should I learn Python 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.

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

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

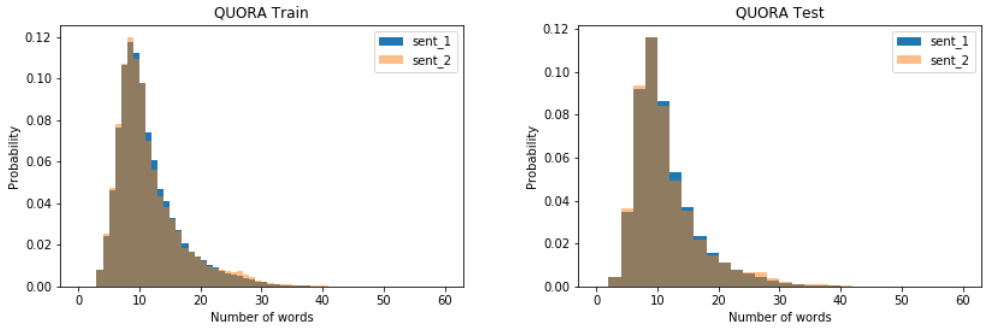
Table 1.7: Word count stats in QUORA training and QUORA testing.*STD* indicates the standard deviation and the other acronyms indicate the common meaning



(a) Is-duplicate distribution of QUORA training test

(b) Is-duplicate distribution of QUORA testing set

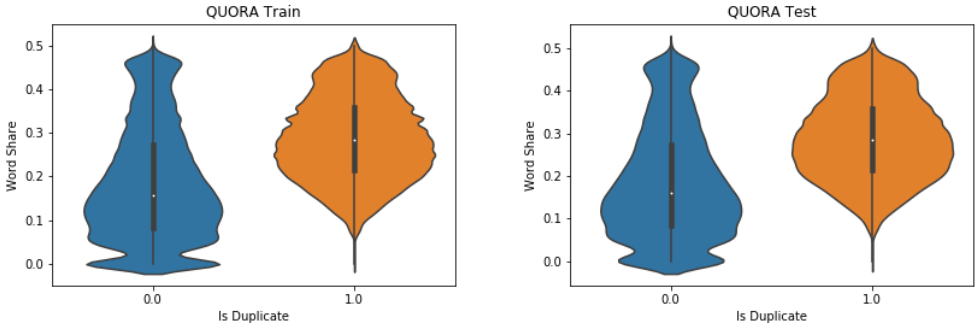
Figure 1.7: Is-duplicate distribution of QUORA train and QUORA test. *Sentence Pairs* shows the number of sentence pairs that a certain *Is-duplicate* has.



(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.



(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*

In Figure 1.7 we show the distribution of the two classes in QUORA dataset. The dataset seems to have more non duplicate question pairs than duplicate sentence pairs which is 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 contains longer texts than SICK and STS 2017 datasets. Therefore, QUORA dataset should be able to test 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. The machine learning algorithm should be able to handle them properly.

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 same intent but in different words [38]. 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 it easier to find high quality answers to questions 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 a STS method without depending on the language. Therefore through out our research 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 [28]. Unlike Spanish, 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 the Table 1.8.

   To prepare the annotated instances, a subset of the English STS 2017 dataset has been selected and human translated into Arabic. Sentences have been

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

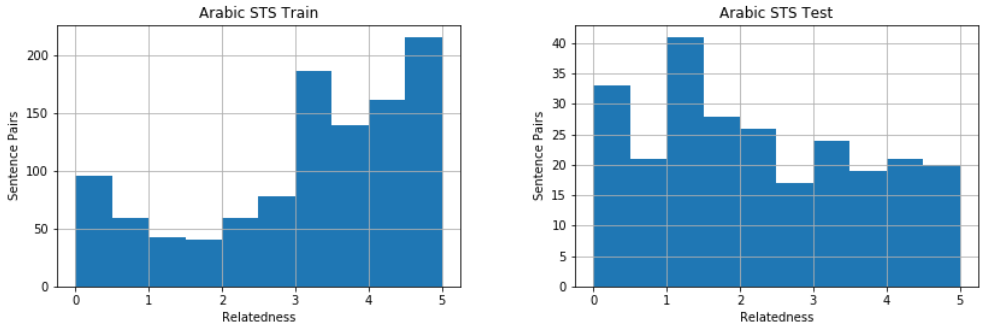
Dataset	Pairs	Source
Trial	23	Mixed STS 2016
MSRpar	510	newswire
MSRvid	368	videos
SMTeuroparl	203	WMT eval.

Table 1.8: 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.

translated independently from their pairs. Arabic translation has been provided by native Arabic speakers with strong English skills in Carnegie Mellon University in Qatar. Translators have been given an English sentence and its Arabic machine translation5 where they have performed post-editing to correct errors. STS labels have been then transferred to the translated pairs. Therefore, annotation guidelines and the template will be 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 1.9 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 a given Arabic sentence pair.

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 [28]. The training set has 1250 sentence pairs annotated with a relatedness score between 0

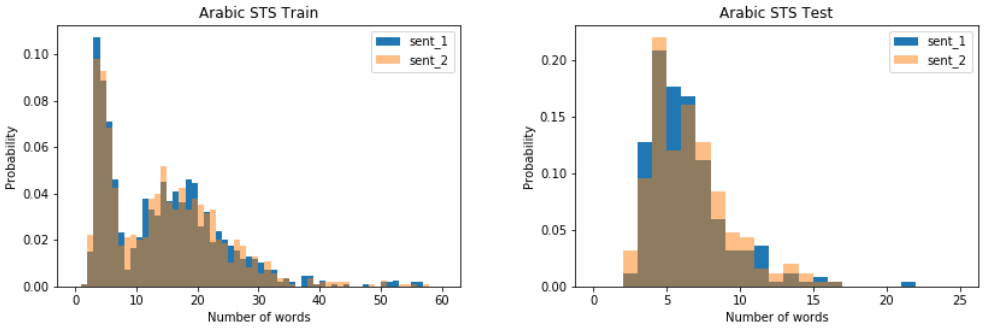
[12]The Spanish STS dataset can be downloaded at http://alt.qcri.org/semeval2017/task1/index.php?id=data-and-tools



(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.



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

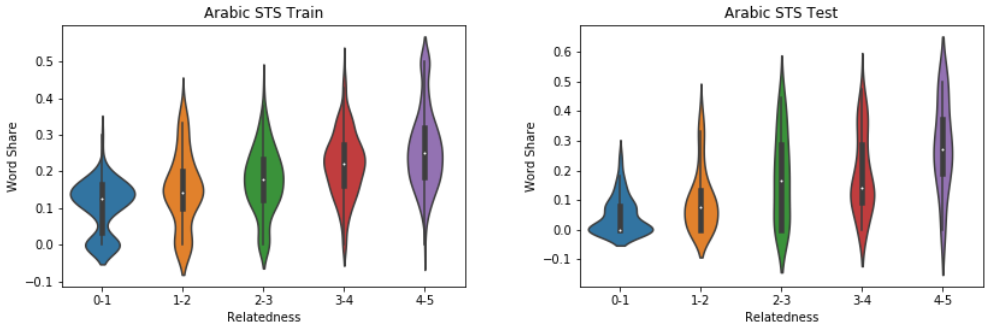
(b) Normalised distribution of word count in Spanish STS test

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.

Sentence Pair	Similarity
1. .■■■■ ■■■■ ■■■■■ *Someone is frying meat.* 2. .■■■■■■■ ■■■■ ■■■■■ *Someone plays the piano.*	0.250
1. .■■■■■■ ■■ ■■■■■■■■ ■■■■ ■■■■■ *A woman cleaning ingredients in the bowl.* 2. .■■■■■■ ■■ ■■■■■ ■■■■■ ■■■■ ■■■■■ *A woman breaks three eggs in a bowl.*	1.750
1. .■■■■■■■■ ■■■■ ■■■■ *A Child is playing harp.* 2. . ■■■■■■■■ ■■■■ ■■■ *A man plays the harp.*	2.250
1. .■■■■■■ ■■■■■ ■■■■ ■■■■■■ *The woman chops green onions.* 2. .■■■■ ■■■■ ■■■■■ *A woman peeling an onion.*	3.250
1. .■■■■■■ ■■■ ■■■ ■■■■■ *The deer jumped over the fence.* 2. .■■■■■■■ ■■■■ ■■■ ■■■■ ■■■ *Deer Jumps Over Hurricane Fence*	4.800

Table 1.9: 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.

and 4. The training set combined several datasets from previous SemEval STS shared tasks also[28]. Table 1.11 shows more information about the training set. There were two sources for test set - Spanish news and Spanish Wikipedia dump having 500 and 250 sentence pairs respectively [28]. Both datasets were annotated with a relatedness score between 0 and 5. Table 1.12 shows few pairs of sentences with their similarity score. The



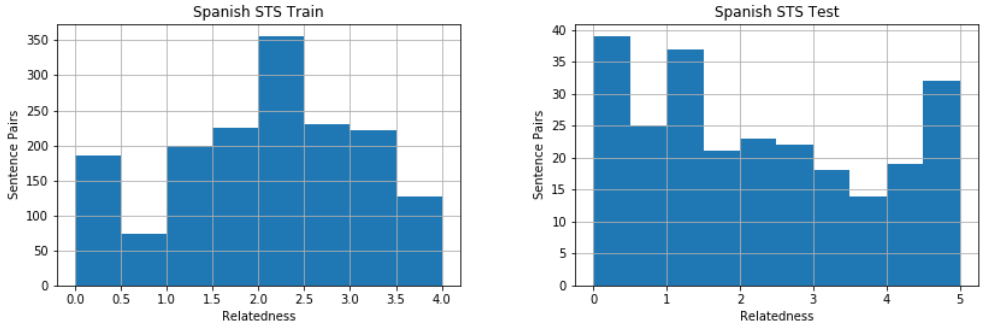
(a) Word share against relatedness bins in Arabic STS train

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



(a) Relatedness distribution of Spanish STS training test

(b) Relatedness distribution of Spanish STS testing set

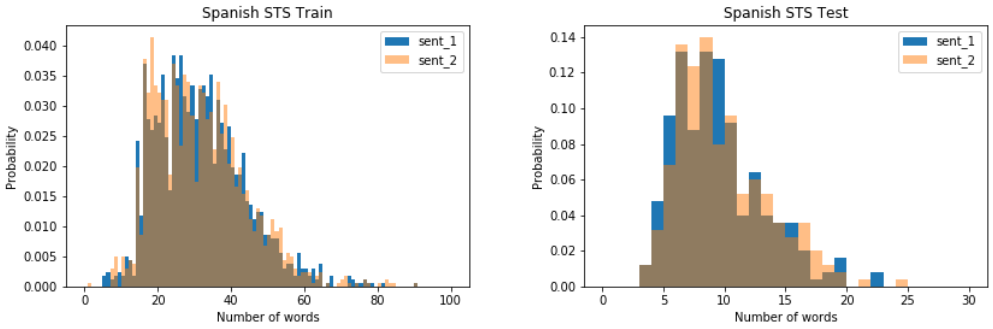
Figure 1.13: Relatedness distribution of Spanish STS train and Spanish STS test. *Sentence Pairs* shows the number of sentence pairs that a certain *Relatedness bin* has.

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 [25]	Trial	56	NR
	Wiki	324	Spanish Wikipedia
	News	480	Newswire
2015 [25]	Wiki	251	Spanish Wikipedia
	News	500	Sewswire

Table 1.11: 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.

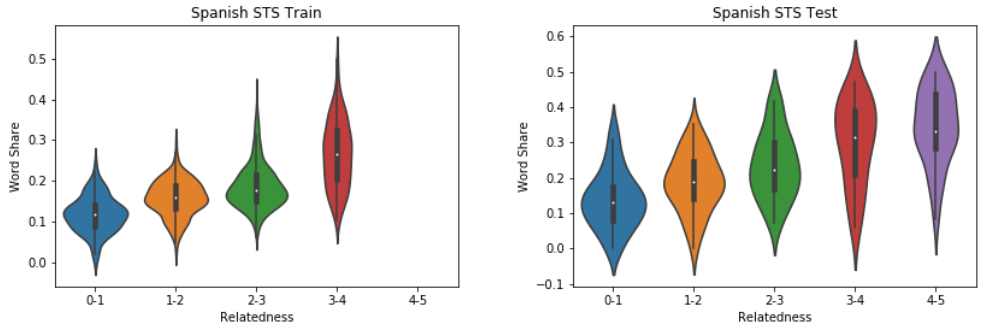
Similar to the English datasets we calculate some statistics and produce some graphs. A key challenge in the Spanish STS dataset is that test set is very different from the training set. As can be seen in Figure 1.13 training set has been annotated with relatedness scores 0-4 while the test set



(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.



(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

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 1.12: 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.

has been annotated with relatedness scores 0-5. Therefore, STS methods should be able to handle that properly. Furthermore, as shown in Figure 1.14 and in Table 1.13 sentence pairs in test set are shorter in word length

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

than the sentence pairs in train set. Therefore, STS methods working on this dataset should be able to properly handle that too. 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 strength of a STS system as well.

1.1.3 Datasets on Different Domains

In order to experiment how our STS methods can be adopted in to 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. [39]. The dataset comprises 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 do-

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

main [14]. The sentence pairs have been evaluated by five different human experts that judged their 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 [23]. Besides the annotation instructions, example sentences from the bio-medical literature have been also provided to the annotators for each of the similarity degrees. 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 bio medical sentence pair.

A dataset as small as this one can not be used by to train a supervised ML method, requiring alternative approaches such as unsupervised methods and transfer learning techniques which we will be exploring in the next few chapters.

1.2 Evaluation Metrics

While training a model is a key 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

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

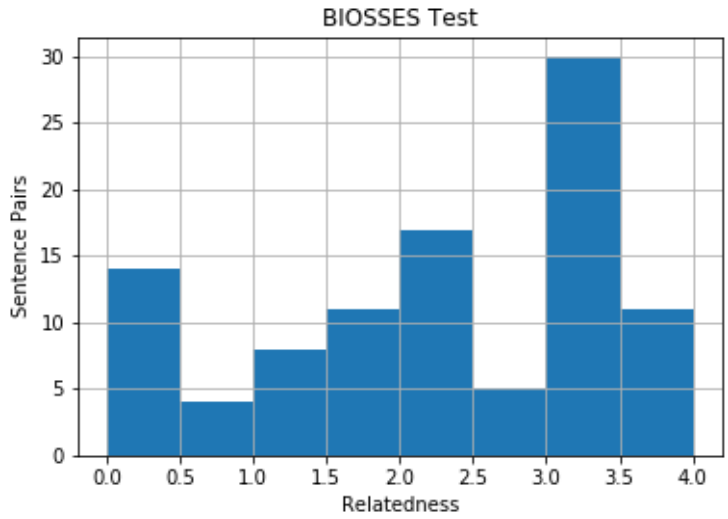


Figure 1.16: Relatedness distribution of BIOSSES. *Sentence Pairs* shows the number of sentence pairs that a certain *Relatedness bin* has.

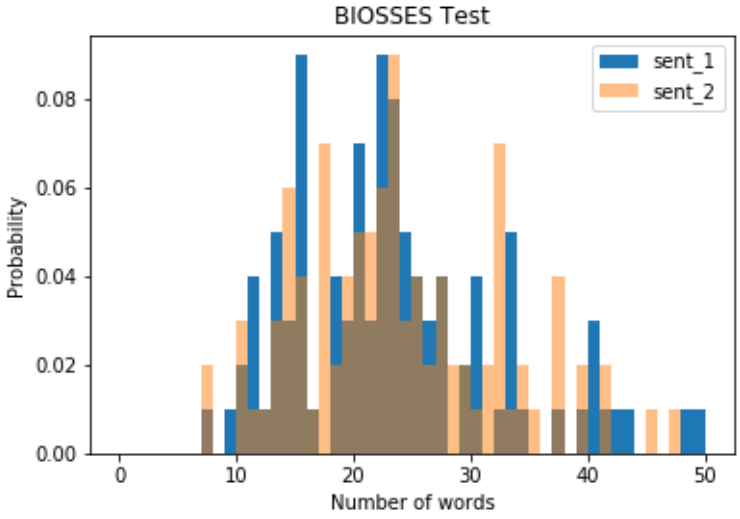


Figure 1.17: Normalised distribution of word count in BIOSSES. *Number of words* indicates the word count and *Probability* shows the total probability of a sentence with that word count appearing in the dataset.

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.

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 true 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 through out the first part of our research.

In the equations presented for each of the evaluation metrics, we represent

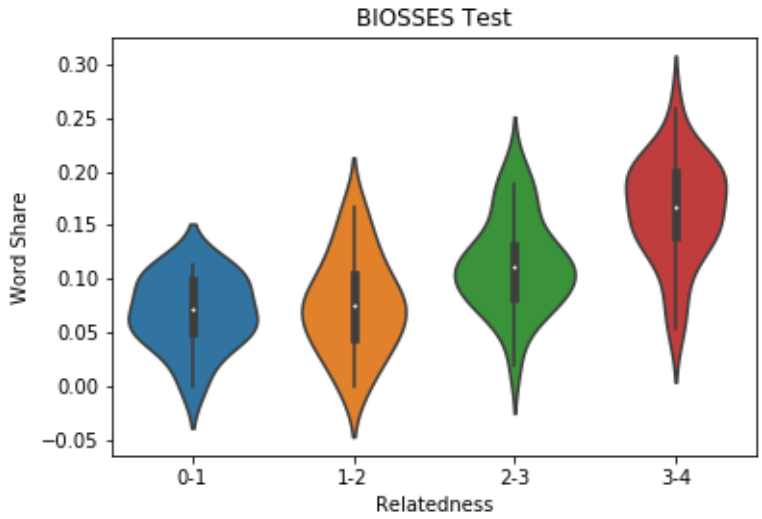


Figure 1.18: Word share against relatedness bins in BIOSSES. *Word Share* indicates the ratio between number of common words in the two sentences to total number of words in the two sentences against each *Relatedness* bins

the gold labels with X and predictions with Y. Therefore, a gold label in i^{th} position will be represented by X_i and a 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) is a measure of the strength of the linear association between the two variables. A value of +1 is total positive linear correlation between the variables, 0 is no linear correlation, and -1 is total negative linear correlation.

   Pearson's Correlation Coefficient is one of the most common evaluation metrics in STS shared tasks [23, 24, 25, 26, 27, 29]. 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 and 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 [23, 24, 25, 26, 27, 29]. 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.

   But not exactly at 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 Correlation Coefficient only works with a linear relationship between the two variables whereas the Correlation Coefficient works

with the monotonic relationships as well. Spearman's Correlation Coefficient equation 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.

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

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 continues. Therefore, in 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.

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

1.3 Contributions

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

1. In each chapter, we cover various supervised and unsupervised techniques to compute semantic similarity that can benefit a wide range of NLP application. 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 which 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 the 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 the each chapter.

1.4 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 any more. With the advances of word em-

beddings, and as a result of 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 in to 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 a timely contribution to the NLP field.

The introduction of competitive STS shared tasks led to the development of standard datasets. From the publicly available datasets we selected three recently released English STS datasets; SICK, STS2017 and Quora Question Pairs. They carry different characteristics. We exploratory analysed these dataset focussing on common properties like size of the dataset, sentence length, common number of words etc. Furthermore, we identified certain 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 selected a Bio-medical STS Dataset. This dataset bring 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 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 and in Quora dataset, Root Mean Squared Error has been used to evaluate the methods. We followed the same evaluation measures in order to compare our methods with other systems submitted 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

Improving State of the Art STS Methods

The biggest challenge that the neural based architectures face when applied to STS tasks is the small size of datasets available to train them. As a result, in many cases the networks cannot be trained properly. Given the amount of human labour required to produce datasets for STS, it is not possible to have high quality large training datasets. As a result researches working in the field have also considered unsupervised methods for STS. Recent unsupervised approaches use pretrained word/sentence embeddings directly for the similarity task without training a neural network model on them. Such approaches have used cosine similarity on sent2vec [40], InferSent [41], Word Mover's Distance [16], Doc2Vec [42] and Smooth Inverse Frequency with GloVe vectors [17]. While these approaches have produced decent results in the final rankings of shared tasks, they have also provided strong baselines for the STS task.

This chapter explores the performance of three unsupervised STS methods - cosine similarity using average vectors, Word Mover's Distance [16] and cosine similarity using Smooth Inverse Frequency [17] and how to improve them 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 perform compared to other popular supervised/ unsupervised STS methods?

RQ3: Can the proposed unsupervised STS method be easily adopted in to different languages?

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

The main contributions of this chapter are as follows.

1. In the Related Work Section (Section 2.1), we cover three unsupervised STS techniques to compute semantic similarity at the sentence level.

2. We evaluate sentence en 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 with the experiments conducted are publicly available to the community[1].

4. We published the findings in this chapter in Ranasinghe et al. [43].

The rest of this chapter is organised as follows. Section 2.1 describes the three unsupervised STS methods we experimented in this section. In section 2.2 we present the methodology, the contextual word embeddings we used followed

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

by the results to the English datasets comparing with the baselines. Section 2.3 and Section 2.4 shows how our method can be applied to different languages and domains and 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 in neural networks, most 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 which employ neural representation models: supervised and unsupervised. Unsupervised approaches use pretrained word/sentence embeddings directly for the similarity task without training a neural network model on them while supervised approaches uses a machine learning model trained to predict the similarity using word embeddings. Since this chapter focuses on unsupervised STS methods, this section would contain the previous research done on unsupervised STS methods.

The three unsupervised STS methods explored in this chapter: Cosine similarity on average vectors, Word Mover's Distance and Cosine similarity using Smooth Inverse Frequency are the most common unsupervised methods explored in STS tasks. Apart from them, cosine similarity of the output from Infersent [41], sent2vec [40] and doc2vec [42] have 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. This is a common way to acquire sentence embeddings from word embeddings. Obviously, this simple baseline leaves considerable room for variation. Researches have investigated the effects of ignoring stopwords and computing an average weighted by tf-idf in particular.

2.1.2 Word Mover's Distance

The second STS state of the arts method that we have considered is Word Mover's Distance introduced by Kusner et al. [16]. 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. [16] shows that this is a good approach than vector averaging since this technique keeps the word vectors as it is through out the operation.

2.1.3 Cosine Similarity Using Smooth Inverse Frequency

The third and the last unsupervised STS method we have considered is to acquire sentence embeddings using Smooth Inverse Frequency proposed by Arora et al. [17] and then calculate the cosine similarity between those sentence embeddings.

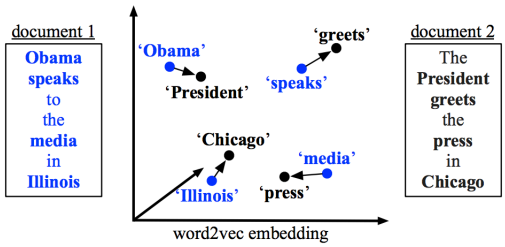


Figure 2.1: The Word Mover's Distance between two sentences

Semantically speaking, taking the average of the word embeddings in a sentence tends to give too much weight to words that are quite 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 that, Smooth Inverse Frequency computes the principal component of the resulting embeddings for a set of sentences. It then subtracts their projections on first principal component from these sentence embeddings. This should remove variation related to frequency and syntax that is less relevant semantically.

As a result, Smooth Inverse Frequency downgrades unimportant words such as *but, just*, etc., and keeps the information that contributes most to the semantics of the sentence. After acquiring the sentence embeddings for a pair of sentences, the cosine similarity between those two vectors were taken to represent the sim-

ilarity between them.

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 "play" which has several meanings, but in standard word embeddings such as GloVe [44], fastText [45] or Word2Vec [46] 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 solved this problem by providing vectors for words considering their context too. In this way the word 'bank' in above sentences have two different embeddings. Contextual word embedding models have improved the results of many natural language processing tasks over traditional word embedding models [20, 47]. However, to the best of our knowledge they have not been applied on unsupervised STS methods.

Therefore, we explore how the 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 just feed the words to get the embeddings. We considered these contextualised word representations mainly considering the popularity they had by the time we were doing the experiments.

1. **ELMo**[2] introduced by Peters et al. [47] use bidirectional language model (biLM) to learn both 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 like text classification [48], named entity recognition [49] which motivated us to explore ELMo in unsupervised STS methods. Also, we were aware about the fact that ELMo has been pre-trained on different languages [50] and different domains [51] which will be easier when we are adopting our methodology for different languages and domains in Sections 2.3 and 2.4.

2. **BERT**[3] introduced by Devlin et al. [20] might probably be the most popular contextualised word embedding model. In contrast to ELMo which uses a shallow concatenation layer [20], 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 like text classification [52], word similarity [53], named entity recognition [54], question and answering [55] etc. Similar to ELMo, BERT too has been widely adopted to different languages[4] such as Arabic [56], French [57], Spanish [58], Greek [59] etc. and different domains such as SciBERT [60], BioBERT [61], LEGAL-BERT [62] etc.

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

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

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

3. **Flair**[5] is another type of popular contextualised word embeddings introduced in Akbik et al. [63]. It takes a different approach by using a character level language model rather than the word level language model used in ELMo and BERT. Flair also has been used successfully in natural language processing tasks such as named entity recognition [64], part-of-speech tagging [63] and has been widely adopted in to different languages and domains [63, 65].

Apart from using these contextual word embedding models individually we also considered **Stacked Embeddings** of these models together. Stacked Embeddings are obtained by concatenating different embeddings. According to Akbik et al. [63] 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 has been applied on unsupervised 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 we explained in Section 1.1. Our implementation was based

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

on *Flair-NLP* Framework [66] which makes it easier to switch between different word embedding models when acquiring word embeddings. Also *Flair-NLP* has their 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 like *small, large etc.*. Usually the larger models provide a better accuracy since they have been trained on a bigger dataset compared to the smaller models. However, this comes with the disadvantage that these larger models are resource-intensive than the smaller models. In order to achieve a better accuracy, we used the largest model available in each contextual word embedding models. We will describe them in the following paragraphs.

For ELMo we used the 'original (5.5B)' pre-trained model provided in Peters et al. [47] 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. [47] mentions that ELMo original (5.5B) has slightly higher performance than other ELMo models and recommend it as a default model. Using this model we represented each word as a vector with a size of 3072 values.

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. [20]. We represented each word as a 4096 lengthened vector using this model.

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

As suggested in Akbik et al. [63] 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 [67] word embeddings. We used the stacked model to represent each word as a 4196 lengthened vector.

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

In order to compare the results of contextualised word embeddings, we used a standard word representation model in each experiment as a baseline. In this research we used Word2vec embeddings [15] 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 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 which we explained on Section 2.1. Table 2.1 shows the results for SICK dataset, Table 2.2 shows the results for STS 2017 dataset and Table 2.3 shows the results for Quora Question Pairs dataset. In order to compare our results with the other systems, we conducted the experiments only on the test data of the three mentioned datasets. Since this method leaves considerable room for variation, we have investigated

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

the following variations and reported their results in each table.

(a) All the word vectors were considered for averaging. Results are 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 its 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 remaining word vectors were weighted from its 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. I, II, III and IV indicates the different variations as explained before. 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. Best result in each variation is marked in **Bold**. Best result from all the variations is marked with †.

From the results in Table 2.1, 2.2 and 2.3 there is no clear indication that

	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. I, II, III and IV indicates the different variations as explained before. 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. Best result in each variation is marked in **Bold**. Best result from all 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. I, II, III and IV indicates the different variations as explained before. For each word embedding model, Root Mean Squared Error (RMSE) is reported on all variations. $\bigoplus$ indicates a stacked word embedding model. Best result in each variation is marked in **Bold**. Best result from all the variations is marked with †.

contextualised word embeddings perform better than the standard word embeddings. In all the datasets considered, the best result was provided 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 state of the art STS methods.

2. **Word Mover's Distance** - As the second unsupervised STS method we experimented with Word Mover's Distance explained on Section 2.1. Similar to the average vectors, we compared having contextualised word embeddings in the place of traditional word embeddings in Word Mover's Distance. Table 2.4 shows the results for SICK dataset. Table 2.5 shows the results for STS 2017 dataset and Table 2.6 shows the results for Quora Questions Pairs dataset. We have investigated the effects of considering/ ignoring stop words before calculating the word mover's distance which are 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 table contextualised word representations could not improve Word Mover's method too over standard word representations. Even though, ELMo $\bigoplus$ BERT model outperforms Word2vec in SICK and STS 2017 dataset with regard to Spearman Correlation ($\boldsymbol{\tau}$) there is no clear indication that contextual word representations would outperform standard

	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. I and II indicates the different variations as explained before. 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. Best result in each variation is marked in **Bold**. Best result from all 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. I and II indicate the different variations as explained before. 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. Best result in each variation is marked in **Bold**. Best result from all the variations is marked with †.

word representations in Word Mover's method. Since the travelling distance is dependent on number of dimensions, the 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 the final unsupervised STS method we experimented with Smooth Inverse Frequency explained on

	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. I and II indicate the different variations as explained before. For each word embedding model, Root Mean Squared Error (RMSE) is reported on all variations. $\bigoplus$ indicates a stacked word embedding model. Best result in each variation is marked in **Bold**. Best result from all the variations is marked with †.

Section 2.1. Similar to the previous STS methods, we compared having contextualised word embeddings in the place of traditional word embeddings in 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 SICK dataset. Table 2.8 shows the results for STS 2017 dataset and Table 2.9 shows the results for 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. Best result from all the variations is marked with †.

As can be seen in the results, unlike the previous unsupervised STS meth-

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. Best result from all 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. Best result is marked with †.

ods, contextualised word embeddings improved the results in Smooth Inverse Frequency method compared to the standard word embeddings in all the three datasets considered. It can be observed that 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 and due to this reason, contextualised word embedding models does not suffer the *Curse of dimensionality* [68] with Smooth Inverse Frequency. In all of the datasets, the stacked embedding model of ELMo and BERT (ELMo $\bigoplus$ BERT) performed best. Further evaluating, from all the unsupervised STS methods we experimented including Vector Averaging and Word Movers

Distance too, 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 Movers Distance, there was clear improvement when they were applied on Smooth Inverse Frequency.

With regard to our *RQ2: How well the proposed unsupervised STS method performs when compared to various other STS methods?*, we compared our best results of the SICK dataset to the results in the SemEval 2014 Task 1 [29] which was the original task that the SICK dataset was initiated as we mentioned before. Our unsupervised method had 0.753 Pearson correlation score, whilst the best result in the competition had 0.828 Pearson correlation [29]. Our approach would be ranked on the ninth position from the top results out of 18 participants, and it is the best unsupervised STS method among the results [29]. 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 in to a Support Vector Machine and obtained a 0.714 Pearson correlation [32]. Compared to these complex approaches our simple unsupervised approach provides a strong baseline to STS tasks. This answers our *RQ2*, that the proposed unsupervised STS method is competitive with the other supervised and unsuper-

vised STS methods.

2.3 Portability to Other Languages

Our **RQ3** targets the multilinguality 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 . Our approach has the advantage that it does not rely on language dependent features and it does not need 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. [50]. Che et al. [50] have trained ELMo models for 44 languages including Arabic and Spanish using the same hyperparameter settings as Peters et al. [47] on Common Crawl and a Wikipedia dump of each language[8]. The models are hosted in NLPL Vectors Repository [69][9]. As for BERT we used the "BERT-Base, Multilingual Cased" model [20] which has been built on the top 100 languages with the largest Wikipedias that includes Arabic and Spanish languages too. Similar to the English experiments, we conducted the experiments through the *Flair-NLP* Framework [66]. In order to compare the results, as traditional word embeddings, we

[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

used AraVec [70] [10] for Arabic and Spanish 3B words Word2Vec Embeddings [71][11] for Spanish.

Similar 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 [72] in the competition had 0.754 Pearson correlation [28]. Our approach would rank eighteenth out of 49 teams in the final results. Similar to English, our approach has the best result for an unsupervised method and surpasses other complex supervised models. For example Kohail et al. [73] proposes a supervised approach, which combines dependency graph similarity and coverage features with lexical similarity measures using regression methods and scored only 0.610 Pearson correlation. This shows that the proposed unsupervised STS method outperforms this supervised STS method.

For Spanish, our approach had 0.712 Pearson correlation whilst the best result [74] in the competition had 0.855 Pearson correlation [28]. 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 Barrow and Peskov [75] uses a supervised machine learning algorithm with word embeddings and scored only

[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

0.516 Pearson correlation. Our fairly simple unsupervised approach outperform 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

In order 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 explained in 1. As we mentioned before Bio-medical STS dataset does not have a training set. Therefore, only the unsupervised approaches can be applied on this dataset which provides an ideal opportunity for the STS method we introduced in this Chapter.

For the experiments, as the contextual word embedding models, we used BioELMo [51][12], BioBERT [61][13] and BioFLAIR [65][14]. Additionally, to compare the performance with standard word embeddings, we used BioWordVec [76][15]. Same as English and multilingual experiments, Smooth Inverse Frequency with ELMo and BERT stacked embeddings performed best with this dataset too. It had 0.708 Pearson correlation, whilst the best performing method had 0.836 Pearson

[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 availble on https://bio.nlplab.org/

correlation. This would rank our approach seventh out of 22 teams in the final results of the task [39].

It should be also noted that it outperforms many complex methods that sometimes uses external tools too. As an example, the UBSM-Path approach is based ontology based similarity which uses METAMAP [77] for extracting medical concepts from text and our simple unsupervised approach outperform them by a significant margin. UBSM-Path only has 0.651 Pearson correlation and compared to that our simple STS method based on contextual embeddings outperform them.

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

2.5 Conclusions

This chapter experimented three unsupervised STS methods namely cosine similarity using average vectors, Word Mover's Distance and cosine similarity using Smooth Inverse Frequency with contextualised word embeddings for calculating 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 Smooth Inverse Frequency method were improved significantly with contextualised word embeddings, instead of standard word embeddings. Further more we learned that stacking ELMo and BERT provides a strong word representation rather than 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, our approach finished on the top half of the final results list in the SICK dataset surpassing many complex and supervised approaches.

Our approach was also applied in the Arabic, Spanish and Bio-medical STS tasks, where our simple unsupervised method finished on the top half of the final result list in all the cases 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 unsupervised method for STS tasks.

Contextual word embedding models are getting popular day by day due to their superior performance compared to standard word embedding models. Contextual word embedding models are available even in low resource languages like Assamese [78], Hebrew [79], Odia [78], Yoruba [80], Twi [80] etc. Very soon, contextual word embedding models would be available in all the languages where standard word embedding models 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 [21], RoBERTa [22], SpanBERT [81] etc. One drawback of using Contextual word embedding models is that

most of the pretrained models only support 512 maximum number of tokens which would be problematic when encoding longer sequences. Therefore, STS with long sequences can be explored with recently released contextual word embedding models like Longformer [82] and Big Bird [83] that supports encoding longer sequences than 512 maximum number of tokens. Taking advantage from the fact that this method is unsupervised and does not need a training dataset, it can further expanded in to many languages and domains as future work.

Chapter 3

Exploring STS with 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 would take a sentence or text as the input and would output 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 down stream tasks, the sentence encoder is often considered as a black box where the users use the sentence encoder to get sentence embeddings without knowing what exactly happens in the encoder itself.

Ideally, the approaches we experimented in Chapter 2 like Vector Averaging, Smooth Inverse Frequency [17] can also be considered as sentence encoders since in those approaches the input is a variable-length text and the output is a fixed-length vector. However, these approaches have major drawbacks in representing sentences. One such drawback is these approaches do not care about the word order. If you consider two sentences *"Food is good but the service is bad"* and *"Food is bad but the service is good"*, would have the same embeddings from these approaches even though the meaning of these two sentences is completely different. Another drawback is those approaches lose information in the

vector aggregation process. If you consider two sentences *"It is great"* and *"It is not great"*, Vector Averaging and Smooth Inverse Frequency would give similar sentence embeddings as there is only one word difference in the two sentences. Even though this affect can be minimised using techniques like TF/IDF weighting that we explored in Chapter 2, a different approach would be to train end-to-end models to get sentence embeddings. These models are commonly addressed as *sentence encoders* in NLP community.

Over the years, various sentence encodes like Sent2vec [40], Infersent [84], Universal Sentence Encoder [41] have been proposed. Even though most of these sentence encoders have sophisticated architectures, since many are using them only as a black box to get the sentence embeddings, they have been very popular in the NLP community. As the sentence encoders provide good quality sentence embeddings efficiently, the word embedding aggregation methods like Vector Averaging, Smooth Inverse Frequency have been often overlooked by the community in favour of sentence encoders.

Once you have the sentence embedding from a sentence encoder, using them to calculate STS is an easy task. Since these sentence embeddings are already semantically powerful, a simple vector comparison technique like cosine similarity between the embeddings can be used to calculate the STS of the two sentences. Therefore, pre-trained sentence encoders can be used as an unsupervised STS method. However, even though the sentence encoders have been commonly used in STS tasks, there is no comprehensive study done on them. Since most of the researchers are using sentence encoders as a black box in many applica-

tions, they don't understand the limitations of using them. In this chapter we are addressing this gap by exploring sentence encoders in different STS datasets adopting them in different languages and domains. With a study like this we can understand the limitations of the sentence encoders and when not to use them.

We address three research questions in this chapter:

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

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

RQ3: How well the 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 with the experiments conducted are publicly available to the community[1].

The rest of this chapter is organised as follows. Section 3.1 describes the three sentence encoders we experimented 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 results comparing with the other unsupervised STS

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

methods. Section 3.3 and Section 3.4 shows how sentence encoders can be applied to different languages and domains and their results. The chapter finishes with conclusions and ideas for future research directions in sentence encoders.

3.1 Related Work

As we mentioned before, the sentence encoders are popular with the NLP community. The three sentence encoders explored in this chapter: Sent2vec [40], Infersent [41] and Universal Sentence Encoder [84] are the most common sentence encoders. Other than them there is also Doc2vec [85] which is can be considered as a sentence encoder. However, due to the various upgrades in different python libraries, the official pre-trained Doc2vec models are not working any more. Therefore, we only used following sentence encoders in our experiments.

Sent2vec Sent2vec presents a simple but efficient unsupervised objective to train distributed representations of sentences [40]. It can be thought 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, as well as 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 [40].

Sent2vec has officially released several pre-trained models to derive the sen-

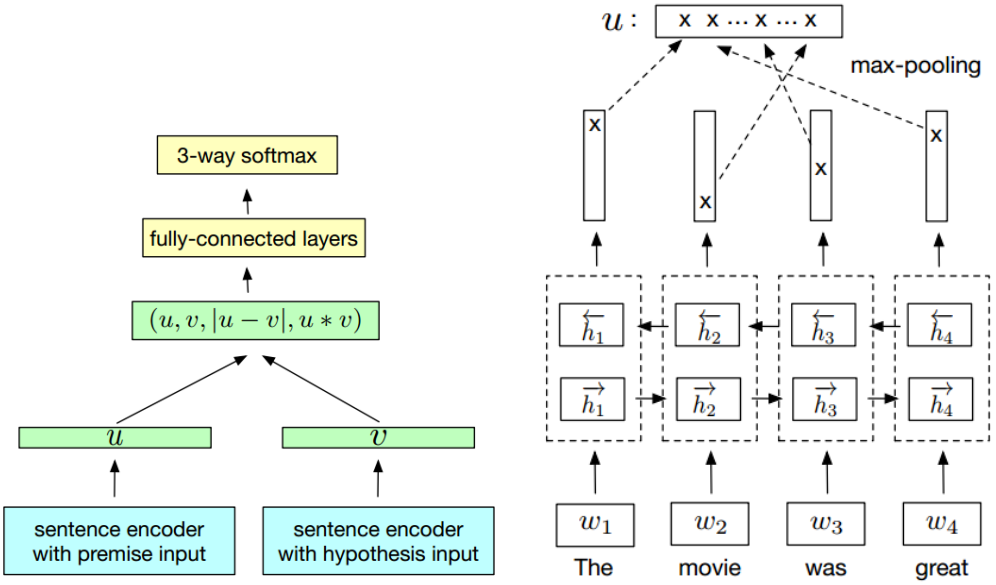
tence embeddings[2]. Also due to its unsupervised approach and simple objective function, Sent2vec has been adopted in different languages and domains too.

Infersent InferSent is an NLP technique for universal sentence representation developed by Facebook which uses supervised training to produce high quality sentence vectors [41]. The authors explore 7 different architectures for sentence encoding including LSTM [86], GRU [87], Bi directional LSTM [88] with mean/max pooling, Self-attentive network and Hierarchical Deep Convolutional Neural Network [41]. 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 them as features in 12 different transfer tasks like Binary and multi-class text classification, semantic textual similarity, paraphrase detection etc. The results indicate that the BiLSTM with the max-pooling operation performs best on these tasks [41]. The architecture of this 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 [67] which in turn has been trained on text preprocessed with the PTB tokeniser and the other model is trained with fastText [45] which has been trained on text preprocessed with the MOSES tokeniser. We used both models in our experiments[3].

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

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



(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 [41].

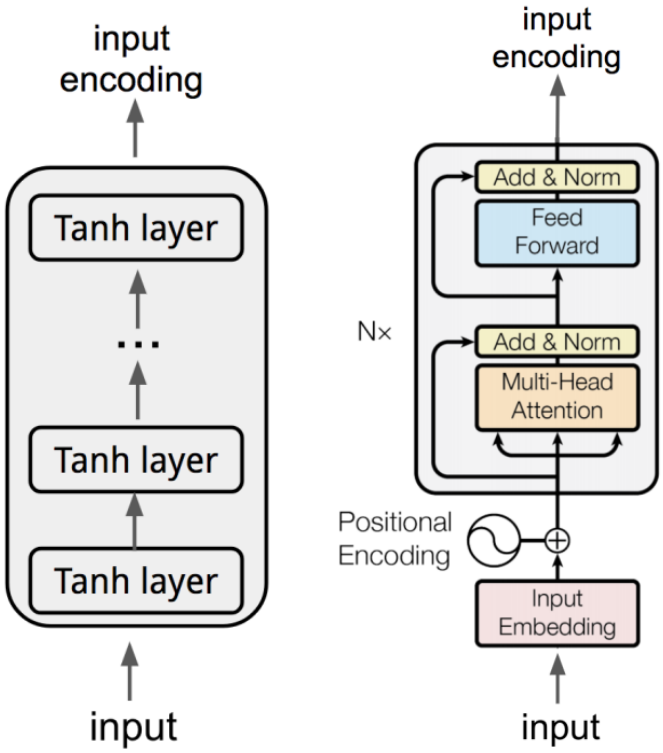
Universal Sentence Encoder The Universal Sentence Encoder [84] released by Google is the last sentence encoder we employed in this chapter. This is again an unsupervised sentence encoder. It comes with two versions i.e. one trained with Transformer encoder and other trained with Deep Averaging Network (DAN). Both architectures are outlined briefly below. The two have a trade-off of accuracy and computational resource requirement. While the one with 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 constitutes an encoder and decoder. Since our research is only focussed on encoding sentences to vectors, we only

use its encoder part. The encoder is composed of a stack of N = 6 identical layers. 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 feed-forward network. Cer et al. [84] also 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, that is what the positional encodings does. 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 advantage of the DAN encoder is that computation time is linear in the length of the input sequence. With this sentence encoder too, 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.



(a) DAN Architecture (b) Transformer Architecture

Figure 3.2: Two architectures in Universal Sentence Encoders.

3.2 Exploring Sentence Encoders in English STS

Adopting sentence encoders for STS is an easy task. If two embeddings from the two sentences are closer, the sentences are said to be semantically similar. As the approach, first the two sentences are passed to the sentence encoders to get the embeddings and the we calculate the cosine similarity between the resulting embeddings which represents the textual similarity of the two input sentences. To be clear, if the two vectors for two sentences X and Y are a and b correspondingly, 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 we explained in Section 1.1. For the experiments we used all the sentence encoders mentioned in Section 3.1. For **Sent2vec**, we used the pre-trained Sent2vec model, *sent2vec_wiki_bigrams* trained on English Wikipedia articles. Using that we could represent a sentence from a 700 dimensional vector. For **Infersent**, as we mentioned before, there are two pre-trained models available; *infersent1* which was trained using GloVe [67] and *infersent2* which was trained using fastText [45]. Both models have been trained on the SNLI dataset which consists of 570k human generated English sentence pairs, manually labelled with one of the three categories: entailment, contradiction and neutral [35]. Using that we could represent a

sentence from a 512 dimensional vector. For **Universal Sentence Encoder**, we used universal-sentence-encoder (DAN architecture) and universal-sentence-encoder-large (Transformer architecture) which were trained on text resources like Wikipedia and news articles. With that too we could represent a sentence from a 512 dimensional vector.

We evaluated these three sentence encoders in three English STS datasets that we explained in Section 1.1; SICK, STS2017 and QUORA. Table 3.1 shows the results for SICK dataset, Table 3.2 shows the results for STS 2017 dataset and Table 3.3 shows the results for 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*. Best result from all the methods is marked with †.

As can be seen in the results, Universal Sentence Encoder outperformed all other sentence encoders in all the English STS datasets. From the two architectures available in the Universal Sentence Encoder, Transformer architecture outperforms the DAN architecture as they have explained in their paper. Furthermore, it should be noted that in all three datasets, sentence encoders outper-

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*. 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*. Best result is marked with †.

form 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 adopted and perform well in English STS tasks. However, most of these models are complex in nature which would result in more processing time/resources which can be chaotic in some situations.

3.3 Portability to Other Languages

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

If you 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 [35]. If someone is adopting **Infersent** for a different language other than English, they need to have a corpus similar to SNLI with a similar size. Annotating such a corpus for a different language would be challenging. Even though there are some attempts like XNLI [89] to create such a corpus, the number of annotated instances are very limited. This makes it difficult to adopt **Infersent** in other languages which is a clear limitation of the Infersent architecture.

However, the other two sentence encoders we experimented in 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 those sentence encoders have been trained on unsupervised textual data which will be easy to find in most of the languages. However, still they need powerful computing resources to train the models which is a challenge

when adopting 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. 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[6] including Arabic and Spanish [90]. This multilingual model is available in both architecture in Universal Sentence Encoder; DAN and Transformer[7]. Using that we could represent the Arabic and Spanish sentences with a 512 dimensional vector. As we mentioned before, for **Infersent**, we could not find any pre-trained models that supports either Arabic nor Spanish. Therefore, we did not use Infersent in our multilingual experiments. The Arabic and Spanish STS results with the mentioned sentence encoders are available in Table 3.4.

As can be seen in results, similar to the English datasets, from the sentence encoders we considered, Universal Sentence Encoder with the Transformer architecture gave the best results for both Arabic and Spanish datasets. From the two architectures available in the Universal Sentence Encoder, Transformer architecture outperforms the DAN architecture in both languages. Furthermore,

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

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

[7]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 model from Chapter 2; *ELMo* $\bigoplus$ *BERT*. Best result for each language is marked with †.

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.

With these experiments, we can answer our **RQ2: How well the sentence encoders can be adopted in different languages?**. Adopting sentence encoders in different languages is challenging since there are no available pre-trained sentence encoder models for many languages. However, in the cases where they are available, it is very easy to use them in STS tasks and they provide good results compared to other unsupervised STS methods.

3.4 Portability to Other Domains

In order to answer our *RQ3*; how well the sentence encoders can be applied in different domains, we evaluated sentence encoders that we explained in this Chapter on the Bio-medical STS dataset explained in 1. As we mentioned before, Bio-medical STS dataset does not have a training set. Therefore, only the

unsupervised approaches can be applied on this dataset which provides an ideal opportunity for the sentence encoders we experimented in this chapter.

However, we faced the same issue we faced in Arabic and Spanish STS experiments in Bio-medical STS experiments too. There are not many options when it comes to sentence encoders that were trained on Bio-medical domain [91]. For **Sent2vec**, we could find a pre-trained model in Bio-medical domain which was trained using PubMed data [92][8]. However, for other two sentence encoders, we could not find any available pre-trained sentence encoder models on Bio-medical domain. Therefore, for Universal Sentence Encoder and Infersent, we had to use pre-trained sentence encoder models trained on English texts for this experiments. The results are shown in Table 3.5.

Model	$\boldsymbol{\rho}$
Infersent2	0.294
Infersent1	0.301
USE(DAN)	o.321
USE(Transformer)	0.345
ELMo $\bigoplus$ *BERT*	0.708
BioSentVec [92]	$0.810^{\dagger}$

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. Best result is marked with †.

As can be observed in the results, sentence encoder trained on the Bio-medical domain; BioSentVec [92] outperformed other approaches. In fact, when com-

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

pared to the best results in the BIOSSES dataset BioSentVec outperforms all of them which means that BioSentVec provides 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 in Chapter 2 outperforms Universal Sentence Encoder and Infersent by a large margin. This can 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 the *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 adopted for STS in different domains, however they won't succeed unless they are pre-trained on that particular domain.

With these finding we answer our **RQ3: Is it possible to adopt sentence encoders in different domains?**. We showed that it is possible to adopt sentence encoders in a different domain. However, it is difficult since pre-trained sentence encoder models are not common in most domains and the sentence encoders pre-trained on a general domain perform poorly on a specific domain like Bio-medical.

3.5 Conclusions

In this chapter we experimented another unsupervised STS method; sentence encoders. We explored three popular sentence encoders; Sent2vec, Infersent and

Universal Sentence Encoder in three English STS datasets. The results show that sentence encoders outperforms other unsupervised STS methods. From the sentence encoders, Universal Sentence Encoder with the Transformer architecture performs best in all three datasets. Furthermore, we evaluated sentence encoders in different languages and domains. We experienced the biggest hurdle in adopting sentence encoders in different languages and domains; the pre-trained sentence encoders that support different languages and domains are not common 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 like Infersent require specific training data, which is hard 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 spend time on creating language specific and domain specific sentence encoders. However, when available they perform very well in the relevant tasks. Also we tried to use sentence encoders that were trained on a general domain in a different but specific domain like Bio-medical. This provided very poor results. Therefore, we can conclude that using sentence encoders on a different domain to their pre-trained domain won't provide good results.

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 in 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 worsened by the fact that sentence encoders perform very poorly on the domains that 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 using embedding aggregation methods like Smooth Inverse Frequency would provide better results.

As future work, the experiments can be extended in to recently released sentence encoders like LASER [93]. LASER supports 93 languages including low resource languages like 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 them in Chapters 4 and 5.

Chapter 4

Siamese Neural Networks for STS

A *Siamese Neural Network* is a class of neural network architectures that contain two or more identical subnetworks which is usually employed in applications that requires 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 at the same time. The subnetworks have the same configuration with the same parameters and weights. The training process is mirrored across all the subnetworks which means that the parameters remain same with all the subnetworks. Each subnetwork contains a traditional perceptron model like LSTM, CNN etc. The neural network compares the output of the two subnetworks through a distance metric like a cosine distance and the error is back-propagated. In the testing phase, the neural network predicts whether the two inputs are different through this similarity measure.

Siamese neural networks have been employed in many applications in different areas like signal processing [94, 95, 96, 97, 98, 99, 100, 101], biology [102, 103], chemistry and pharmacology [104], geometry [105], computer vision [106, 107, 108, 109, 110, 111, 112, 113, 114, 115], physics [116, 117], robotics [118, 119, 120], video processing [121, 122, 123, 124] etc. They have also been used in NLP tasks

[125, 126, 127] including STS [19, 128, 129]. In fact, the best result in the SICK dataset [29] was provided by a Siamese neural network [19] which shows that state-of-the-art in STS is Siamese neural networks.

In addition to providing state-of-the-art results in STS, there are additional advantage in using Siamese neural networks. As we mentioned before, in Siamese neural networks, as the weights are shared across subnetworks there are fewer parameters to train, which in turn means they require less training data and less tendency to over-fit. Given the amount of human labour required to produce datasets for STS, Siamese neural networks can provide the ideal solution for the STS task. As the second advantage, the Siamese neural network architecture when trained on a STS task, can be adopted as sentence encoders. The output vector of the subnetwork is a semantically rich vector representation of the input sentence [19]. These advantages motivate us to explore Siamese neural network architectures more in STS.

We address four research questions in this chapter:

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

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

RQ3: Can the proposed Siamese neural network be easily adopted in to different languages?

RQ4: How well 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 outperforms state-of-the-art LSTM based Siamese neural network in small STS datasets.

2. We propose a LSTM (Long short-term memory) and Self-attention based Siamese neural network that outperforms 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 initial findings of this chapter is published in Ranasinghe et al. [130].

6. 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 past research done with Siamese neural networks. Section 4.2 discusses the methodology and the experiments done with three English STS datasets. Section 4.2.1 and 4.2.2 provide more experiments to improve the results. Experiments done with

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

other languages and domains are shown in Section 4.3 and Section 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 have been very popular in the machine learning community. The first appearance of Siamese neural networks date back to 1994. It was first introduced by Bromley et al. [131] to detect forged signatures. By comparing two handwritten signatures, this Siamese neural network was able to predict if the two signatures were both original or if one was a forgery. Even before that, Baldi and Chauvin [106] introduced a similar artificial neural network able to recognize fingerprints, though by a different name.

Siamese neural networks have been applied in various applications after that. In audio and speech signal processing field, Thiolliere et al. [94] merged a dynamic-time warping based spoken term discovery (STD) system with a Siamese deep neural network for automatic discovery of linguistic units from raw speech, Manocha et al. [95] used a Siamese model to detect all the semantically similar audio clips in an input audio recording and Shon et al. [96] employed a Siamese model to recognize Arabic dialects from Arabic speech content found in media broadcasts. In biology, Zheng et al. [102] implemented a Siamese neural network to compare DNA sequences and recently Szubert et al. [103] present a Siamese neural network-based technique for visualization and interpretation of single-cell datasets. Image analysis is the field with the highest number of applications

for the Siamese neural networks. Recognising fingerprints [106], similar image detection [107, 108, 109, 110, 111], face verification [112], gesture recognition [113], hand writing analysis [114] and patch matching [115] are some of them. As you can observe, all of these tasks involve in comparing two or more things.

Recently, Siamese neural networks have been employed in NLP too. Yih et al. [125] proposed similarity Learning via Siamese Neural Network (S2Net), a technique able to discriminatingly learn concept vector representations of text words. Kumar et al. [126] used Siamese neural network to recognize clickbaits in online media outlets. González et al. [127] proposed a natural language processing application of the Siamese neural network for extractive summarization, which means that their technique can extrapolate most relevant sentences in a document. Not limited to those applications Siamese neural networks have been implemented in STS tasks too in NLP. Das et al. [128] 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. [129] employed a Siamese neural network based on Bidirectional LSTMs to identify similar job titles. The baseline we used for this chapter; MALSTM [19] uses a LSTM based Siamese neural network to perform semantic textual similarity and it provides the best results for SICK dataset outperforming other STS methods like Tree-LSTMs [18]. 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 which 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 back propagated during training, stems from the similarity between these 50 dimensional vectors, which can be also used as a sentence representation. Initially, the similarity function we used was based on Manhattan distance. To make sure that the prediction is between 0 and 1, we took the exponent of the negative Manhattan distance between 2 sentence representations. The similarity function was adopted from Mueller and Thyagarajan [19]. 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 [19]

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 [88].

3. GRU - Block A in Figure 4.1 contains a single GRU cell. GRUs have been shown to exhibit better performance on smaller datasets [87].

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 GRU [132].

5. LSTM + Attention - Block A in Figure 4.1 contains a single LSTM cell with self attention [133].

6. GRU + Attention - Block A in Figure 4.1 contains a single GRU cell with self attention [133].

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 [134].

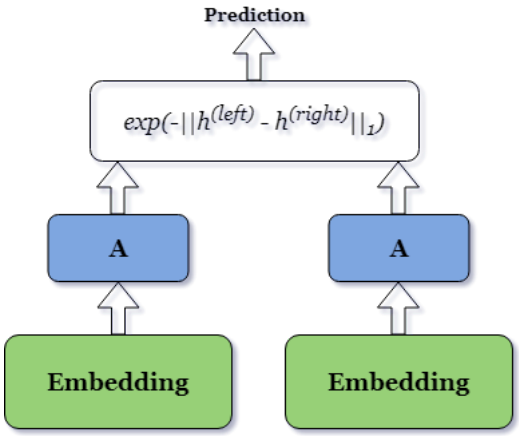


Figure 4.1: Basic structure of the Siamese neural network. Unit A is changed over the architectures.

As the word embedding model we used Word2vec embeddings [15] pre-trained

on Google news corpus[2]. We represented each word as a 300 lengthened vector using this model. For the words that do not appear in this model we used a random vector. We evaluated all 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 on those datasets and evaluated them on the testing sets. The results are shown in Table 4.1, Table 4.2 and Table 4.3 respectively.

Model	ρ	τ
LSTM	0.802	0.733
Bi-LSTM	0.784	0.708
GRU	$0.838^{\dagger}$	$0.780^{\dagger}$
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. Best result from all the variations is marked with †.

As can be seen in Table 4.1 and 4.2, for SICK and STS 2017 datasets, 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 involves Bi-directional RNNs, Attention and Capsule mechanisms did not perform well compared to the simple architectures like GRU. We can conclude that for the smaller datasets like STS 2017 and SICK, GRU based architecture performs better because GRU has less pa-

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

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. Best result from all the variations is marked with †.

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. Best result from all the variations is marked with †.

rameters than LSTM [87]. With less parameters, the architecture does not need a lot of training instances to optimise the training process.

However, when it comes to the big STS dataset; QUORA, the way that the variants of the Siamese neural network behaves is different. As we introduced in Chapter , QUORA was the biggest STS dataset we experimented which has 320,000 training instances. As a result, even the 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, LSTM + Atten-

tion 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. 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 experimented variants, one notable observation is the poor performance of capsules in Siamese architectures. Despite providing good results in many NLP tasks like text classification [134, 135] capsule based variant fails to outperform the simple LSTM based variant even in the bigger STS dataset. This implies that capsule based Siamese neural networks won' 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 the scenarios where there is not enough data to train a neural network so that starting from already tuned weights would be advantageous [136, 137]. 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 dataset; Siamese neural network with GRU and Siamese neural network with LSTM + Attention. We again initiated training for each dataset, however rather than training from scratch, we used the weights of the models trained on other STS dataset. We compared this transfer learning results to the results we got from training the model from scratch. We conducted this transfer learning experiment only on STS2017 and SICK dataset since the QUORA dataset was already big and transfer learning from a smaller dataset to a larger dataset won't make much sense.

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 transfer learning and without transfer learning. Non-grey values are the results of the experiments without transfer learning which we showed in the previous section too. We only report the Pearson correlation due to ease of visualisation.

As can be seen in Table 4.4 some of the transfer learning experiments improved the results for STS2017 and SICK datasets with both architectures. When

we performed transfer learning from STS2017 ⇒ SICK and SICK ⇒ STS2017 the results improve. This shows that transfer learning can improve the results in Siamese neural networks. However, when we performed transfer learning from QUORA ⇒ STS2017 and QUORA ⇒ SICK the results did not improve, in fact, they decrease, despite QUORA being the largest STS dataset we experimented. This finding somewhat controversial to the general belief in the community that transfer learning from a larger dataset improves the result. In this case, we believe that this happens due to the fact that the QUORA dataset is very different to the other two datasets as we 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 that were already optimised for a very different dataset; QUORA. This result in a decrease in the result. 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 considering the similarity of the two datasets too, rather than only considering the size of the dataset.

4.2.2 Impact of Data Augmentation

As we observed before, the neural networks perform better when there is a large number of training instances. Therefore, many approaches have been taken to

increase the number of training instances. Usually this has resulted better performance with neural networks [138]. Therefore, we experimented the impact of data augmentation with the Siamese neural network architectures we proposed before. We only conducted this experiment with 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 found in Wordnet [13]. A similar approach has been successfully adopted by Mueller and Thyagarajan [19], Zhang et al. [139] too. 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 the transfer learning improved the results in previous experiment, we trained the augmented training set on the transferred models; models trained on STS2017 for SICK experiments and models trained on SICK for STS2017.

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 Network. For each data augmentation experiment we show the difference between with dat augmentation and without data augmentation. We only report the Pearson correlation (ρ) due to ease of visualisation.

As can be seen in the results, data augmentation improved the results of all the experiments. However, even with the additional 10,000 training instances,

GRU based Siamese neural network outperformed LSTM + Attention based Siamese neural network. We can conclude that simple data augmentation techniques can improves the performance of Siamese neural networks in STS task. From the Siamese neural network experiments we conducted, our best result for both STS2017 and SICK datasets were provided by GRU based Siamese neural network when combined with both transfer learning and data augmentation.

This answers 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. [140]	0.807
Bjerva et al. [141]	0.827
Zhao et al. [142]	0.841
Siamese LSTM	0.863
Siamese GRU	0.882

Table 4.6: Results for SICK 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
Tian et al. [74]	0.851
Siamese LSTM	0.852
Maharjan et al. [143]	0.854
Cer et al. [28]	0.855
Siamese GRU	0.862

Table 4.7: Results for 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 Siamese neural network variant with the best results submitted to the competitions [28, 29] and with the unsupervised STS methods we have experimented so far in this part of the thesis. As can be seen in Table 4.6 and 4.7 GRU based Siamese neural network architecture outperforms the best system submitted to both competition. It also outperforms the unsupervised STS methods we have so far explored in this part of the thesis. Therefore, we can conclude that Siamese architecture is currently the best system we have experimented so far for English STS.

4.3 Portability to Other Languages

Our *RQ3* targets the multilinguality aspect of the proposed approach; *Can the proposed Siamese neural network be easily adopted in to 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 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. As word embedding models we used AraVec [70] [3] for Arabic and Spanish 3B words Word2Vec Embeddings [71][4] for Spanish.

As can be seen in Tables 4.9 and 4.8 GRU based Siamese neural network outperformed all the other variants we experimented in both Arabic and Spanish.

[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. 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. Best result from all the variations is marked with †.

As we discussed in Chapter 1, both Arabic and Spanish STS datasets we considered are small in size similar to the English STS2017 and SICK datasets. Therefore, similar to STS2017 and SICK datasets, GRU outperform other architecture as GRU does not need a lot of training instances to optimise its weights. It should be noted that it is very easy to adopt this STS method in a different language. We only changed the embeddings to the new language and performed the training.

Furthermore, we compared the results of the best Siamese neural network variant with the best results submitted to the competition [28] and with the un-

supervised STS methods we have experimented so far in this part of the thesis.

Model	ρ
Tian et al. [74]	0.744
Nagoudi et al. [144]	0.746
Siamese LSTM	0.746
Wu et al. [72]	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. [145]	0.848
Wu et al. [72]	0.850
Tian et al. [74]	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 Table 4.10 and 4.11 Siamese neural network based on GRU outperforms the top three systems of the competition in both languages. Furthermore, it outperforms the unsupervised STS methods we have experimented with so far in this part of the Thesis. Therefore, we can conclude that Siamese neural network based on GRU is currently the best system we have experimented so far for Arabic and Spanish too.

This answers our **RQ3:**, the Siamese architectures that we propose in this chapter, can be successfully adopted in different language by changing the word

embeddings and the training dataset.

4.4 Portability to Other Domains

In order to answer our *RQ4*; how well the proposed Siamese neural network architecture can be applied in different domains, we evaluated our method on Bio-medical STS dataset explained in 1 (BIOSSES). As we mentioned before Bio-medical STS dataset does not have a training set. Therefore, we had to follow a transfer learning strategy to evaluate it 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 call it 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 model; the general Word2vec model we used before [15] that were pre-trained on Google news corpus and BioWordVec [76] 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 on English experiments) and evaluated them on the BIOSSES dataset.

As you can see in the Table 4.12 Siamese neural architecture provided satisfactory results. We got the best result from Siamese neural network based on GRU when trained on STS 2017 using BioWordVec. However, the results from SICK dataset is also not far behind. There was a clear improvement when

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

Model	Word2vec	BioWordVec
$STS2017_{GRU}$	0.651	0.721
$STS2017_{LSTM+Aten}$	0.612	0.701
$SICK_{GRU}$	0.642	0.719
$SICK_{LSTM+Aten}$	0.608	0.699
$QUORA_{GRU}$	0.591	0.622
$QUORA_{LSTM+Aten}$	0.603	0.634

Table 4.12: Results for transfer learning with different variants of Siamese Neural Network in BIOSSES dataset. Two considered word embedding models are Word2vec and BioWordVec. We only report the Pearson correlation due to ease of visualisation.

the English STS model was trained using BioWordVec rather than using general Word2vec embeddings. This can be due to the fact that most of the Bio-medical words that appear in BIOSSES dataset are out of vocabulary in general Word2vec embeddings which can cause problems to the neural network when it observes them in the testing phase. Furthermore, it should be noted that in this experiment too, when we performed transfer learning from QUORA dataset the results are lower than performing transfer learning from SICK or STS 2017. This again can be due to the reason SICK and STS 2017 datasets have a similar annotation strategy to the BIOSSES dataset as we 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.

Furthermore we compared our results with the best results reported for the dataset. The results are shown in Table 4.13.

As shown in the results, our method provides satisfactory results when compared with best approaches. However, it should be noted that the unsupervised method we experimented in the previous chapter with BioSentVec [92] com-

Model	ρ
ELMo $\bigoplus$ *BERT*	0.708
$STS2017_{GRU}$	0.719
Soancolu et al. [39]	0.754
BioSentVec [92]	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.

fortably outperformed Siamese neural network approaches we explored in this chapter. We can answer our **RQ4: How well the proposed Siamese neural network perform in a different domain?** with these findings. The Siamese neural network architectures can be adopted to different domains by changing the pre-trained word embeddings. However, without a proper training set the results won't be strong.

4.5 Conclusions

This chapter experimented 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 [19] 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 Siamese neural network based on GRU outperforms the baseline and 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 improve the results

more with transfer learning and data augmentation techniques. However, we experienced that performing transfer learning from a bigger dataset won't always improve the results. The quality of the dataset which was used for transfer learning matter too. We show that Siamese neural network based on GRU performs better than the top submissions in both SemEval 2017 task 1 [28] and SemEval 2014 task 1 [29]. The data augmentation techniques we used in this chapter are language dependent as they rely on WordNet [13]. However, as future work we can consider data augmentation techniques that are not language dependant and relies on word embeddings by itself [146].

We extended the experiments with Siamese neural network architectures to Arabic and Spanish STS datasets in SemEval 2017 [28]. In them too the GRU based Siamese neural network architecture outperformed all the systems submitted to the shared task and also outperformed all the STS methods we have explored so far in this part of the thesis. This proves that the Siamese neural network that we propose here can be adopted in different languages. Furthermore, we performed experiments with the BIOSSES dataset. However since the BIOSSES dataset does not have a training set, we had to use transfer learning based zero-shot learning when we are applying Siamese neural networks to this dataset. Even though they provided satisfactory results, Siamese neural networks could not outperform the sentence vector based method we explored in Chapter 2. We can conclude that despite the fact that the Siamese neural networks can be adopted in different domains by changing the word embedding model, they won't provide strong results without a proper training set.

Since word embedding model are now available in most of the languages including the low resource languages like Urdu [147], Telugu [148] and domains like legal domain [149], this method we explored in this chapter can be useful for many languages and domains. However, one drawback is that the need for STS training data in each language and domain which can be challenging in many scenarios.

As future work, it would be interesting to experiment transfer learning between languages with cross-lingual embeddings like fastText [45] using Siamese neural networks. Such approach will be able to train a STS model on resource rich language like English and project the prediction for other languages using zero-shot transfer learning we experimented here. It would be a potential solution for the training data requirement for the low resource languages.

With the introduction of transformer models like BERT [20], Siamese neural networks has evolved incorporating transformers in their architectures too [150]. We will discuss them in Chapter 5.

Chapter 5

Adopting Transformers for STS

Transformer models adopt a pre-training followed by fine-tuning scheme which means that once pre-trained these models can be fine-tuned to a large number of down-stream NLP tasks like text classification, named entity recognition etc. Transformer models, which we have considered 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 sentence-level fine-tuning tasks and the final hidden state of each token is used for token-level fine-tuning tasks. The fine-tuning scheme in transformers is usually simple like adding a softmax layer on top of [CLS] token for the text classification tasks. Furthermore, the fine-tuning scheme is very efficient as the parameters in the transformer model are already optimised with the pre-training process. Therefore, transformer models have been extremely popular and successful in many NLP tasks [20].

In this chapter, we explore different transformers models in variety of STS experiments. We address four research questions in this chapter:

RQ1: How well the existing state-of-the-art transformer models perform in

STS task?

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

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

RQ4: How well 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 methods and show that transformer based STS methods outperform all the other STS methods we have experimented.

2. We propose further enhancements to the architecture using inter dataset 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.

[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/

5.1 Related Work

As we mentioned before, after the introduction of BERT [20], many variants of different transformer models have been proposed by adding minor modifications to the original BERT transformer. Usually these modifications has resulted in improvements in the fine-tuning scheme for the down-stream NLP tasks. Expecting a similar behaviour for the STS task too, we considered following transformer models for the experiments in this chapter.

BERT [20] proposes a masked language modelling (MLM) objective, where some of the tokens of a 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 actual 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 [22] makes a few changes to the BERT architecture and achieves substantial improvements. The 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.

ALBERT [151] proposes two parameter-reduction techniques (factorised embedding parameterisation and cross-layer parameter sharing) to lower memory consumption and speed up training. Furthermore, ALBERT [151] argues that the NSP objective in BERT lacks difficulty, as the negative examples are created by pairing segments from different documents, this mixes topic prediction and coherence prediction into a single task. ALBERT instead 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.

ELECTRA Compared to BERT, ELECTRA [152] proposes a more 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 each token in the corrupted input was replaced by the generator 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.

XLNET [21] identifies a key weakness in BERT pre-training. Yang et al. [21] argues that the symbols such as [MASK] that are introduced by BERT during pre-training, causes discrepancy between pre-training and fine-tuning as they never occur in real data. Therefore, XLNET suggests a new auto-regressive method based on permutation language modelling (PLM) [153] without introducing any

new symbols.

Upon the introduction all of these transformer models have been evaluated in many down-stream NLP tasks including STS too. However, there is no comprehensive study done on STS using transformers considering big and small 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 mean-squared-error loss as the objective function. As the configurations, we used a batch-size of eight, Adam optimiser with 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, 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 after each 100 batches, 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. As these transformer models are computationally expensive, we used an Nvidia Tesla T4 GPU for the training process. We have kept these configuration same for all the experiments; in order to ensure consistency between all the languages

and domains. This also provides a good starting configuration for researchers who intend to use transformers on a new language pair. The implementation is based on PyTorch [154] and HuggingFace [155].

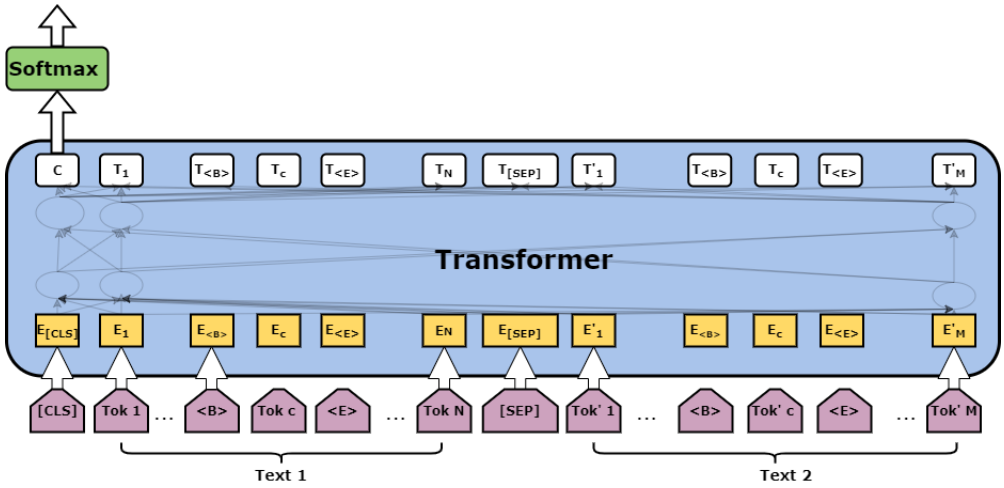


Figure 5.1: Architecture for using Transformers in STS.

5.3 Exploring Transformers in English STS

We evaluated all the above transformer variations in the three English STS datasets we introduced in 1; SICK, STS 2017 and QUORA. All of the transformer models we considered have several models that supports English (e.g. *bert-large-cased* & *bert-base-cased* for BERT, *text*).

We trained the transformer models on the training sets on those datasets and evaluated them on the testing sets.

The results are shown in Table 5.1, Table 5.2 and Table 5.3 respectively.

Model	ρ	τ
BERT	0.802	0.733

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. Best result from all the variations is marked with †.

Model	ρ	τ
LSTM	0.831	0.762
Bi-LSTM	0.784	0.708
GRU	$0.853^{\dagger}$	$0.811^{\dagger}$
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 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. Best result from all the variations is marked with †.

Model	RMSE
LSTM	0.412

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. Best result from all the variations is marked with †.

Part II

Applications - Translation Memories

Chapter 1

Introduction

1.1 What is Translation Memory?

[156]

1.2 Datasets

1.3 Related Work

1.4 STS for Translation Memories

Chapter 2

Sentence Encoders for Translation Memories

[157]

2.1 Methodology

2.2 Results and Evaluation

Part III

Applications - Translation Quality Estimation

Chapter 1

Introduction

1.1 What is Translation Quality Estimation?

1.2 Datasets

We used the architectures described above to predict two standard measures that express the quality of a translation: Human-mediated Translation Edit Rate (HTER) and Direct Assessment (DA). All the datasets that we used are publicly available and were released in WMT quality estimation tasks in recent years [158, 159, 160]. This was done to ensure replicability of our experiments and to allow us to compare our results with the state of the art. In the reminder of the section we provide more details about the datasets used.

1.2.1 Predicting HTER

The performance of QE systems has typically been assessed using the semiautomatic HTER (Human-mediated Translation Edit Rate). HTER is an edit-distance-based measure which 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 lan-

Language Pair	Source	MT system	Competition	train, dev, test size
De-En	Pharmaceutical	Phrase-based SMT	WMT 2018 [158]	25,963, 1,000, 1,000
En-Zh	Wiki	fairseq based NMT	WMT 2020 [160]	7,000, 1,000, 1,000
En-Cs	IT	Phrase-based SMT	WMT 2018 [158]	40,254, 1,000, 1,000
En-De	IT	fairseq based NMT	WMT 2019 [159]	13,442, 1,000, 1,000
En-De	IT	Phrase-based SMT	WMT 2018 [158]	26,273, 1,000, 1,000
En-Ru	Tech	Online NMT	WMT 2019 [159]	15,089, 1,000, 1,000
En-Lv	Pharmaceutical	Attention-based NMT	WMT 2018 [158]	12,936, 1,000, 1,000
En-Lv	Pharmaceutical	Phrase-based SMT	WMT 2018 [158]	11,251, 1,000, 1,000

Table 1.1: 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.

guage pairs for which HTER information was available: English-Chinese (En-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 1.1 and in [158, 159, 160].

1.2.2 Predicting DA

Even though HTER has been typically used to assess quality in machine translations, the reliability of this metric for assessing the performance of quality estimation systems has been questioned by researchers [161]. The current practice in MT evaluation is the so-called Direct Assessment (DA) of MT quality [162], where raters evaluate the machine translation on a continuous 1-100 scale. This

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

method has been shown to improve the reproducibility of manual evaluation and to provide a more reliable gold standard for automatic evaluation metrics [163].

We used a recently created dataset to predict DA in machine translations which was released for the WMT 2020 quality estimation shared task 1 [160]. 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 [164]. 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 [165] and annotated with DA scores by professional translators. Each translation was rated with a score from 0-100 according to the perceived translation quality by at least three translators [160]. 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.

1.3 Related Work

[166]

1.4 STS for Translation Quality Estimation

1.5 Conclusion

Chapter 2

TransQuest: STS Architectures for QE

Neural-based QE methods constitute the state-of-the-art in quality estimation. This is clear as in recent years, neural-based QE systems have consistently topped the leader boards in WMT quality estimation shared tasks [166]. 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 [167]. As a result most of the recent open-source QE frameworks like OpenKiwi [166] and DeepQuest [168] 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 [166] is the stacked model that used the Predictor-Estimator model we described in Chapter 1 which requires extensive predictor pre-training and relies on a large parallel dataset and computational resources. This is similar for POSTECH architecture in DeepQuest [168] too. This complex nature of the state-of-the-art QE frameworks has hindered their ability to be applied in real-life applications.

To overcome this, we propose to remodel the QE task as a crosslingual STS

task. In other words, measuring the quality between a source and a target can be interpreted as computing the crosslingual 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 QE task. However, since we used monolingual embeddings for STS, we need to change them so that the embeddings can represent both languages in QE task. For that, we propose to use crosslingual embeddings. We assume that using the crosslingual embeddings that are 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 the state-of-the-art QE architectures in OpenKiwi [166] and DeepQuest [168]. Furthermore, crosslingual embeddings that we intend to use with the STS architectures are already fine-tuned to reflect the properties between languages of source and target. As a result, this removes the dependency on large parallel data, which also entails the need for powerful computational resources. This motivated us to explore these architecture in QE task. We believe that simpler and efficient architectures would improve the popularity of QE in real-life applications. As far as we know, this would be the first work to apply STS architectures in 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 task by just modifying the embeddings?

RQ2: Does crosslingual embeddings have an advantage over multilingual embeddings in the QE task?

RQ3: Can the models 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 [169, 170].

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 [168] and OpenKiwi [166].

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

4. The findings of this chapter are published in Ranasinghe et al. [171].

5. The two architectures introduced here were released as part of a QE framework; *TransQuest*. TransQuest participated in WMT 2020 QE shared task 1[1] [160] and won it in all the language pairs outperforming 50 teams around the globe [172].

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

6. 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 2.1 discusses the methodology and the experiments done with 15 language pairs in both aspects of sentence-level QE. Section 2.2 shows the results and Section 2.3 provide further fine-tuning strategies to improve the results. In Section 2.4, we provide a basic error analysis. The chapter finishes with conclusions and ideas for future research directions in QE.

2.1 Methodology

As we mentioned before, we considered sentence-level QE as a crosslingual 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 have to use crosslingual Transformers for the QE task, that can represent both languages in the same vector space.

Although there are several multilingual transformer models like multilin-

[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

gual BERT (mBERT) [20] and multilingual DistilBERT (mDistilBERT) [173], researchers expressed some reservations about their ability to represent all the languages [174]. In addition, although mBERT and mDistilBERT showed some crosslingual characteristics, they do not perform well on crosslingual benchmarks [175] as they have not being trained using crosslingual data.

XLM-RoBERTa (XML-R) was released in November 2019 [176] as an update to the XLM-100 model [177]. 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[22] 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 [22] masked language modelling (MLM) objective. Surprisingly, this strategy provided better results in crosslingual tasks. XLM-R outperforms mBERT on a variety of crosslingual benchmarks such as crosslingual natural language inference and crosslingual question answering [176]. This superior performance of XLM-R in crosslingual 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 [176] 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 the way 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 computational intensive than state-of-the-art QE solutions like predictor-estimator [169, 170].

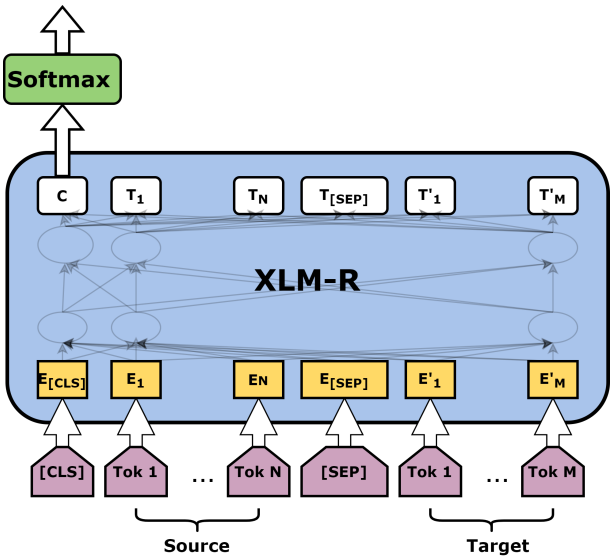
There are two pre-trained XLM-R models released by HuggingFace's Transformers library [155]; XLM-R-base and XLM-R-large. We used both of these pre-trained models in our experiments[5]. Both of these pre-trained XLM-R models cover 104 languages [176], making it potentially very useful to estimate the 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 2.1a. This 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 out-

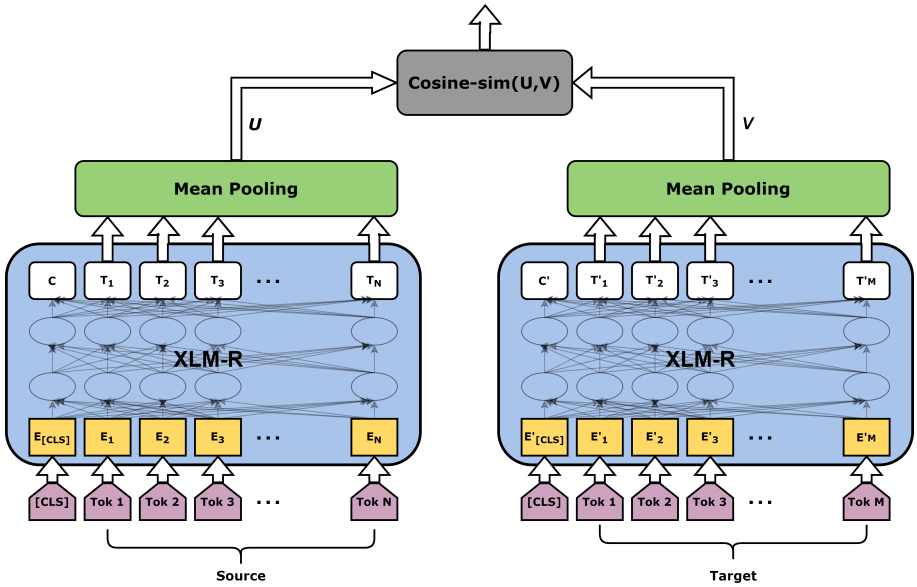
[5]XLM-R-large is available on `https://huggingface.co/xlm-roberta-large` and XLM-R-base is available on `https://huggingface.co/xlm-roberta-base`

put 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 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 paper relies on the Siamese architecture depicted in Figure 2.1b showed promising results in monolingual STS tasks [150] 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 mean-squared-error loss as the objective function. In 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.



(a) *MonoTransQuest* Architecture



(b) *SiameseTransQuest* Architecture

Figure 2.1: Two architectures employed in TransQuest.

2.1.1 Experimental Setup

We evaluated these two architectures in both aspects of sentence-level quality estimation using the data introduced in Chapter 1. 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 learning rate 2e−5, and a linear learning rate warm-up over 10% of the training data. During the training process, the parameters of XLM-R 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 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.

2.2 Results and Discussion

The results for HTER and DA aspects are shown in Tables 2.1 and 2.2. 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 [158, 159, 160] as mentioned in Chapter 1. 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 2.1 and 2.2 shows the results for MonTransQuest and SiameseTransQuest with XLM-R-large and Raw II in Tables 2.1 and 2.2 shows the results for MonTransQuest 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

	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 2.1: Pearson correlation (ρ) between *TransQuest* algorithm predictions and human post-editing effort. Best results for each language by any method are marked 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 2.3. 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.

we got from XLM-R-large for the following analysis.

In the HTER aspect of sentence-level quality estimation, as shown in Table 2.1, *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 2.1 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

		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 2.2: Pearson correlation (ρ) between *TransQuest* algorithm predictions and human DA judgments. Best results for each language (any method) are marked 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 2.3. Row V shows the results of the baseline method; OpenKiwi. Row VI presents the results of the multilingual BERT (mBERT) model in TransQuest Architectures.

the exception of the En-De SMT, En-De NMT and En-Zh NMT datasets. *SiameseTransQuest* fall behind *MonoTransQuest* architecture, however still it outperforms OpenKiwi and Quest++ in all the language pairs except En-De SMT. This shows that simple STS models with crosslingual embeddings outperforms the complex state-of-the-art QE models in predicting HTER.

As shown in Table 2.2, in the DA aspect of quality estimation, *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 too *SiameseTransQuest* fall 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 crosslingual embeddings outperforms the complex state-of-the-art QE models in predicting DA too.

If you 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 inline with 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 a 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.

[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 these results we can answer our **RQ1:** Can the state-of-the-art STS models applied in sentence-level QE task by changing the embeddings? We show that state-of-the-art STS methods based on crosslingual 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 2.1 and 2.2 shows the results of multilingual BERT (mBERT) in MonoTransQuest and SiameseTransQuest architecture. 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 2.1 and 2.2 even XLM-R-base model outperform mBERT model in all the languages pairs with both architectures. Therefore, we can safely assume that the cross lingual nature of the XLM-R transformers had a clear impact to the results.

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

2.3 Further Fine-tuning

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

Ensemble Learning uses multiple learning algorithms to obtain better predictive performance than could be obtained from 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 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 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 2.1 and 2.2 as MTransQuest ⊗ and STransQuest ⊗.

As shown in Tables 2.1 and 2.2 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 of the machine learning algorithms. Therefore, to experiment how TransQuest performs with more data, we trained TransQuest on a data augmented setting. Alongside the training, development and testing datasets, the shared task organisers also provided the parallel sentences which were 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 training dataset we used to train TransQuest. In order to find the best setting for the data augmentation we experimented with adding 1000, 2000, 3000, up to 5000 sentence pairs randomly. 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 imbalance and affect the performance of the machine learning models negatively. We report the results from the two architectures from this step in row IV of Tables 2.1 and 2.2 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 2.1 and 2.2 both architectures in TransQuest with the data augmentation setting gained ≈ 0.01-0.09 Pearson correlation boost over the XLM-R-large 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 used to improve the results? We show that ensemble learning and data augmentation can be used improve the results. In fact, data augmentation combined with ensemble learning provided 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 [160]. The official results of the competition show that *TransQuest* won the first place in all the language pairs in 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 the first place with Fomicheva et al. [178], whose results are not statistically different from ours. Therefore, TransQuest is the new state-of-the-art in sentence-level QE.

2.4 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 choice of language pairs we analysed was determined by the availability of native speakers to perform this analysis. We focused on the cases where the difference between the predicted score and expected score was the greatest. This included both cases where the predicted score was underestimated and overestimated.

Analysis of the results does not reveal very clear patterns. The largest number of errors seem to be caused by the presence of 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 urm explorrilor Cpitanului James Cook, Australia i Noua Zeeland au devenit inte 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 alt problem important cu care trupele Antantei au fost obligate s 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). In the opinion of the authors of this paper, it is debatable whether the expected scores for these two pairs should be so different. Both of them have obvious problems and cannot be clearly understood without reading the source. For this reason, we would expect that both of them have low scores. Instances like this also occur in the training data. As a result of this, it may be that *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 fâiile în poziie vertical (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 run the system with the pair *RO: Drapelul cu fâiile în poziie vertical - 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 the majority of problems seem to be caused by the presence of named entities in the source sentences. For an 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 a 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.

2.5 Conclusion

In this paper we introduced *TransQuest*, a new open source framework for quality estimation based on cross-lingual transformers. *TransQuest* is implemented

using PyTorch [154] and HuggingFace [155] and supports 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. As far as we know, it is the first time that STS architectures have been proposed to the QE task. 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 the first place in all the language pairs. We can conclude that TransQuest is the 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 a 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 researches, TransQuest is popular among the industry too. ModelFront, which is a leading company in translation risk predic-

[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

tion lists TransQuest as an option in their website[9]. We believe that the simplicity of the architectures in TransQuest is the reason for this popularity.

Sentence-level QE models are difficult 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 3. One limitation with TransQuest is that the pre-trained QE models are big in size as they depend on transformer models. Therefore, managing a 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 4.

As future work, we hope to explore crosslingual embeddings more with TransQuest for the sentence-level QE task. For an example XeroAlign [179] provides a simple method for task-specific alignment of crosslingual pre-trained transformers which would be interesting to apply in QE task too. Furthermore, Facebook has introduced large crosslingual models recently; XLM-R XL and XLM-R XXL which have improved results for many crosslingual NLP tasks [180]. These models have obvious disadvantage of being big in size; yet would be interesting to apply in QE.

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

Chapter 3

Extending TransQuest for Word-Level QE

Translation quality can be estimated at different levels of granularity: word, sentence and document level [168]. So far in this thesis, we have only explored sentence-level QE [160], 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 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 we mentioned before, sentence-level QE models are difficult to explain as they just outputs a single score for the translation. Users would face the difficulty of interpreting the score and deciding the steps after the QE process. However, in contrast to that word-level QE models provide more fine-grained information about the quality of a translation and as a result, they can

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

With these advantages researches 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 1, most of the open source quality estimation frameworks like OpenKiwi, uses same architectures for both word-level and sentence-level QE. For example the predictor-estimator architecture used in OpenKiwi provides best results for word-level QE too. Therefore word-level QE also suffers from the same limitation that we mentioned in 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.

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 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 retain word-level qualities. This architecture is very simple and effective when compared with the current state-of-the-art word-level QE architectures like predictor-estimator. We believe that a simpler architecture like that would improve the popularity of word-level QE in real-world applications.

As we mentioned in Chapter 1 in Part III of this thesis, 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 quality of the gaps in the target sentence. For example Marmot [181] only supports predicting the quality of the words in target and OpenKiwi [166] only supports predicting the quality of the words in source and target. They completely ignore predicting the quality of the gaps. Despite that, predicting the quality of the gaps in 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 quality of the words in target, predicting quality of the words in source and predicting quality of the gaps in target. We believe an approach that supports every feature in word-level QE would 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 features in word-level QE task by just modifying the embeddings and output layer?

RQ2: Does crosslingual embeddings have an advantage over multilingual embeddings in the word-level QE task too?

RQ3: Can the proposed model 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 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 [181] and OpenKiwi [166].

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

4. The initial findings of this chapter are published in Ranasinghe et al. [182].

5. We integrated the architecture with TransQuest, which already had two sentence-level architectures we described in Chapter 2[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 3.1 discusses the methodology and the experiments done with eight language pairs in word-level QE. Section 3.2 shows the results and perform the evaluation on three evaluation

[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

metrics uses in word-level QE. Section **??** provide further fine-tuning strategies to improve the results. The chapter finishes with conclusions and ideas for future research directions in word-level QE.

3.1 Methodology

The proposed architecture for the word-level QE is very similar to the *MonoTransQuest* architecture in Chapter 2 where the source and the target is processed through a single transformer model. The difference here is since we need quality for each individual word, we do not use a pooling mechanism like we did with *MonoTransQuest*. Instead we keep the state of the individual tokens to get their quality.

As we need to consider the quality of the GAPs too 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 in to a single transformer. A simple linear layer is added on top of word and <GAP> embeddings to predict whether it is "Good" or "Bad" as shown in Figure 3.1. 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 smallest unit possible in QE, we named the proposed word-level architecture as

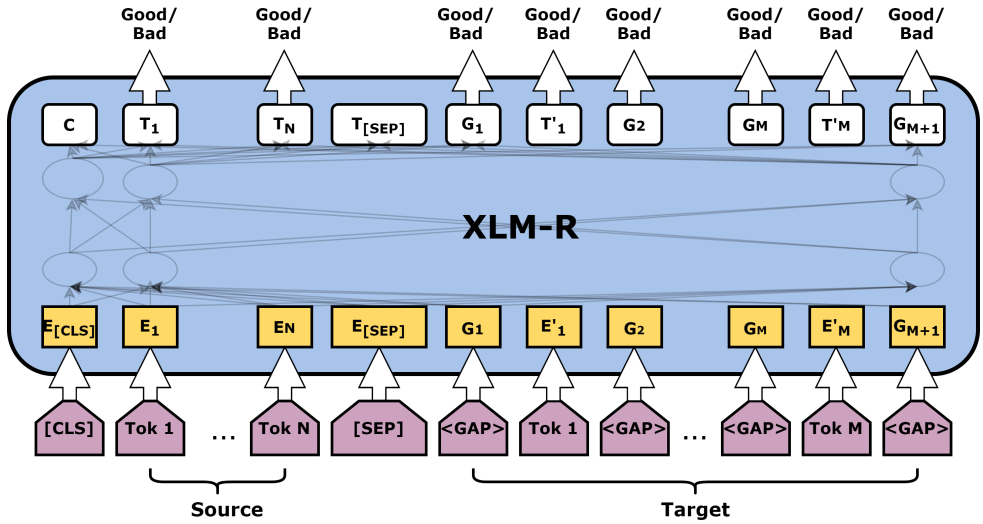


Figure 3.1: MicroTransQuest Architecture

MicroTransQuest.

Similar to the sentence-level QE architectures, we used crosslingual transformer models for this architecture too. As we explained in Chapter 2 XLM-R provides state-of-the-art crosslingual transformer models. There are two pre-trained XLM-R models released by HuggingFace's Transformers library [155]; 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 [176], making it potentially very useful to estimate the word-level translation quality for a large number of language pairs.

3.1.1 Experimental Setup

We evaluated the architecture in word-level quality estimation using the data introduced in Chapter 1. We applied the same set of configurations for all the

[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

language pairs in order 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 learning rate 1e−5, and a linear learning rate warm-up over 10% of the training data. During the training process, the parameters of XLM-R 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 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.

3.2 Results and Evaluation

For 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) [158] as we explained in Chapter 1 in Part III of the thesis. 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

	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 3.1: F1-Multi Target between the algorithm predictions and human annotations. Best results for each language by any method are marked 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 3.3. 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.

shared task.

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 3.1, 3.2 and 3.3 respectively. Prior to 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 3.2. For the language pairs in WMT 2019 and 2020 we report a single result for target gaps and words as (F1-Multi Target) in Table 3.1.

Raw I in Tables 3.1, 3.2 and 3.3 shows the results for MicroTransQuest with XLM-R-large and Raw II in Tables 3.1, 3.2 and 3.3 shows the results for Micro-

	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 3.2: F1-Multi GAPS between the algorithm predictions and human annotations. Best results for each language by any method are marked 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 3.3. 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.

TransQuest 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 3.1, *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 ≈ 0.15-0.2 F1-Multi score boost over Marmot in all the language pairs. Table 3.1 also 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

	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 3.3: F1-Multi Source between the algorithm predictions and human annotations. Best results for each language by any method are marked 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 3.3. 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.

all the language pairs with the exception of the En-De NMT and En-Zh NMT datasets.

In F1-Multi GAP evaluation metric in word-level QE, as shown in Table 3.2 *MicroTransQuest* outperforms best systems submitted to the respective shared tasks in all the language pairs we experimented. *MicroTransQuest* achieves ≈ 0.05-0.2 F1-Multi score boost over best systems. Notably *MicroTransQuest* achieves ≈ 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 target sentence. Since *MicroTransQuest* supports this and produces state-of-the-results in this evaluation metric too, we believe that using *MicroTransQuest* in word-level QE would be beneficial than

using OpenKiwi and Marmot.

Finally in F1-Multi Source evaluation metric too, as shown in Table 3.3 *Micro-TransQuest* 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 source. On the other hand, OpenKiwi supports this, but still *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 STS architecture be used to predict all features in word-level QE task by just modifying the embeddings and the output layer? We show that state-of-the-art STS method based on crosslingual transformer models can be used in word-level QE too by just changing the output layer and it outperforms current state-of-the-art word-level QE methods like 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 3.1, 3.2 and 3.3 shows the results of multilingual BERT (mBERT) in MicroTransQuest architecture. 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. As shown in row II in Tables 3.1, 3.2 and 3.3 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 to the results. This observation is similar to what we experienced in Chapter 2 with sentence-level QE experiments.

With this we can answer our **RQ2:** Using crosslingual embeddings with STS architecture proved advantageous rather than using just 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 quality of the gaps in target, we show that XLM-R can be used in tasks similar to detecting missing words too.

3.3 Further Improvements

In this section. we try improve the results of MicroTransQuest through ensemble learning as we did with sentence-level quality estimation. As me mentioned before in Section 3.1 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 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 combi-

nations that gives higher weight to XLM-R-base transformer model results. We obtained best results when we used the weights 0.6:0.4. We report the results from this step in row III of Tables 3.1, 3.2 and 3.3 as MicroTransQuest ⊗.

As shown in Tables 3.1, 3.2 and 3.3 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 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.

3.4 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 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 target sentence. The architecture we proposed in this Chapter is very simple when compared with the predictor-estimator architecture in OpenKiwi, yet this architecture produce better results. We further improved the results with ensemble learning showed that *MicroTransQuest* outperforms majority of the best systems submitted for that language in each shared task. We can conclude that state-of-the-art STS architecture can be used in word-level QE too by doing small modifications to the output layers.

The implementation of the architecture, which is based on PyTorch [154] and Hugging Face [155], has been integrated into the TransQuest framework [171] which won the WMT 2020 QE task [160] on sentence-level direct assessment [172][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 word-level and sentence-level qualities separately would be chaotic in some situations. Therefore, in the future we hope to explore multi-task learning for word-level and sentence-level QE [183]. 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 at the same time can be advantageous.

As we discussed at multiple points in this thesis, pre-trained models based on transformers are big in size. Therefore, managing a several of them for each

[4]The details about the word-level QE architecture in TransQuest is available on http://tharindu.co.uk/TransQuest/architectures/word_level_architecture/

language pair would be chaotic in a real-world application for word-level QE too. As a solution for that we explore multilingual word-level QE models in Chapter 4 in Part III of the thesis.

Chapter 4

Multilingual Quality Estimation with TransQuest

Machine translation quality estimation is traditionally framed as a supervised machine learning problem [166, 169] 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 each machine learning model for each language pair.

This traditional approach has obvious drawbacks. As we 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, which makes 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 [171].

To understand the scale of this, consider a real-word application where it is required to build a quality estimation solution for European Parliament. Euro-

pean Parliament has 24 languages which would result in 24*23 language pairs which is 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 [184, 185] including NMT [186, 187], multilingual approaches have been rarely used in QE [188]. Therefore in this chapter we explore multilingual models with the *TransQuest* architectures we introduced in Chapters 2 and 3 for sentence-level QE and word-level QE respectively. Since we used a crosslingual transformer model that supports 104 languages [176] it was possible to explore multilingual learning with the same setup.

We address three research questions in this chapter:

RQ1: Do multilingual models based existing state-of-the-art sentence-level and word-level QE architectures perform competitively with the related bilingual models?

RQ2: How does 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 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 multilingual pre-trained models of *TransQuest* are publicly available to the community[1].

4.1 Methodology

We explored word-level multilingual QE too with the same datasets we introduced in Chapter 1 in Part III of the Thesis. We conducted similar experiments we performed in multilingual sentence-level QE to word-level QE. We used the *MicroTransQuest* architecture with XLM-R-large model [176] for the multilingual word-level QE models. We did not use the ensemble models to keep the multilingual experiments simpler. We applied the same set of configurations for all the training processes in order to ensure consistency between all the experiments.

[1]The pre-trained multilingual QE models are available on HuggingFace model repository on https://huggingface.co/TransQuest

We used a batch-size of eight, Adam optimiser with learning rate 1e−5, and a linear learning rate warm-up over 10% of the training data. During the training process, the parameters of XLM-R 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 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.

4.2 Results

For the evaluations we used the same evaluation metrics we used for bilingual word-level QE in Chapter 3 in Part III of the thesis which in turn was used for 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 4.3, 4.4 and 4.5 respectively. The values displayed diagonally across row I of Tables 4.3, 4.4 and 4.5 show the results for supervised, bilingual, word-level 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 row I of Tables 3.1, 3.2 and 3.3 in last chapter which we got with *MicroTransQuest* using XLM-R-large model [176]. In the following sections, we explore different settings of multilingual word-level QE and we compare the multilingual results to these results we got with supervised,

		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 4.1: Pearson correlation (ρ) between *MonoTransQuest* algorithm predictions and human post-editing effort. Best results for each language by any method are marked 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.

bilingual, word-level QE models.

4.2.1 Multilingual QE

We combined instances from all the language pairs and built a single word-level QE model. Our results, displayed in row II ("All") of Tables 4.3, 4.4 and 4.5, show that multilingual models perform on par with bilingual models or even better for some language pairs in all the evaluation metrics. For example, as shown in Table 4.3, multilingual word-level QE model outperforms bilingual word-level

		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 4.2: Pearson correlation (ρ) between *MonoTransQuest* algorithm predictions and human DA judgments. Best results for each language by any method are marked 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.

QE model in all the language pairs except En-Zh NMT, En-Ru NMT and De-En SMT with regard to Target F1-Multi. Similar observations can be made in other evaluation metrics too in Tables 4.4 and 4.5.

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. However as shown in sections III and IV ofTables 4.3, 4.4 and 4.5, for the majority of the language pairs, specialised multilingual word-level QE models built on certain domains or MT types do not perform better than multilingual models which con-

		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 4.3: Target F1-Multi between MicroTransQuest predictions and human annotations in Multilingual Experiments. Best results for each language by any method are marked 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.

tain all the data.

With these obseverations, we revisit our **RQ1** with regard to word-level QE. Multilingual models based on existing state-of-the-art word-level QE architectures perform competitively with the related bilingual models and in majority of the language-pairs multilingual models outperformed the related bilingual models. This is similar to the observations we had with sentence-level multilingual QE.

		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 4.4: GAP F1-Multi between MicroTransQuest predictions and human annotations in multilingual experiments. Best results for each language by any method are marked 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.

4.2.2 Zero-shot QE

To test whether a word-level QE model trained on a particular language pair can be generalised to other language pairs, different domains and MT types, we performed zero-shot quality estimation. 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 4.3, 4.4 and 4.5 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**

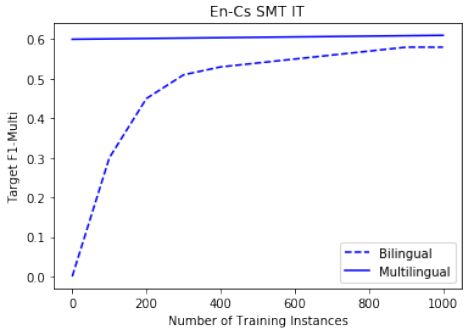
Table 4.5: SOURCE F1-Multi between MicroTransQuest predictions and human annotations in multilingual experiments. Best results for each language by any method are marked in bold. Row I, II, III and IV indicate the different multilingual settings. Row V shows the results of the baselines and the best 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 of Tables 4.3, 4.4 and 4.5 show by how much the score changes when the zero-shot word-level QE model is used instead of the bilingual word-level QE model. As can be seen, the scores decrease, but this decrease is negligible and is to be expected. For most pairs, the word-level 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 analysing

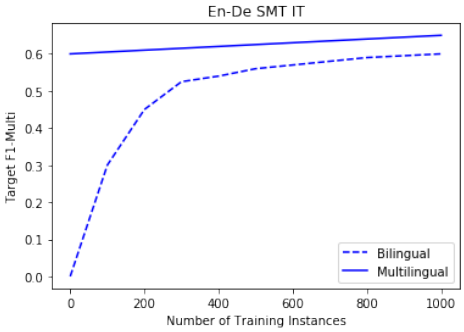
the results, we can see that zero-shot word-level QE performs better when the language pair shares some properties such as domain, MT type or language direction. For example, En-De SMT ⇒ En-Cs SMT is better than En-De NMT ⇒ En-Cs SMT and En-De SMT ⇒ En-De NMT is better than En-Cs SMT ⇒ En-De NMT in Target F1-Multi. Similar observations can be made on other evaluation metrics too.

We also experimented with zero-shot QE with multilingual word-level QE models. We trained the word-level QE model in all the pairs except one and performed prediction on the test set of the language pair left out. In section II ("All-1"), we show its difference to the multilingual word-level 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.

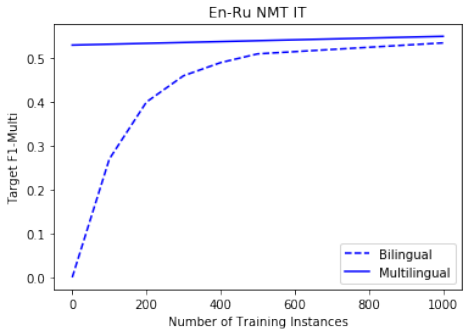
One limitation of the zero-shot QE is its inability to perform when the language direction changes. In the scenario where we performed zero-shot learning from De-En to other language pairs, results degraded considerably from the bilingual result. Similarly, the performance is rather poor when we test on De-En for the multilingual zero-shot experiment as the direction of all the other pairs used for training is different. This is in line with results reported by Ranasinghe et al. [171] for sentence level.



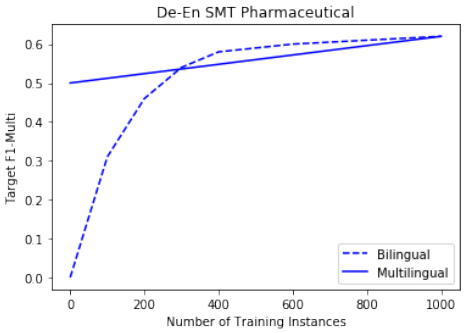
(a) En-Cs SMT IT



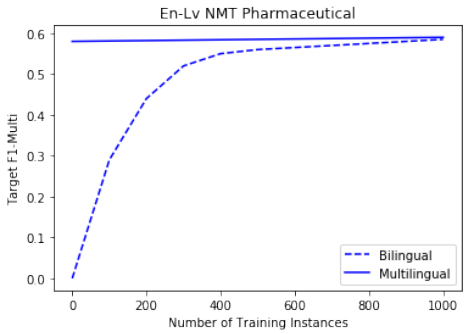
(b) En-De SMT IT



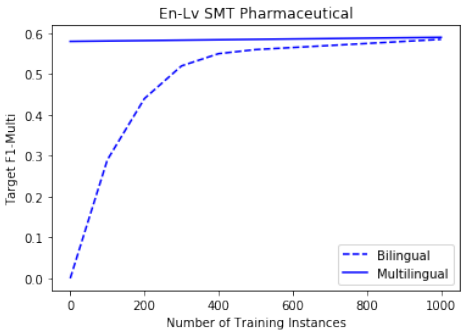
(c) En-Ru NMT IT



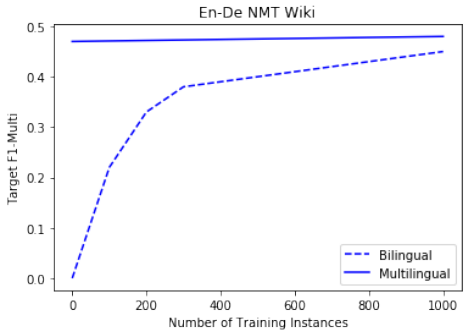
(d) De-En SMT Pharmaceutical



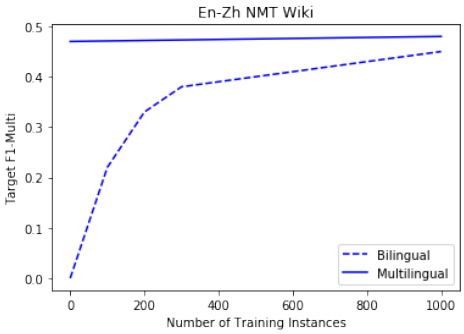
(e) En-Lv NMT Pharmaceutical



(f) En-Lv SMT Pharmaceutical



(g) En-De NMT Wiki



(h) En-Zh NMT Wiki

Figure 4.1: Target F1-Multi scores with Few-shot learning

4.2.3 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 4.1 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, 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.

4.3 Conclusion

Conclusions

Bibliography

[1] Giannis Varelas, Epimenidis Voutsakis, Paraskevi Raftopoulou, Euripides G.M. Petrakis, and Evangelos E. Milios. Semantic similarity methods in wordnet and their application to information retrieval on the web. In *Proceedings of the 7th Annual ACM International Workshop on Web Information and Data Management*, WIDM '05, page 1016, New York, NY, USA, 2005. Association for Computing Machinery. ISBN 1595931945. doi: 10.1145/1097047.1097051. URL https://doi.org/10.1145/1097047.1097051.

[2] R. Subhashini and V. Jawahar Senthil Kumar. Evaluating the performance of similarity measures used in document clustering and information retrieval. In *2010 First International Conference on Integrated Intelligent Computing*, pages 27–31, 2010. doi: 10.1109/ICIIC.2010.42.

[3] Ramiz M. Aliguliyev. A new sentence similarity measure and sentence based extractive technique for automatic text summarization. *Expert Systems with Applications*, 36(4):7764–7772, 2009. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2008.11.022. URL https://www.sciencedirect.com/science/article/pii/S0957417408008737.

[4] Eike Schallehn, Kai-Uwe Sattler, and Gunter Saake. Efficient similarity-based operations for data integration. *Data & Knowledge Engineering*, 48(3):361–387, 2004. ISSN 0169-023X. doi: https://doi.org/10.1016/

j.datak.2003.08.004. URL https://www.sciencedirect.com/science/article/pii/S0169023X03001381.

[5] Michael Mohler, Razvan Bunescu, and Rada Mihalcea. Learning to grade short answer questions using semantic similarity measures and dependency graph alignments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 752–762, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P11-1076.

[6] Junmei Wang, Min Pan, Tingting He, Xiang Huang, Xueyan Wang, and Xinhui Tu. A pseudo-relevance feedback framework combining relevance matching and semantic matching for information retrieval. *Information Processing & Management*, 57(6):102342, 2020. ISSN 0306-4573. doi: https://doi.org/10.1016/j.ipm.2020.102342. URL https://www.sciencedirect.com/science/article/pii/S0306457320308372.

[7] Baoli Li and Liping Han. Distance weighted cosine similarity measure for text classification. In Hujun Yin, Ke Tang, Yang Gao, Frank Klawonn, Minho Lee, Thomas Weise, Bin Li, and Xin Yao, editors, *Intelligent Data Engineering and Automated Learning – IDEAL 2013*, pages 611–618, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[8] Shereen Albitar, Sébastien Fournier, and Bernard Espinasse. An effec-

tive tf/idf-based text-to-text semantic similarity measure for text classification. In Boualem Benatallah, Azer Bestavros, Yannis Manolopoulos, Athena Vakali, and Yanchun Zhang, editors, *Web Information Systems Engineering – WISE 2014*, pages 105–114, Cham, 2014. Springer International Publishing. ISBN 978-3-319-11749-2.

[9] Khaled Abdalgader and Andrew Skabar. Short-text similarity measurement using word sense disambiguation and synonym expansion. In Jiuyong Li, editor, *AI 2010: Advances in Artificial Intelligence*, pages 435–444, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. ISBN 978-3-642-17432-2.

[10] William W. Cohen. Data integration using similarity joins and a word-based information representation language. *ACM Trans. Inf. Syst.*, 18(3): 288321, July 2000. ISSN 1046-8188. doi: 10.1145/352595.352598. URL https://doi.org/10.1145/352595.352598.

[11] Xianchao Zhang, Wen Xu, and Wenxin Liang. Extracting local web communities using lexical similarity. In Masatoshi Yoshikawa, Xiaofeng Meng, Takayuki Yumoto, Qiang Ma, Lifeng Sun, and Chiemi Watanabe, editors, *Database Systems for Advanced Applications*, pages 327–337, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. ISBN 978-3-642-14589-6.

[12] Shi Feng, Le Zhang, Binyang Li, Daling Wang, Ge Yu, and Kam-Fai Wong. Is Twitter a better corpus for measuring sentiment similarity? In *Pro-*

ceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 897–902, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL https://aclanthology.org/D13-1091.

[13] George A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):3941, November 1995. ISSN 0001-0782. doi: 10.1145/219717.219748. URL https://doi.org/10.1145/219717.219748.

[14] Hanna Béchara, Hernani Costa, Shiva Taslimipoor, Rohit Gupta, Constantin Orasan, Gloria Corpas Pastor, and Ruslan Mitkov. MiniExperts: An SVM approach for measuring semantic textual similarity. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 96–101, Denver, Colorado, June 2015. Association for Computational Linguistics. doi: 10.18653/v1/S15-2017. URL https://www.aclweb.org/anthology/S15-2017.

[15] Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In Yoshua Bengio and Yann LeCun, editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013. URL http://arxiv.org/abs/1301.3781.

[16] Matt J. Kusner, Yu Sun, Nicholas I. Kolkin, and Kilian Q. Weinberger. From word embeddings to document distances. In *Proceedings of the 32nd In-*

ternational Conference on International Conference on Machine Learning - Volume 37, ICML'15, page 957966. JMLR.org, 2015.

[17] Sanjeev Arora, Yingyu Liang, and Tengyu Ma. A simple but tough-to-beat baseline for sentence embeddings. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=SyK00v5xx.

[18] Kai Sheng Tai, Richard Socher, and Christopher D. Manning. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-1150. URL https://www.aclweb.org/anthology/P15-1150.

[19] Jonas Mueller and Aditya Thyagarajan. Siamese recurrent architectures for learning sentence similarity. *Proceedings of the AAAI Conference on Artificial Intelligence*, 30(1), Mar 2016. URL https://ojs.aaai.org/index.php/AAAI/article/view/10350.

[20] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North Ameri-*

can Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://www.aclweb.org/anthology/N19-1423.

[21] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5753–5763, 2019.

[22] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692*, 2019.

[23] Eneko Agirre, Daniel Cer, Mona Diab, and Aitor Gonzalez-Agirre. SemEval-2012 task 6: A pilot on semantic textual similarity. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 385–393, Montréal, Canada, 7-8 June 2012. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/S12-1051.

[24] Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. *SEM 2013 shared task: Semantic textual similarity. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 32–43, Atlanta, Georgia, USA, June 2013. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/S13-1004.

[25] Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. SemEval-2014 task 10: Multilingual semantic textual similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 81–91, Dublin, Ireland, August 2014. Association for Computational Linguistics. doi: 10.3115/v1/S14-2010. URL https://www.aclweb.org/anthology/S14-2010.

[26] Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Iñigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, German Rigau, Larraitz Uria, and Janyce Wiebe. SemEval-2015 task 2: Semantic textual similarity, English, Spanish and pilot on interpretability. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 252–263, Denver, Colorado, June 2015. Association for Computational Linguistics. doi: 10.18653/v1/S15-2045. URL https://www.aclweb.org/anthology/S15-2045.

[27] Eneko Agirre, Carmen Banea, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Rada Mihalcea, German Rigau, and Janyce Wiebe. SemEval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 497–511, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/S16-1081. URL https://www.aclweb.org/anthology/S16-1081.

[28] Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/S17-2001. URL https://www.aclweb.org/anthology/S17-2001.

[29] Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. SemEval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 1–8, Dublin, Ireland, August 2014. Association for Computational Linguistics. doi: 10.3115/v1/S14-2001. URL https://www.aclweb.org/anthology/S14-2001.

[30] Cyrus Rashtchian, Peter Young, Micah Hodosh, and Julia Hockenmaier. Collecting image annotations using Amazon's mechanical turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 139–147, Los Angeles, June 2010. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W10-0721.

[31] Darnes Vilariño, David Pinto, Saúl León, Mireya Tovar, and Beatriz Beltrán. BUAP: Evaluating features for multilingual and cross-level semantic textual similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 149–153, Dublin, Ireland, August 2014. Association for Computational Linguistics. doi: 10.3115/v1/S14-2022. URL https://www.aclweb.org/anthology/S14-2022.

[32] Rohit Gupta, Hanna Béchara, Ismail El Maarouf, and Constantin Orăsan. UoW: NLP techniques developed at the University of Wolverhampton for semantic similarity and textual entailment. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 785–789, Dublin, Ireland, August 2014. Association for Computational Linguistics. doi: 10.3115/v1/S14-2139. URL https://www.aclweb.org/anthology/S14-2139.

[33] André Lynum, Partha Pakray, Björn Gambäck, and Sergio Jimenez. NTNU: Measuring semantic similarity with sublexical feature representations and soft cardinality. In *Proceedings of the 8th International Workshop on Seman-*

tic Evaluation (SemEval 2014), pages 448–453, Dublin, Ireland, August 2014. Association for Computational Linguistics. doi: 10.3115/v1/S14-2078. URL https://www.aclweb.org/anthology/S14-2078.

[34] Alexander Chávez, Héctor Dávila, Yoan Gutiérrez, Antonio Fernández-Orquín, Andrés Montoyo, and Rafael Muñoz. UMCC_DLSI_SemSim: Multilingual system for measuring semantic textual similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 716–721, Dublin, Ireland, August 2014. Association for Computational Linguistics. doi: 10.3115/v1/S14-2128. URL https://www.aclweb.org/anthology/S14-2128.

[35] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1075. URL https://www.aclweb.org/anthology/D15-1075.

[36] Yuxia Wang, Fei Liu, Karin Verspoor, and Timothy Baldwin. Evaluating the Utility of Model Configurations and Data Augmentation on Clinical Semantic Textual Similarity. In *Proceedings of the 19th SIGBioMed Workshop on Biomedical Language Processing*, pages 105–111, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.bionlp-1.11. URL https://www.aclweb.org/anthology/2020.bionlp-1.11.

[37] Junyi Li, Xuejie Zhang, and Xiaobing Zhou. ALBERT-Based Self-Ensemble Model With Semisupervised Learning and Data Augmentation for Clinical Semantic Textual Similarity Calculation: Algorithm Validation Study. *JMIR Med Inform*, 9(1):e23086, Jan 2021. ISSN 2291-9694. doi: 10.2196/23086. URL http://medinform.jmir.org/2021/1/e23086/.

[38] Z. Imtiaz, M. Umer, M. Ahmad, S. Ullah, G. S. Choi, and A. Mehmood. Duplicate questions pair detection using siamese malstm. *IEEE Access*, 8: 21932–21942, 2020. doi: 10.1109/ACCESS.2020.2969041.

[39] Gizem Soancolu, Hakime Öztürk, and Arzucan Özgür. BIOSSES: a semantic sentence similarity estimation system for the biomedical domain. *Bioinformatics*, 33(14):i49–i58, 07 2017. ISSN 1367-4803. doi: 10.1093/bioinformatics/btx238. URL https://doi.org/10.1093/bioinformatics/btx238.

[40] Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. Unsupervised learning of sentence embeddings using compositional n-gram features. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 528–540, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1049. URL https://www.aclweb.org/anthology/N18-1049.

[41] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and An-

toine Bordes. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D17-1070.

[42] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, page II1188II1196. JMLR.org, 2014.

[43] Tharindu Ranasinghe, Constantin Orasan, and Ruslan Mitkov. 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)*, pages 994–1003, Varna, Bulgaria, September 2019. INCOMA Ltd. doi: 10.26615/978-954-452-056-4_115. URL https://www.aclweb.org/anthology/R19-1115.

[44] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014. URL http://www.aclweb.org/anthology/D14-1162.

[45] Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and

Armand Joulin. Advances in pre-training distributed word representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May 2018. European Language Resources Association (ELRA). URL https://www.aclweb.org/anthology/L18-1008.

[46] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, page 31113119, Red Hook, NY, USA, 2013. Curran Associates Inc.

[47] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1202. URL https://www.aclweb.org/anthology/N18-1202.

[48] Ye Jiang, Johann Petrak, Xingyi Song, Kalina Bontcheva, and Diana Maynard. Team bertha von suttner at SemEval-2019 task 4: Hyperpartisan news detection using ELMo sentence representation convolutional network. In *Proceedings of the 13th International Workshop on Semantic Eval-*

uation, pages 840–844, Minneapolis, Minnesota, USA, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/S19-2146. URL https://www.aclweb.org/anthology/S19-2146.

[49] Ling Luo, Nan Li, Shuaichi Li, Zhihao Yang, and Hongfei Lin. DUTIR at the CCKS-2018 Task1: A neural network ensemble approach for Chinese clinical named entity recognition. In *CEUR Workshop Proceedings*, volume 2242, pages 7–12. CEUR-WS, 2018.

[50] Wanxiang Che, Yijia Liu, Yuxuan Wang, Bo Zheng, and Ting Liu. Towards better UD parsing: Deep contextualized word embeddings, ensemble, and treebank concatenation. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 55–64, Brussels, Belgium, October 2018. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/K18-2005.

[51] Qiao Jin, Bhuwan Dhingra, William Cohen, and Xinghua Lu. Probing biomedical embeddings from language models. In *Proceedings of the 3rd Workshop on Evaluating Vector Space Representations for NLP*, pages 82–89, 2019.

[52] Tharindu Ranasinghe, Marcos Zampieri, and Hansi Hettiarachchi. BRUMS at HASOC 2019: Deep learning models for multilingual hate speech and offensive language identification. In *CEUR Workshop Proceedings*, volume

2517, pages 199–207, 2019. URL http://ceur-ws.org/Vol-2517/T3-3.pdf.

[53] Hansi Hettiarachchi and Tharindu Ranasinghe. Brums at semeval-2020 task 3: Contextualised embeddings for predicting the (graded) effect of context in word similarity. In *Proceedings of the 14th International Workshop on Semantic Evaluation*, Barcelona, Spain, 2020. Association for Computational Linguistics.

[54] Chen Liang, Yue Yu, Haoming Jiang, Siawpeng Er, Ruijia Wang, Tuo Zhao, and Chao Zhang. Bond: Bert-assisted open-domain named entity recognition with distant supervision. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '20, page 10541064, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450379984. doi: 10.1145/3394486.3403149. URL https://doi.org/10.1145/3394486.3403149.

[55] Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. End-to-end open-domain question answering with BERTserini. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 72–77, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-4013. URL https://www.aclweb.org/anthology/N19-4013.

[56] Wissam Antoun, Fady Baly, and Hazem Hajj. AraBERT: Transformer-based model for Arabic language understanding. In *Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection*, pages 9–15, Marseille, France, May 2020. European Language Resource Association. ISBN 979-10-95546-51-1. URL https://www.aclweb.org/anthology/2020.osact-1.2.

[57] Louis Martin, Benjamin Muller, Pedro Javier Ortiz Suárez, Yoann Dupont, Laurent Romary, Éric de la Clergerie, Djamé Seddah, and Benoît Sagot. CamemBERT: a tasty French language model. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7203–7219, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.645. URL https://www.aclweb.org/anthology/2020.acl-main.645.

[58] José Cañete, Gabriel Chaperon, Rodrigo Fuentes, Jou-Hui Ho, Hojin Kang, and Jorge Pérez. Spanish Pre-Trained BERT Model and Evaluation Data. In *PML4DC at ICLR 2020*, 2020.

[59] John Koutsikakis, Ilias Chalkidis, Prodromos Malakasiotis, and Ion Androutsopoulos. GREEK-BERT: The Greeks Visiting Sesame Street. In *11th Hellenic Conference on Artificial Intelligence*, SETN 2020, page 110117, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450388788. doi: 10.1145/3411408.3411440. URL https://doi.org/10.1145/3411408.3411440.

[60] Iz Beltagy, Kyle Lo, and Arman Cohan. SciBERT: A pretrained language model for scientific text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3615–3620, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1371. URL https://www.aclweb.org/anthology/D19-1371.

[61] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 09 2019. ISSN 1367-4803. doi: 10.1093/bioinformatics/btz682. URL https://doi.org/10.1093/bioinformatics/btz682.

[62] Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, Nikolaos Aletras, and Ion Androutsopoulos. LEGAL-BERT: The muppets straight out of law school. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2898–2904, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.261. URL https://www.aclweb.org/anthology/2020.findings-emnlp.261.

[63] Alan Akbik, Duncan Blythe, and Roland Vollgraf. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, Santa Fe, New

Mexico, USA, August 2018. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/C18-1139.

[64] Alan Akbik, Tanja Bergmann, and Roland Vollgraf. Pooled contextualized embeddings for named entity recognition. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 724–728, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1078. URL https://www.aclweb.org/anthology/N19-1078.

[65] Shreyas Sharma and Ron Daniel Jr. Bioflair: Pretrained pooled contextualized embeddings for biomedical sequence labeling tasks. *arXiv preprint arXiv:1908.05760*, 2019.

[66] Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. FLAIR: An easy-to-use framework for state-of-the-art NLP. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-4010. URL https://www.aclweb.org/anthology/N19-4010.

[67] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Con-*

ference on Empirical Methods in Natural Language Processing (EMNLP), pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1162. URL https://www.aclweb.org/anthology/D14-1162.

[68] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, page 604613, New York, NY, USA, 1998. Association for Computing Machinery. ISBN 0897919629. doi: 10.1145/276698.276876. URL https://doi.org/10.1145/276698.276876.

[69] Murhaf Fares, Andrey Kutuzov, Stephan Oepen, and Erik Velldal. Word vectors, reuse, and replicability: Towards a community repository of large-text resources. In *Proceedings of the 21st Nordic Conference on Computational Linguistics*, pages 271–276, Gothenburg, Sweden, May 2017. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W17-0237.

[70] Abu Bakr Soliman, Kareem Eissa, and Samhaa R. El-Beltagy. Aravec: A set of arabic word embedding models for use in arabic nlp. *Procedia Computer Science*, 117:256–265, 2017. ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.2017.10.117. URL https://www.sciencedirect.com/science/article/pii/S1877050917321749.

[71] Aritz Bilbao-Jayo and Aitor Almeida. Automatic political discourse analysis with multi-scale convolutional neural networks and contextual data. *International Journal of Distributed Sensor Networks*, 14(11): 1550147718811827, 2018. doi: 10.1177/1550147718811827. URL https://doi.org/10.1177/1550147718811827.

[72] Hao Wu, Heyan Huang, Ping Jian, Yuhang Guo, and Chao Su. BIT at SemEval-2017 task 1: Using semantic information space to evaluate semantic textual similarity. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 77–84, Vancouver, Canada, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/S17-2007. URL https://www.aclweb.org/anthology/S17-2007.

[73] Sarah Kohail, Amr Rekaby Salama, and Chris Biemann. STS-UHH at SemEval-2017 task 1: Scoring semantic textual similarity using supervised and unsupervised ensemble. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 175–179, Vancouver, Canada, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/S17-2025. URL https://www.aclweb.org/anthology/S17-2025.

[74] Junfeng Tian, Zhiheng Zhou, Man Lan, and Yuanbin Wu. ECNU at SemEval-2017 task 1: Leverage kernel-based traditional NLP features and neural networks to build a universal model for multilingual and cross-lingual semantic textual similarity. In *Proceedings of the 11th International*

Workshop on Semantic Evaluation (SemEval-2017), pages 191–197, Vancouver, Canada, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/S17-2028. URL https://www.aclweb.org/anthology/S17-2028.

[75] Joe Barrow and Denis Peskov. UMDeep at SemEval-2017 task 1: End-to-end shared weight LSTM model for semantic textual similarity. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 180–184, Vancouver, Canada, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/S17-2026. URL https://www.aclweb.org/anthology/S17-2026.

[76] Yijia Zhang, Qingyu Chen, Zhihao Yang, Hongfei Lin, and Zhiyong Lu. Biowordvec,ăimproving biomedical word embeddings with subword information and mesh. *Scientific Data*, 6(1):52, May 2019. doi: 10.1038/s41597-019-0055-0. URL https://doi.org/10.1038/s41597-019-0055-0.

[77] A. R. Aronson. Effective mapping of biomedical text to the umls metathesaurus: the metamap program. *Proceedings. AMIA Symposium*, pages 17–21, 2001. ISSN 1531-605X. URL https://pubmed.ncbi.nlm.nih.gov/11825149. 11825149[pmid].

[78] Divyanshu Kakwani, Anoop Kunchukuttan, Satish Golla, Gokul N.C., Avik Bhattacharyya, Mitesh M. Khapra, and Pratyush Kumar. IndicNLPSuite:

Monolingual corpora, evaluation benchmarks and pre-trained multilingual language models for Indian languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4948–4961, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.445. URL https://www.aclweb.org/anthology/2020.findings-emnlp.445.

[79] Avihay Chriqui and Inbal Yahav. Hebert & hebemo: a hebrew bert model and a tool for polarity analysis and emotion recognition. *arXiv preprint arXiv:2102.01909*, 2021.

[80] Jesujoba Alabi, Kwabena Amponsah-Kaakyire, David Adelani, and Cristina España-Bonet. Massive vs. curated embeddings for low-resourced languages: the case of Yorùbá and Twi. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 2754–2762, Marseille, France, May 2020. European Language Resources Association. ISBN 979-10-95546-34-4. URL https://www.aclweb.org/anthology/2020.lrec-1.335.

[81] Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. SpanBERT: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77, 2020. doi: 10.1162/tacl_a_00300. URL https://www.aclweb.org/anthology/2020.tacl-1.5.

[82] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-

document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

[83] Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences. *arXiv preprint arXiv:2007.14062*, 2021.

[84] Daniel Cer, Yinfei Yang, Sheng yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. Universal sentence encoder, 2018.

[85] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In Eric P. Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1188–1196, Bejing, China, 22–24 Jun 2014. PMLR. URL http://proceedings.mlr.press/v32/le14.html.

[86] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 11 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

[87] Junyoung Chung, Çaglar Gülçehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning*, Dec 2014.

[88] M. Schuster and K.K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997. doi: 10.1109/78.650093.

[89] Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov. XNLI: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2475–2485, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1269. URL https://aclanthology.org/D18-1269.

[90] Yinfei Yang, Daniel Cer, Amin Ahmad, Mandy Guo, Jax Law, Noah Constant, Gustavo Hernandez Abrego, Steve Yuan, Chris Tar, Yun-hsuan Sung, Brian Strope, and Ray Kurzweil. Multilingual universal sentence encoder for semantic retrieval. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 87–94, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-demos.12. URL https://aclanthology.org/2020.acl-demos.12.

[91] Noha S. Tawfik and Marco R. Spruit. Evaluating sentence representations for biomedical text: Methods and experimental results. *Journal of Biomedical Informatics*, 104:103396, 2020. ISSN 1532-0464. doi: https:

//doi.org/10.1016/j.jbi.2020.103396. URL https://www.sciencedirect.com/science/article/pii/S1532046420300253.

[92] Qingyu Chen, Yifan Peng, and Zhiyong Lu. Biosentvec: creating sentence embeddings for biomedical texts. In *2019 IEEE International Conference on Healthcare Informatics (ICHI)*, pages 1–5, 2019. doi: 10.1109/ICHI.2019.8904728.

[93] Mikel Artetxe and Holger Schwenk. Massively Multilingual Sentence Embeddings for Zero-Shot Cross-Lingual Transfer and Beyond. *Transactions of the Association for Computational Linguistics*, 7:597–610, 09 2019. ISSN 2307-387X. doi: 10.1162/tacl_a_00288. URL https://doi.org/10.1162/tacl_a_00288.

[94] Roland Thiolliere, Ewan Dunbar, Gabriel Synnaeve, Maarten Versteegh, and Emmanuel Dupoux. A hybrid dynamic time warping-deep neural network architecture for unsupervised acoustic modeling. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[95] Pranay Manocha, Rohan Badlani, Anurag Kumar, Ankit Shah, Benjamin Elizalde, and Bhiksha Raj. Content-based representations of audio using siamese neural networks. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3136–3140, 2018. doi: 10.1109/ICASSP.2018.8461524.

[96] Suwon Shon, Ahmed Ali, and James Glass. Mit-qcri arabic dialect identi-

fication system for the 2017 multi-genre broadcast challenge. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 374–380, 2017. doi: 10.1109/ASRU.2017.8268960.

[97] Yichi Zhang, Bryan Pardo, and Zhiyao Duan. Siamese style convolutional neural networks for sound search by vocal imitation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(2):429–441, 2019. doi: 10.1109/TASLP.2018.2868428.

[98] Jan Švec, Luboš Šmídl, and Josef V. Psutka. An analysis of the rnn-based spokenătermădetection training. In Alexey Karpov, Rodmonga Potapova, and Iosif Mporas, editors, *Speech and Computer*, pages 119–129, Cham, 2017. Springer International Publishing.

[99] Batuhan Gündodu, Bolaji Yusuf, and Murat Saraçlar. Joint learning of distance metric and query model for posteriorgram-based keyword search. *IEEE Journal of Selected Topics in Signal Processing*, 11(8):1318–1328, 2017. doi: 10.1109/JSTSP.2017.2762080.

[100] Aditya Siddhant, Preethi Jyothi, and Sriram Ganapathy. Leveraging native language speech for accent identification using deep siamese networks. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 621–628, 2017. doi: 10.1109/ASRU.2017.8268994.

[101] Neil Zeghidour, Gabriel Synnaeve, Nicolas Usunier, and Emmanuel Dupoux. Joint learning of speaker and phonetic similarities with

siamese networks. In *Interspeech 2016*, pages 1295–1299, 2016. doi: 10.21437/Interspeech.2016-811. URL http://dx.doi.org/10.21437/Interspeech.2016-811.

[102] Wei Zheng, Le Yang, Robert J Genco, Jean Wactawski-Wende, Michael Buck, and Yijun Sun. SENSE: Siamese neural network for sequence embedding and alignment-free comparison. *Bioinformatics*, 35(11):1820–1828, 10 2018. ISSN 1367-4803. doi: 10.1093/bioinformatics/bty887. URL https://doi.org/10.1093/bioinformatics/bty887.

[103] Benjamin Szubert, Jennifer E. Cole, Claudia Monaco, and Ignat Drozdov. Structure-preserving visualisation of high dimensional single-cell datasets. *Scientific Reports*, 9(1):8914, Jun 2019. ISSN 2045-2322. doi: 10.1038/s41598-019-45301-0. URL https://doi.org/10.1038/s41598-019-45301-0.

[104] Minji Jeon, Donghyeon Park, Jinhyuk Lee, Hwisang Jeon, Miyoung Ko, Sunkyu Kim, Yonghwa Choi, Aik-Choon Tan, and Jaewoo Kang. ReSimNet: drug response similarity prediction using Siamese neural networks. *Bioinformatics*, 35(24):5249–5256, 05 2019. ISSN 1367-4803. doi: 10.1093/bioinformatics/btz411. URL https://doi.org/10.1093/bioinformatics/btz411.

[105] Zhiyu Sun, Yusen He, Andrey Gritsenko, Amaury Lendasse, and Stephen Baek. Embedded spectral descriptors: learning the point-wise corre-

spondence metric via Siamese neural networks. *Journal of Computational Design and Engineering*, 7(1):18–29, 03 2020. ISSN 2288-5048. doi: 10.1093/jcde/qwaa003. URL https://doi.org/10.1093/jcde/qwaa003.

[106] Pierre Baldi and Yves Chauvin. Neural networks for fingerprint recognition. *Neural Computation*, 5(3):402–418, 1993. doi: 10.1162/neco.1993.5.3.402.

[107] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 539–546 vol. 1, 2005. doi: 10.1109/CVPR.2005.202.

[108] Haiqing He, Min Chen, Ting Chen, and Dajun Li. Matching of remote sensing images with complex background variations via siamese convolutional neural network. *Remote Sensing*, 10(2), 2018. ISSN 2072-4292. doi: 10.3390/rs10020355. URL https://www.mdpi.com/2072-4292/10/2/355.

[109] Nektarios Paisios, Alex Rubinsteyn, and Lakshminarayanan Subramanian. Choosing which clothes to wear confidently: A tool for pattern matching. In *Frontiers of Accessibility for Pervasive Computing*, June 2012.

[110] Dong Yi, Zhen Lei, Shengcai Liao, and Stan Z. Li. Deep metric learning for person re-identification. In *2014 22nd International Conference on Pattern Recognition*, pages 34–39, 2014. doi: 10.1109/ICPR.2014.16.

[111] Grégoire Lefebvre and Christophe Garcia. Learning a bag of features based

nonlinear metric for facial similarity. In *2013 10th IEEE International Conference on Advanced Video and Signal Based Surveillance*, pages 238–243, 2013. doi: 10.1109/AVSS.2013.6636646.

[112] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[113] Samuel Berlemont, Grégoire Lefebvre, Stefan Duffner, and Christophe Garcia. Siamese neural network based similarity metric for inertial gesture classification and rejection. In *2015 11th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*, volume 1, pages 1–6, 2015. doi: 10.1109/FG.2015.7163112.

[114] Majeed Kassis, Jumana Nassour, and Jihad El-Sana. Alignment of historical handwritten manuscripts using siamese neural network. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 01, pages 293–298, 2017. doi: 10.1109/ICDAR.2017.56.

[115] Muhammad Shehzad Hanif. Patch match networks: Improved two-channel and siamese networks for image patch matching. *Pattern Recognition Letters*, 120:54–61, 2019. ISSN 0167-8655. doi: https://doi.org/10.1016/j.patrec.2019.01.005. URL https://www.sciencedirect.com/science/article/pii/S0167865519300054.

[116] Yanbiao Zou, Jinchao Li, Xiangzhi Chen, and Rui Lan. Learning siamese networks for laser vision seam tracking. *J. Opt. Soc. Am. A*, 35(11):1805–1813, Nov 2018. doi: 10.1364/JOSAA.35.001805. URL http://josaa.osa.org/abstract.cfm?URI=josaa-35-11-1805.

[117] Leen De Baets, Chris Develder, Tom Dhaene, and Dirk Deschrijver. Detection of unidentified appliances in non-intrusive load monitoring using siamese neural networks. *International Journal of Electrical Power & Energy Systems*, 104:645–653, 2019. ISSN 0142-0615. doi: https://doi.org/10.1016/j.ijepes.2018.07.026. URL https://www.sciencedirect.com/science/article/pii/S0142061517331253.

[118] L. V. Utkin, V. S. Zaborovsky, and S. G. Popov. Siamese neural network for intelligent information security control in multi-robot systems. *Automatic Control and Computer Sciences*, 51(8):881–887, Dec 2017. ISSN 1558-108X. doi: 10.3103/S0146411617080235. URL https://doi.org/10.3103/S0146411617080235.

[119] L. V. Utkin, Yu. A. Zhuk, and V. S. Zaborovsky. An anomalous behavior detection of a robot system by using a hierarchical siamese neural network. In *2017 XX IEEE International Conference on Soft Computing and Measurements (SCM)*, pages 630–634, 2017. doi: 10.1109/SCM.2017.7970671.

[120] Andy Zeng, Shuran Song, Kuan-Ting Yu, Elliott Donlon, Francois R. Hogan, Maria Bauza, Daolin Ma, Orion Taylor, Melody Liu, Eudald Romo,

Nima Fazeli, Ferran Alet, Nikhil Chavan Dafle, Rachel Holladay, Isabella Morena, Prem Qu Nair, Druck Green, Ian Taylor, Weber Liu, Thomas Funkhouser, and Alberto Rodriguez. Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3750–3757, 2018. doi: 10.1109/ICRA.2018.8461044.

[121] Michael Ryoo, Kiyoon Kim, and Hyun Yang. Extreme low resolution activity recognition with multi-siamese embedding learning. In *AAAI Conference on Artificial Intelligence*, 2018. URL https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16790.

[122] Xinchen Liu, Wu Liu, Tao Mei, and Huadong Ma. Provid: Progressive and multimodal vehicle reidentification for large-scale urban surveillance. *IEEE Transactions on Multimedia*, 20(3):645–658, 2018. doi: 10.1109/TMM.2017.2751966.

[123] Wu Liu, Cheng Zhang, Huadong Ma, and Shuangqun Li. Learning efficient spatial-temporal gait features with deep learning for human identification. *Neuroinformatics*, 16(3):457–471, Oct 2018. ISSN 1559-0089. doi: 10.1007/s12021-018-9362-4. URL https://doi.org/10.1007/s12021-018-9362-4.

[124] Sangyun Lee and Euntai Kim. Multiple object tracking via feature pyramid

siamese networks. *IEEE Access*, 7:8181–8194, 2019. doi: 10.1109/ACCESS.2018.2889442.

[125] Wen-tau Yih, Kristina Toutanova, John C. Platt, and Christopher Meek. Learning discriminative projections for text similarity measures. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, CoNLL '11, page 247256, USA, 2011. Association for Computational Linguistics. ISBN 9781932432923.

[126] Vaibhav Kumar, Dhruv Khattar, Siddhartha Gairola, Yash Kumar Lal, and Vasudeva Varma. Identifying clickbait: A multi-strategy approach using neural networks. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '18, page 12251228, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450356572. doi: 10.1145/3209978.3210144. URL https://doi.org/10.1145/3209978.3210144.

[127] José-Ángel González, Encarna Segarra, Fernando García-Granada, Emilio Sanchis, and Llu’ıs-F. Hurtado. Siamese hierarchical attention networks for extractive summarization. *Journal of Intelligent & Fuzzy Systems*, 36:4599–4607, 2019. ISSN 1875-8967. doi: 10.3233/JIFS-179011. URL https://doi.org/10.3233/JIFS-179011. 5.

[128] Arpita Das, Harish Yenala, Manoj Chinnakotla, and Manish Shrivastava. Together we stand: Siamese networks for similar question retrieval. In

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 378–387, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1036. URL https://aclanthology.org/P16-1036.

[129] Paul Neculoiu, Maarten Versteegh, and Mihai Rotaru. Learning text similarity with Siamese recurrent networks. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 148–157, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/W16-1617. URL https://aclanthology.org/W16-1617.

[130] Tharindu Ranasinghe, Constantin Orasan, and Ruslan Mitkov. Semantic textual similarity with Siamese neural networks. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pages 1004–1011, Varna, Bulgaria, September 2019. INCOMA Ltd. doi: 10.26615/978-954-452-056-4_116. URL https://www.aclweb.org/anthology/R19-1116.

[131] Jane Bromley, James W Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. Signature verification using a siamese time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, 07(04):669–688, 1993. doi: 10.1142/S0218001493000339. URL https://doi.org/10.1142/S0218001493000339.

[132] Vedran Vukotić, Christian Raymond, and Guillaume Gravier. A step beyond local observations with a dialog aware bidirectional GRU network for Spoken Language Understanding. In *Interspeech*, San Francisco, United States, September 2016. URL https://hal.inria.fr/hal-01351733.

[133] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[134] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/2cad8fa47bbef282badbb8de5374b894-Paper.pdf.

[135] Hansi Hettiarachchi and Tharindu Ranasinghe. Emoji powered capsule network to detect type and target of offensive posts in social media. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pages 474–480, Varna, Bulgaria,

September 2019. INCOMA Ltd. doi: 10.26615/978-954-452-056-4_056. URL https://aclanthology.org/R19-1056.

[136] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR, 09–15 Jun 2019. URL http://proceedings.mlr.press/v97/houlsby19a.html.

[137] Sebastian Ruder, Matthew E. Peters, Swabha Swayamdipta, and Thomas Wolf. Transfer learning in natural language processing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials*, pages 15–18, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-5004. URL https://aclanthology.org/N19-5004.

[138] Jason Wei and Kai Zou. EDA: Easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6382–6388, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1670. URL https://aclanthology.org/D19-1670.

[139] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, page 649657, Cambridge, MA, USA, 2015. MIT Press.

[140] Sergio Jimenez, George Dueñas, Julia Baquero, and Alexander Gelbukh. UNAL-NLP: Combining soft cardinality features for semantic textual similarity, relatedness and entailment. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 732–742, Dublin, Ireland, August 2014. Association for Computational Linguistics. doi: 10.3115/v1/S14-2131. URL https://aclanthology.org/S14-2131.

[141] Johannes Bjerva, Johan Bos, Rob van der Goot, and Malvina Nissim. The meaning factory: Formal semantics for recognizing textual entailment and determining semantic similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 642–646, Dublin, Ireland, August 2014. Association for Computational Linguistics. doi: 10.3115/v1/S14-2114. URL https://aclanthology.org/S14-2114.

[142] Jiang Zhao, Tiantian Zhu, and Man Lan. ECNU: One stone two birds: Ensemble of heterogenous measures for semantic relatedness and textual entailment. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 271–277, Dublin, Ireland, August 2014. Association for Computational Linguistics. doi: 10.3115/v1/S14-2044. URL https://aclanthology.org/S14-2044.

[143] Nabin Maharjan, Rajendra Banjade, Dipesh Gautam, Lasang J. Tamang, and Vasile Rus. DT_Team at SemEval-2017 task 1: Semantic similarity using alignments, sentence-level embeddings and Gaussian mixture model output. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 120–124, Vancouver, Canada, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/S17-2014. URL https://aclanthology.org/S17-2014.

[144] El Moatez Billah Nagoudi, Jérémy Ferrero, and Didier Schwab. LIM-LIG at SemEval-2017 task1: Enhancing the semantic similarity for Arabic sentences with vectors weighting. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 134–138, Vancouver, Canada, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/S17-2017. URL https://aclanthology.org/S17-2017.

[145] Basma Hassan, Samir AbdelRahman, Reem Bahgat, and Ibrahim Farag. FCICU at SemEval-2017 task 1: Sense-based language independent semantic textual similarity approach. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 125–129, Vancouver, Canada, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/S17-2015. URL https://aclanthology.org/S17-2015.

[146] Varun Kumar, Ashutosh Choudhary, and Eunah Cho. Data augmentation using pre-trained transformer models. In *Proceedings of the 2nd Workshop on Life-long Learning for Spoken Language Systems*, pages 18–26, Suzhou,

China, December 2020. Association for Computational Linguistics. URL https://aclanthology.org/2020.lifelongnlp-1.3.

[147] Samar Haider. Urdu word embeddings. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May 2018. European Language Resources Association (ELRA). URL https://aclanthology.org/L18-1155.

[148] Saurav Kumar, Saunack Kumar, Diptesh Kanojia, and Pushpak Bhattacharyya. "a passage to India": Pre-trained word embeddings for Indian languages. In *Proceedings of the 1st Joint Workshop on Spoken Language Technologies for Under-resourced languages (SLTU) and Collaboration and Computing for Under-Resourced Languages (CCURL)*, pages 352–357, Marseille, France, May 2020. European Language Resources association. ISBN 979-10-95546-35-1. URL https://aclanthology.org/2020.sltu-1.49.

[149] Ilias Chalkidis and Dimitrios Kampas. Deep learning in law: early adaptation and legal word embeddings trained on large corpora. *Artificial Intelligence and Law*, 27(2):171–198, Jun 2019. ISSN 1572-8382. doi: 10.1007/s10506-018-9238-9. URL https://doi.org/10.1007/s10506-018-9238-9.

[150] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th*

International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 3982–3992, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1410. URL https://aclanthology.org/D19-1410.

[151] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1eA7AEtvS.

[152] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. Electra: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=r1xMH1BtvB.

[153] Benigno Uria, Marc-Alexandre Côté, Karol Gregor, Iain Murray, and Hugo Larochelle. Neural autoregressive distribution estimation. *Journal of Machine Learning Research*, 17(205):1–37, 2016. URL http://jmlr.org/papers/v17/16-272.html.

[154] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An im-

perative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[155] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-demos.6. URL https://aclanthology.org/2020.emnlp-demos.6.

[156] Peter J. Arthern. Machine translation and computerized terminology systems: A translators viewpoint. *Translating and the Computer, Proceedings of a Seminar, London 14th November 1978. Amsterdam: North-Holland Publishing Company*, pages 77–108, 1979.

[157] Tharindu Ranasinghe, Constantin Orasan, and Ruslan Mitkov. Intelligent translation memory matching and retrieval with sentence encoders. In *Proceedings of the 22nd Annual Conference of the European Associa-*

tion for Machine Translation, pages 175–184, Lisboa, Portugal, November 2020. European Association for Machine Translation. URL https://www.aclweb.org/anthology/2020.eamt-1.19.

[158] Lucia Specia, Frédéric Blain, Varvara Logacheva, Ramón Astudillo, and André F. T. Martins. Findings of the WMT 2018 shared task on quality estimation. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 689–709, Belgium, Brussels, October 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-6451. URL https://www.aclweb.org/anthology/W18-6451.

[159] Erick Fonseca, Lisa Yankovskaya, André F. T. Martins, Mark Fishel, and Christian Federmann. Findings of the WMT 2019 shared tasks on quality estimation. In *Proceedings of the Fourth Conference on Machine Translation (Volume 3: Shared Task Papers, Day 2)*, pages 1–10, Florence, Italy, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-5401. URL https://www.aclweb.org/anthology/W19-5401.

[160] Lucia Specia, Frédéric Blain, Marina Fomicheva, Erick Fonseca, Vishrav Chaudhary, Francisco Guzmán, and André F. T. Martins. Findings of the WMT 2020 shared task on quality estimation. In *Proceedings of the Fifth Conference on Machine Translation*, pages 743–764, Online, November 2020. Association for Computational Linguistics. URL https://aclanthology.org/2020.wmt-1.79.

[161] Yvette Graham, Timothy Baldwin, Meghan Dowling, Maria Eskevich, Teresa Lynn, and Lamia Tounsi. Is all that glitters in machine translation quality estimation really gold? In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3124–3134, Osaka, Japan, December 2016. The COLING 2016 Organizing Committee. URL https://www.aclweb.org/anthology/C16-1294.

[162] Yvette Graham, Timothu Baldwin, Alistair Moffat, and Justin Zobel. Can machine translation systems be evaluated by the crowd alone. *Natural Language Engineering*, 23(1):330, 2017. doi: 10.1017/S1351324915000339.

[163] Yvette Graham, Timothy Baldwin, and Nitika Mathur. Accurate evaluation of segment-level machine translation metrics. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1183–1191, Denver, Colorado, May–June 2015. Association for Computational Linguistics. doi: 10.3115/v1/N15-1124. URL https://www.aclweb.org/anthology/N15-1124.

[164] Marina Fomicheva, Shuo Sun, Lisa Yankovskaya, Frédéric Blain, Francisco Guzmán, Mark Fishel, Nikolaos Aletras, Vishrav Chaudhary, and Lucia Specia. Unsupervised quality estimation for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:539–

555, 2020. doi: 10.1162/tacl_a_00330. URL https://aclanthology.org/2020.tacl-1.35.

[165] Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-4009. URL https://www.aclweb.org/anthology/N19-4009.

[166] Fabio Kepler, Jonay Trénous, Marcos Treviso, Miguel Vera, and André F. T. Martins. OpenKiwi: An open source framework for quality estimation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 117–122, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-3020. URL https://www.aclweb.org/anthology/P19-3020.

[167] Hyun Kim, Jong-Hyeok Lee, and Seung-Hoon Na. Predictor-estimator using multilevel task learning with stack propagation for neural quality estimation. In *Proceedings of the Second Conference on Machine Translation*, pages 562–568, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-4763. URL https://aclanthology.org/W17-4763.

[168] Julia Ive, Frédéric Blain, and Lucia Specia. deepQuest: A framework for neural-based quality estimation. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3146–3157, Santa Fe, New Mexico, USA, August 2018. Association for Computational Linguistics. URL https://aclanthology.org/C18-1266.

[169] Dongjun Lee. Two-phase cross-lingual language model fine-tuning for machine translation quality estimation. In *Proceedings of the Fifth Conference on Machine Translation*, pages 1024–1028, Online, November 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.wmt-1.118.

[170] Jiayi Wang, Kai Fan, Bo Li, Fengming Zhou, Boxing Chen, Yangbin Shi, and Luo Si. Alibaba submission for WMT18 quality estimation task. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 809–815, Belgium, Brussels, October 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-6465. URL https://aclanthology.org/W18-6465.

[171] Tharindu Ranasinghe, Constantin Orasan, and Ruslan Mitkov. TransQuest: Translation quality estimation with cross-lingual transformers. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5070–5081, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics. doi:

10.18653/v1/2020.coling-main.445. URL https://aclanthology.org/2020.coling-main.445.

[172] Tharindu Ranasinghe, Constantin Orasan, and Ruslan Mitkov. TransQuest at WMT2020: Sentence-level direct assessment. In *Proceedings of the Fifth Conference on Machine Translation*, pages 1049–1055, Online, November 2020. Association for Computational Linguistics. URL https://aclanthology.org/2020.wmt-1.122.

[173] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. In *Proceedings of the fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing*, 2019.

[174] Telmo Pires, Eva Schlinger, and Dan Garrette. How multilingual is multilingual BERT? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4996–5001, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1493. URL https://www.aclweb.org/anthology/P19-1493.

[175] Karthikeyan K, Zihan Wang, Stephen Mayhew, and Dan Roth. Cross-lingual ability of multilingual bert: An empirical study. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HJeT3yrtDr.

[176] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaud-

hary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.747. URL https://www.aclweb.org/anthology/2020.acl-main.747.

[177] Alexis Conneau and Guillaume Lample. Cross-lingual language model pretraining. In *Advances in Neural Information Processing Systems 32*, pages 7059–7069. Curran Associates, Inc., 2019. URL http://papers.nips.cc/paper/8928-cross-lingual-language-model-pretraining.pdf.

[178] Marina Fomicheva, Shuo Sun, Lisa Yankovskaya, Frédéric Blain, Vishrav Chaudhary, Mark Fishel, Francisco Guzmán, and Lucia Specia. BERGAMOT-LATTE submissions for the WMT20 quality estimation shared task. In *Proceedings of the Fifth Conference on Machine Translation*, pages 1010–1017, Online, November 2020. Association for Computational Linguistics. URL https://aclanthology.org/2020.wmt-1.116.

[179] Milan Gritta and Ignacio Iacobacci. XeroAlign: Zero-shot cross-lingual transformer alignment. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 371–381, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-acl.32. URL https://aclanthology.org/2021.findings-acl.32.

[180] Naman Goyal, Jingfei Du, Myle Ott, Giri Anantharaman, and Alexis Conneau. Larger-scale transformers for multilingual masked language modeling. *arXiv preprint arXiv:2105.00572*, 2021.

[181] Varvara Logacheva, Chris Hokamp, and Lucia Specia. MARMOT: A toolkit for translation quality estimation at the word level. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 3671–3674, Portorož, Slovenia, May 2016. European Language Resources Association (ELRA). URL https://www.aclweb.org/anthology/L16-1582.

[182] Tharindu Ranasinghe, Constantin Orasan, and Ruslan Mitkov. 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)*, pages 434–440, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-short.55. URL https://aclanthology.org/2021.acl-short.55.

[183] Rich Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, Jul 1997. ISSN 1573-0565. doi: 10.1023/A:1007379606734. URL https://doi.org/10.1023/A:1007379606734.

[184] Tharindu Ranasinghe and Marcos Zampieri. Multilingual offensive lan-

guage identification with cross-lingual embeddings. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5838–5844, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.470. URL https://aclanthology.org/2020.emnlp-main.470.

[185] Tharindu Ranasinghe and Marcos Zampieri. MUDES: Multilingual detection of offensive spans. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Demonstrations*, pages 144–152, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-demos.17. URL https://aclanthology.org/2021.naacl-demos.17.

[186] Toan Q. Nguyen and David Chiang. Transfer learning across low-resource, related languages for neural machine translation. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 296–301, Taipei, Taiwan, November 2017. Asian Federation of Natural Language Processing. URL https://www.aclweb.org/anthology/I17-2050.

[187] Roee Aharoni, Melvin Johnson, and Orhan Firat. Massively multilingual neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Pa-*

pers), pages 3874–3884, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1388. URL https://www.aclweb.org/anthology/N19-1388.

[188] Shuo Sun, Marina Fomicheva, Frédéric Blain, Vishrav Chaudhary, Ahmed El-Kishky, Adithya Renduchintala, Francisco Guzmán, and Lucia Specia. An exploratory study on multilingual quality estimation. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 366–377, Suzhou, China, December 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.aacl-main.39.