Chapter 3

Exploring STS with Sentence Encoders

3.1 Introduction

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 [28] 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 com-

pletely 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 [24], Infersent [71], Universal Sentence Encoder [25] 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 applications, 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 3.2), 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 3.2 describes the three sentence encoders we experimented in this section. In Section 3.3 we present the experiments we conducted with the three sentence encoders in English STS

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

datasets followed by the results comparing with the other unsupervised STS methods. Section 3.4 and Section 3.5 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.

3.2 Related Work

As we mentioned before, the sentence encoders are popular with the NLP community. The three sentence encoders explored in this chapter: Sent2vec [24], Infersent [25] and Universal Sentence Encoder [71] are the most common sentence encoders. Other than them there is also Doc2vec [72] 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 [24]. 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 [24].

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

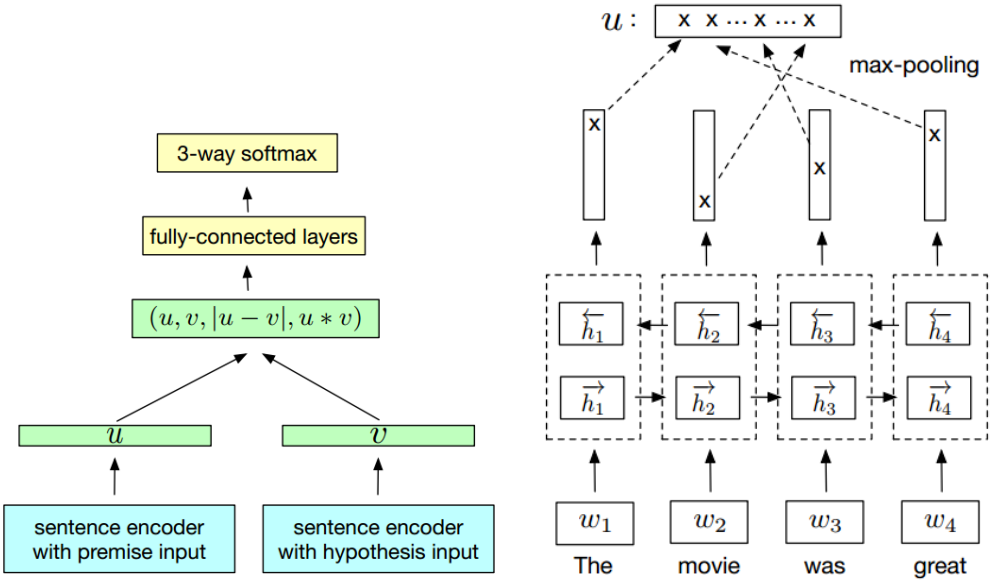
tence embeddings[2]. Also due to its unsupervised approach and simple objective function, Sent2vec has been adopted in different languages and domains too.

Infersent InferSent is an NLP technique for universal sentence representation developed by Facebook which uses supervised training to produce high quality sentence vectors [25]. The authors explore 7 different architectures for sentence encoding including LSTM [74], GRU [75], Bi directional LSTM [76] with mean/max pooling, Self-attentive network and Hierarchical Deep Convolutional Neural Network [25]. 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 [25]. 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 [53] which in turn has been trained on text preprocessed with the PTB tokeniser and the other model is trained with fastText [31] 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 [25].

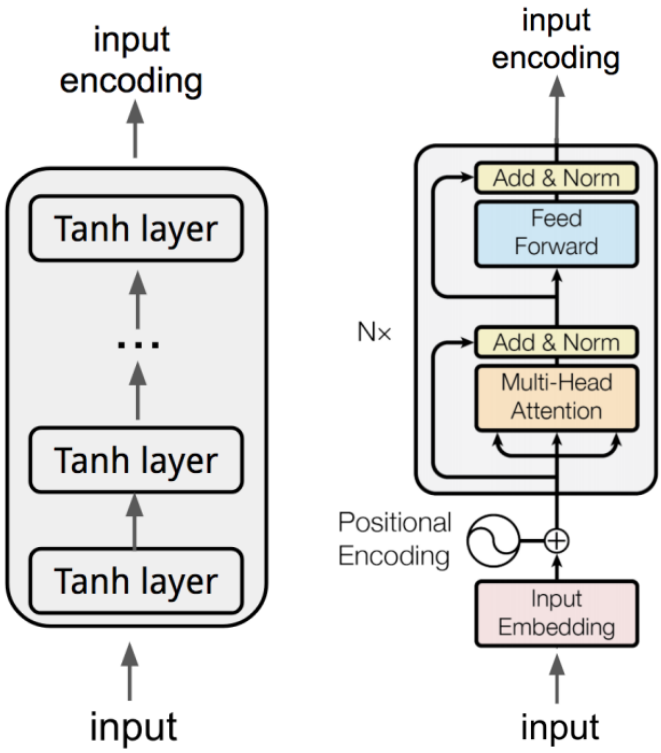
Universal Sentence Encoder The Universal Sentence Encoder [71] released by Google is the last sentence encoder we employed in this chapter. This is again an unsupervised sentence encoder. It comes with two versions i.e. one trained with Transformer encoder and other trained with Deep Averaging Network (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. [71] 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.

3.3 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 1.2. For the experiments we used all the sentence encoders mentioned in Section 3.2. 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 [53] and *infersent2* which was trained using fastText [31]. 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 [19]. 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 1.2; 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†	0.721†

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†

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.

3.4 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 [19]. 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 [77] 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 [78]. 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.

3.5 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 be-

fore, 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 [79]. For **Sent2vec**, we could find a pre-trained model in Bio-medical domain which was trained using PubMed data [80][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 [80]	0.810†

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

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

domain; BioSentVec [80] outperformed other approaches. In fact, when compared 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.

3.6 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 [81]. 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.