Chapter 2

Vector Aggregation based STS Methods

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

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

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

We address four research questions in this chapter:

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

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

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

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

The main contributions of this chapter are as follows.

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

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

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

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

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

2.1 Related Work

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

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

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

2.1.1 Cosine Similarity on Average Vectors

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

2.1.2 Word Mover's Distance

The second state-of-the-art STS method that we have considered is Word Mover's Distance introduced by Kusner et al. (2015). Word Mover's Distance uses the word embeddings of the words in two texts to measure the minimum distance that the words in one text need to "travel" in semantic space to reach the words in the other text as shown in Figure 2.1. Kusner et al. 2015 shows that this is a better approach than vector averaging since this technique keeps the word vectors as it is, throughout the operation.

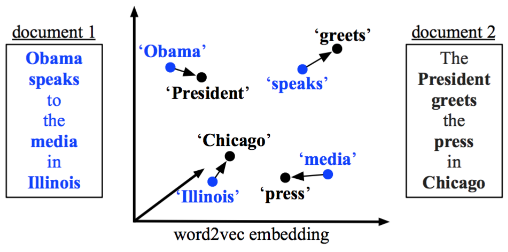

Figure 2.1: The Word Mover's Distance between two sentences (Kusner et al. 2015)

2.1.3 Cosine Similarity Using Smooth Inverse Frequency

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

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

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

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

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

All of these STS methods are based on word embeddings/vectors. The main weakness of word vectors is that each word has the same unique vector regardless of the context it appears. Consider the word "bank", which has several meanings. Still, in standard word embeddings such as GloVe (Pennington et al. 2014b), fastText (Mikolov et al. 2018) or Word2Vec (Mikolov et al. 2013b) each instance of the word has the same representation regardless of the meaning which is used. For example, the word ‘bank’ in two sentences - “I am walking by the river bank” and “I deposited money to the bank” would have the same embeddings, which can be confusing for machine learning models. The recent introduction of contextualised word representations such as BERT (Devlin et al. 2019) and XLNet (Yang et al. 2019b) solved this problem by providing vectors for words taking their context into consideration. In this way, the word ’bank’ in the above sentences would have two different embeddings. Contextual word embedding models have improved the results of many natural language processing tasks over traditional word embedding models (Peters et al. 2018; Devlin et al. 2019). However, they have not been applied to unsupervised vector aggregation-based STS methods to the best of our knowledge.

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

1. **ELMo**[2] introduced by Peters et al. (2018), uses bidirectional language model (biLM) to learn both the word (e.g., syntax and semantics) and linguistic context. After pre-training, an internal state of vectors can be transferred to downstream natural language processing tasks. ELMo vectors have been successfully used in many natural language processing tasks such as text classification (Jiang et al. 2019) and named entity recognition (Luo et al. 2018), which motivated us to explore ELMo in unsupervised STS methods. Also, we were aware of the fact that ELMo has been pre-trained on different languages (Che et al. 2018) and different domains (Jin et al. 2019) which will be useful for when we are adapting our methodology for different languages and domains in Sections 2.3 and 2.4.

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

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

3. **Flair**[5] is another popular contextualised word embedding model introduced by Akbik et al. (2018). It takes a different approach, using a character-level language model rather than the word level language model used in ELMo and BERT. Flair has also been used successfully in natural language processing tasks such as named entity recognition (Akbik et al. 2019b), part-of-speech tagging (Akbik et al. 2018) and has been widely adapted for different languages and domains (Akbik et al. 2018; Sharma

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

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

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

and Daniel Jr 2019).

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

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

2.2 Improving State of the Art STS Methods

As mentioned before, we applied different contextual word embeddings on three unsupervised STS methods and their variants. First, we experimented with English STS datasets that we explained in Section 1.1. Our implementation was based on the *Flair-NLP* Framework (Akbik et al. 2019a) which makes it easier to switch between different word embedding models when acquiring word embeddings. Additionally, *Flair-NLP* has its own *model zoo* of pre-trained models to allow researchers to use state-of-the-art NLP models in their applications. For English, all of these contextualised word embedding models come with different variants such as *small, large etc.* Usually, the larger models provide a better accuracy since they have been trained on a larger dataset than the smaller models. However, this comes with the disadvantage that these larger models are more resource-intensive than the smaller models. To achieve better accuracy, we used the largest model available in each contextual word embedding model. We will describe these models in the following paragraphs.

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

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

As suggested in Akbik et al. (2018), the recommended way to use Flair embeddings is to stack pre-trained 'news-forward' flair embeddings and pre-trained flair 'news-backward' embeddings with GloVe (Pennington et al. 2014a) 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 the pre-trained 'bert-large-uncased' model and 'original

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

(5.5B)' pre-trained ELMo model to represent each word as a 4096 + 3072 vector.

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

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

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

   (a) All the word vectors were considered for averaging. Results are shown in column I of Tables 2.1, 2.2 and 2.3

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

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

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

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

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

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

From the results in Tables 2.1, 2.2 and 2.3 there is no clear indication that contextualised word embeddings perform better than the standard word embeddings. In all of the datasets considered, the best result was provided by Word2vec.

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

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

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

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

All the contextualised word embedding models we considered have more than 3000 dimensions for the word representation, which is higher than the number of dimensions for the word representation we had for standard embeddings - 300. As the vector averaging model is highly dependent on the number of dimensions that a vector can have, the curse of dimensionality might be the reason for the poor performance of contextualised word embeddings in vector averaging variants (Ranasinghe et al. 2019a).

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Model	ρ	τ
Word2vec	0.638	0.601
ELMo	0.641	0.609
Flair	0.639	0.606
BERT	0.650	0.612
ELMo $\bigoplus$ BERT	0.654†	0.616†

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

Model	RMSE
Word2vec	0.599
ELMo	0.585
Flair	0.589
BERT	0.572
ELMo $\bigoplus$ BERT	0.566†

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

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

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

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

2.3 Portability to Other Languages

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

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

[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

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

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

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

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

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

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

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

2.4 Portability to Other Domains

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

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

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

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

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

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

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

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

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

2.5 Conclusions

This chapter experimented with three unsupervised STS methods: cosine similarity using average vectors, Word Mover's Distance and cosine similarity using Smooth Inverse Frequency, with contextualised word embeddings to calculate semantic similarity between pairs of texts and compared them with other unsupervised/ supervised approaches. Contextualised word embeddings could not improve cosine similarity using average vectors and Word Mover's Distance methods, but the results when using the Smooth Inverse Frequency method were improved with contextualised word embeddings instead of standard word embeddings. Furthermore we report that stacking ELMo and BERT provides a stronger word representation than the individual representations of ELMo and BERT. The results indicated that calculating cosine similarity using Smooth Inverse Frequency with stacked embeddings of ELMo and BERT is the best unsupervised method from the available approaches. Also, the performance of our approach was ranked in the top half of the final results list in the SICK dataset, surpassing many complex and supervised approaches.

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

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

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