Chapter 5

Adapting Transformers for STS

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

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

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

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

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

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

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

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

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

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

The main contributions of this chapter are as follows.

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

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

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

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

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

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

5.1 Related Work

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

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

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

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

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

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

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

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

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

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

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

5.2 Transformer Architecture for STS

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

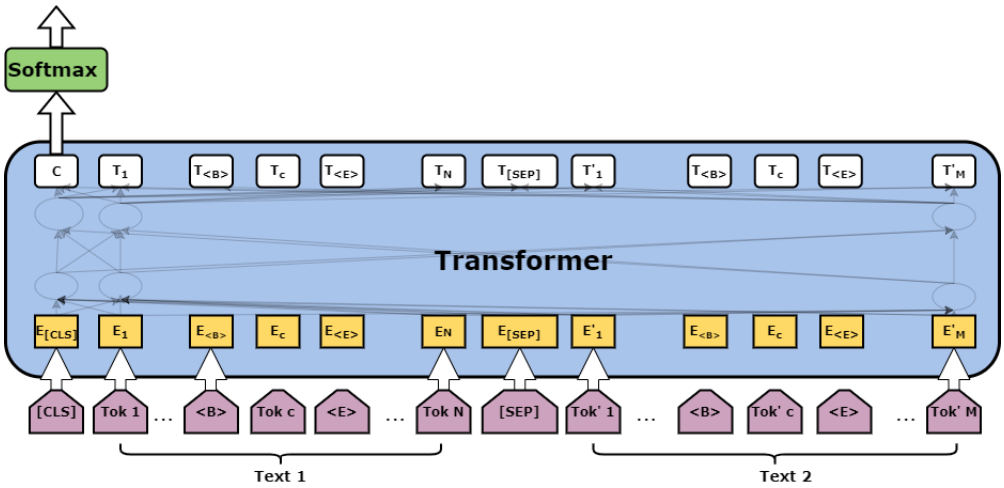

Figure 5.1: Architecture for using Transformers in STS.

5.3 Exploring Transformers in English STS

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

5.3.1 Impact of Transfer Learning

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

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

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

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

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

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

5.3.2 Impact of Data Augmentation

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

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

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

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

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

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

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

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

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

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

5.4 Portability to Other Languages

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

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

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

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

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

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

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

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

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

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

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

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

5.5 Potability to Other Domains

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

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

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

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

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

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

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

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

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

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

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

Chapter.

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

5.6 Recent Developments: Siamese Transformers

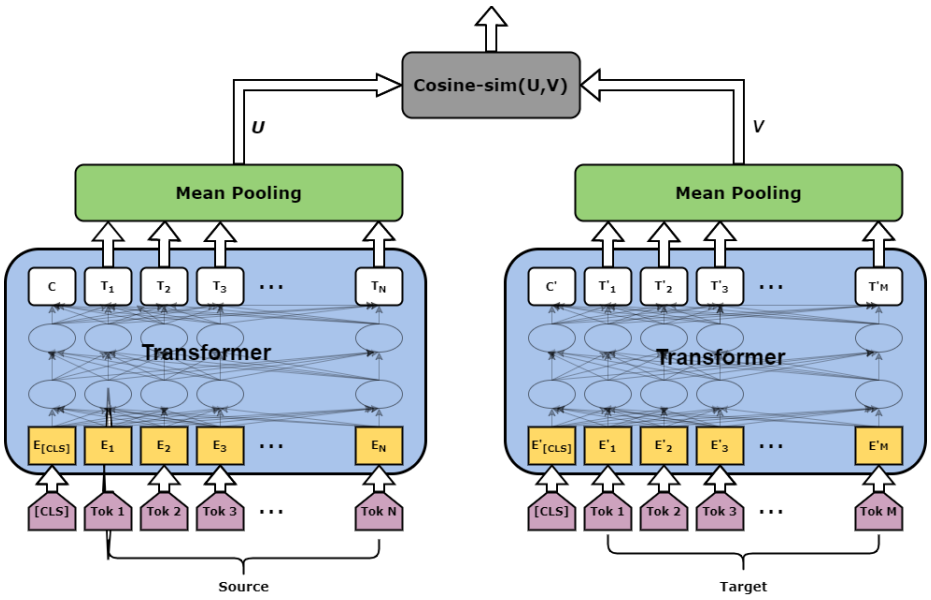

Figure 5.2: Architecture for using Siamese Transformers in STS.

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

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

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

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

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

5.7 Conclusions

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

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

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

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

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

With this, we conclude Part I of the thesis. We explored numerous STS methods based on word vectors and easy to adapt in different languages and

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