Chapter 3

STS with Neural Sentence Encoders

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

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

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

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

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

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

We address three research questions in this chapter:

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

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

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

The main contributions of this chapter are as follows.

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

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

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

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

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

3.1 Related Work

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

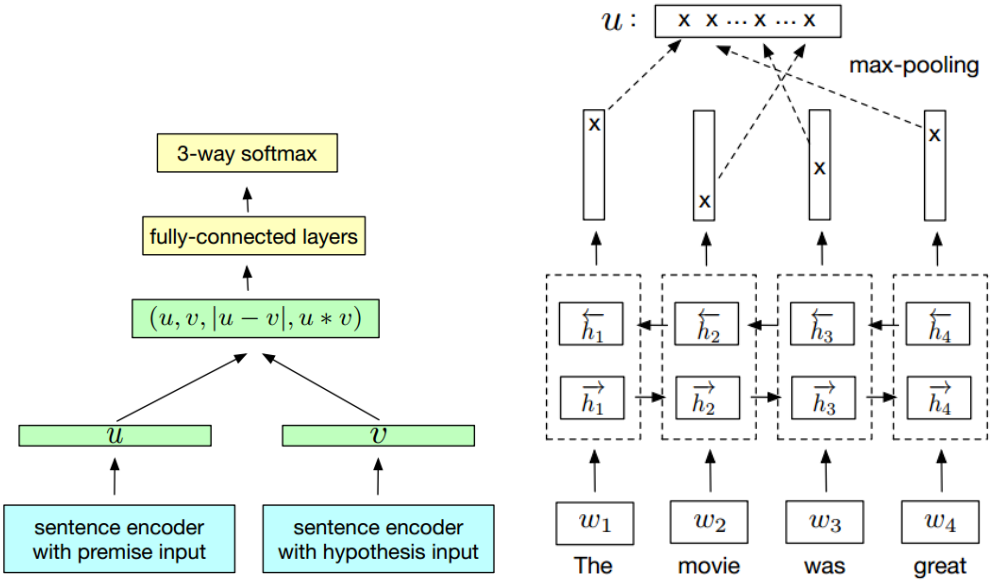
Sent2vec Sent2vec presents a simple but efficient unsupervised objective to train distributed representations of sentences (Pagliardini et al. 2018). It can be considered as an extension of Word2vec (CBOW) to sentences. The key differences between CBOW and Sent2Vec are the removal of the input

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

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

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

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



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

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

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

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

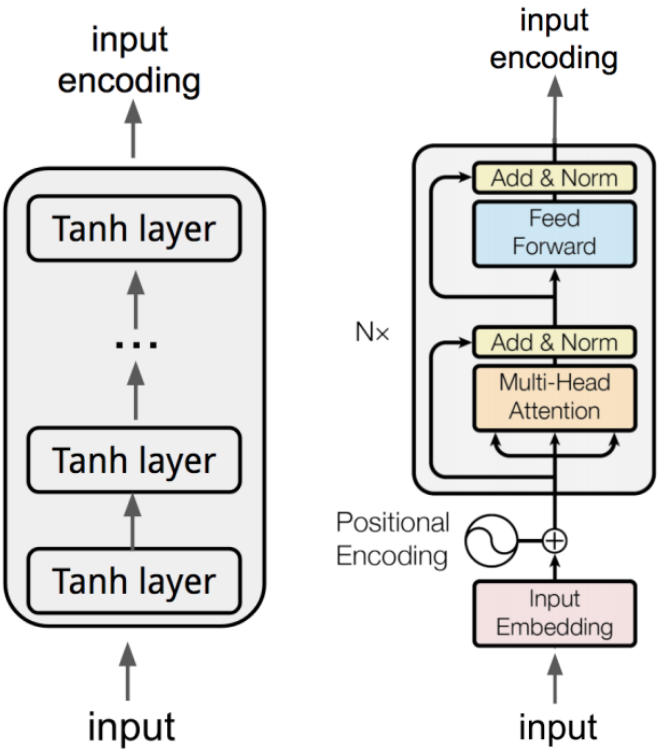
Universal Sentence Encoder The Universal Sentence Encoder (Cer et al. 2018) released by Google is the third and final sentence encoder we employed

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

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

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

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



(a) DAN Architecture (b) Transformer Architecture

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

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

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

3.2 Exploring Sentence Encoders in English STS

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

3.3 Portability to Other Languages

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

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

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

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

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

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

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

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

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

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

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

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

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

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

3.4 Portability to Other Domains

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

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

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

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

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

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

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

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

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

3.5 Conclusions

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

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

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

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