WarwickNLP at SemEval-2024 Task 1: Low-Rank Cross-Encoders for Efficient Semantic Textual Relatedness

Fahad Ebrahim
The University of Warwick
Coventry, United Kingdom
Fahad.Ebrahim@warwick.ac.uk

Mike Joy
The University of Warwick
Coventry, United Kingdom
M.S.Joy@warwick.ac.uk

Abstract

This work participates in SemEval 2024 Task 1 on Semantic Textural Relatedness (STR) in Track A (supervised regression) in two languages, English and Moroccan Arabic. The task consists of providing a score of how two sentences relate to each other. The system developed in this work leveraged a cross-encoder with a merged fine-tuned Low-Rank Adapter (LoRA). The system was ranked eighth in English with a Spearman coefficient of 0.842, while Moroccan Arabic was ranked seventh with a score of 0.816. Moreover, various experiments were conducted to see the impact of different models and adapters on the performance and accuracy of the system.

1 Introduction

Semantic Textual Relatedness (STR) is a measure of how closely related or connected two linguistic units are in terms of their meanings or concepts (Abdalla et al., 2023). STR is a valuable concept in Natural Language Processing (NLP), as it helps us to understand the connections and similarities between different pieces of text. By determining the degree of relatedness between sentences or phrases, we can improve various NLP tasks such as information retrieval, question answering, and text summarisation. This understanding of semantic relatedness enables us to create more accurate word embeddings and sentence representations, enhancing the performance of language processing models. One way to represent STR is a supervised regression task in which the output is a continuous score number between 0 and 1.

For the STR task in SemEval 2024 (Ousidhoum et al., 2024b), the organisers on Track A (supervised) provided datasets (Ousidhoum et al., 2024a) for nine languages or dialects. They provided pairs of sentences and annotated the degree of relatedness via a human score between 0 and 1. The languages considered in this work are English and Moroccan Arabic. These two were selected because they are comprehensive for the team.

There are various methods that can be used to estimate the relatedness of two sentences. One of the methods is to utilise the Pre-Trained Language Models (PLMs). PLMs are currently state-of-the-art in the field of NLP, and follow the transformer architecture introduced in (Vaswani et al., 2017) with the attention mechanism. Another variation that uses a mechanism of cross-attention is the cross encoder (Reimers and Gurevych, 2019), whichthat of takes two inputs and outputs a score between 0 and 1 on how related these two inputs are. They are efficient in determining the correlation between two inputs.

Parameter-efficient fine-tuning (PEFT) aims to tune the pre-trained model with high accuracy but with less cost and complexity. One of the PEFT methods is adapters (Poth et al., 2023), which tune extra parameters or layers instead of tuning the whole model while maintaining competitive accuracy. They can be considered as few-shot learners as per (Beck et al., 2022). One type of adapter is the low-rank adapter (LoRA). Instead of tuning the whole weight, LoRA adds small matrices in each layer, and these matrices would be fine-tuned.

Hence, this work applies a tuned LoRA adapter on a pre-trained cross-encoder to estimate the score of the relatedness of two sentences. The code is publicly available[1].

The paper is organised as follows: Section 2 presents the background, including related work and dataset overview; Section 3 covers the system overview; Section 4 presents the results; Section 4 discusses the error analysis and limitations; and the paper concludes.

[1] https://github.com/FahadEbrahim/STR_LoRA

2 Background

This section will cover the related work, dataset description and a brief introduction to PEFT.

2.1 Related Work

There have been previously related versions of the STR datasets, such as (Asaadi et al., 2019) and (Abdalla et al., 2023). Different versions use different languages, annotations or available datasets.

Another related dataset is Semantic Textual Similarity (STS) (Cer et al., 2017). There are differences between STS and STR. Specifically, STS tasks aim to assess how similar two text segments or sentences are, focussing on tasks such as identifying paraphrases or entailment relationships. On the other hand, STR looks at the overall closeness in meaning between linguistic units, considering various factors such as topic-relatedness and stylistic similarities. Thus, STR is more general, and STS can be treated as a subset of STR. Secondly, the outputs differ slightly, as the output of the previous STS tasks is between zero and five, while in STR, the output is between zero and 1. STS datasets were beneficial in this task, as can be seen later in the paper.

The task of STS is a common natural language understanding task. Several approaches have been used for STS. One of the popular approaches utilises the generation of robust contextual embeddings and then uses a similarity measurement like cosine similarity to get the required score. The embeddings can be extracted with a Universal Sentence Encoder (USE) (Cer et al., 2018), Language-agnostic BERT Sentence Embedding (LabSE) (Feng et al., 2022) or Sentence BERT (SBERT) (Reimers and Gurevych, 2019). These are different approaches to get meaningful embeddings that can capture the semantics of the input sentences.

2.2 Dataset

The datasets (Ousidhoum et al., 2024a) provided for training consist of two sentences and a score of how related they are between 0 and 1. Sample instances of the English training dataset can be seen in Table 1. The first example shows two sentences with the same meaning, and therefore the score is 1. The second example shows partially related sentences with a score of 0.5. The last example includes two unrelated sentences with a score of around 0.

Sentences	Score
Actor Gazzara dead at 81 Actor Ben Gazzara dies at 81	1.0
yeah and so is bubbles lol Bubbles used to reside next door	0.5
A child wielding a snow shovel. A cat bites a human's nose.	0.03

Table 1: Dataset training sample instances.

The datasets are split into 3 sets: training, development, and testing. The number of instances in each set in the English and Moroccan Arabic languages can be seen in Table 2. The main reason for injecting adapters directly is that there are few training samples in Moroccan Arabic. So, few-shot learning is a better approach for this language and, therefore, for the overall task.

Set/Language	English	Moroccan Arabic
Train	5500	925
Evaluation	250	70
Testing	2500	427

Table 2: STR Dataset split.

2.3 Parameter Efficient Fine-Tuning

PLMs follow the Encoder/Decoder architecture introduced by Transformer (Vaswani et al., 2017). Models such as BERT (Devlin et al., 2018), RoBERTa (Liu et al., 2019), ALBERT (Lan et al., 2019), T5 (Raffel et al., 2020), and DeBERTa (He et al., 2020) are other variations of the transformer architecture.

Another variation that uses cross-attention is the cross-encoder (Reimers and Gurevych, 2019), which takes two inputs and outputs a score between zero and one. A cross-encoder trained on the STS-B benchmark (Cer et al., 2017) would result in an output score between 0 and 1 instead of 0 to 5.

One of the PEFT techniques is the use of adapters, which are efficient few-shot learners as per (Poth et al., 2023). There are various adapter architectures. Instead of tuning the entire model weights, the adapters would tune additional parameters, layers or weight matrices. The three types of adapters investigated in this work are Houlsby (Houlsby et al., 2019), Pfeiffer (Pfeiffer et al., 2021), and LoRA (Hu et al., 2021). The Houlsby adapter adds two additional layers before and after the feed-forward (FF) layer in each encoder, while

Pfeiffer adds only a single layer after the FF layer. The LoRA adapter adds small weight matrices in each layer of the transformer layers.

This work tunes several adapters on different pre-trained models and checks which combination performs best. The best architecture will be explained in the next section.

3 System Development

This section covers the cross-encoder, LoRA adapter, the developed system architecture, and the evaluation metric.

3.1 Cross-Encoder

The cross-encoder takes two inputs simultaneously and uses the concept of cross-attention allowing the model to capture interactions between the two sentences. The cross-encoder would generate a score between 0 and 1 indicating how similar the two inputs are. The technical architecture of the cross-encoder can be seen in Figure 1. The cross-encoder adds two special tokens: SEP to separate the two sentences and CLS. The CLS token gets added at the beginning of the concatenated sentences along with the SEP token and represents a classification vector. The initial CLS token identifies the representation of two sentences. The final CLS token captures the cross-attention between all previous CLS tokens and produces one final semantic vector. A classification head maps the final CLS vector to a score between 0 and 1 based on the chosen model.

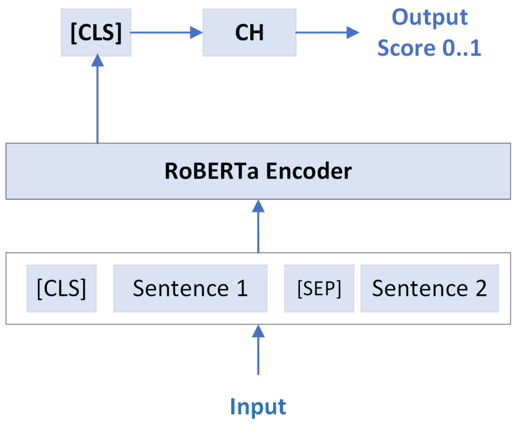

Figure 1: Cross-Encoder Architecture.

3.2 LoRA Adapter

The LoRA adapter (Hu et al., 2021) adds two additional low-rank matrices that are trainable instead of training the whole model. To explain LoRA mathematically, assume the input to a neural network to be X and the output to a single hidden layer is $h(x)$, then the output with full fine-tuning would equal the input multiplied by a weight matrix W_0 as per Equation 1. The weight matrix W_0 belongs to the dimension of $(d * k)$.

$$h(x) = W_0 X \qquad W_0 \in \mathbb{R}^{d \times k} \tag{1}$$

In LoRA, an additional weight matrix ΔW_0 is added into the input and initial weight matrices to get the hidden layer output as per equation 2. It belongs to the same dimension as the original matrix.

$$h(x) = W_0 X + \Delta W_0 X \qquad W_0, \Delta W_0 \in \mathbb{R}^{d \times k} \tag{2}$$

The new matrix is decomposed into two trainable matrices, B and A, with dimensions $(d * r)$ and $(r * k)$, respectively. The value of r (rank) is much smaller than d and k ($r \ll d, r \ll k$), so the new two matrices are smaller than the initial matrix. This reduces the model's trainable parameters while maintaining high accuracy. The value of r is a hyper-parameter and it is used as 8 in this work.

$$h(x) = W_0 X + (BA)X \qquad B \in \mathbb{R}^{d \times r}, A \in \mathbb{R}^{r \times k} \tag{3}$$

Applying LoRA can reduce the size of the model from hundreds of megabytes to just a few megabytes (Hu et al., 2021) while maintaining a high level of accuracy.

3.3 System Architecture

The architecture of the system developed for the STR task can be seen in Figure 2. The following are the simplified steps.

1. Initialise the adapter with random weights.

2. The LoRA adapter is trained given the training sets. During training, only the low-rank matrices are fine-tuned.

3. The adapter merges with the classification head of the cross-encoder.

4. The testing data are fed into the cross-encoder with the attached adapter to get the relatedness score.

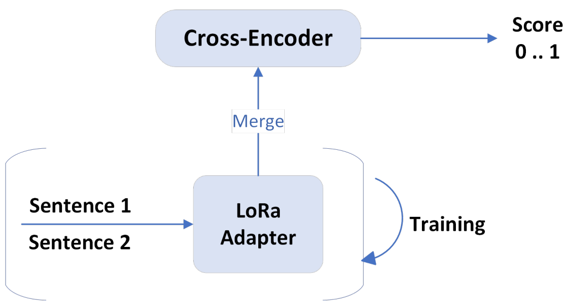

Figure 2: System Overview.

3.4 Evaluation

The evaluation is based on Pearson's correlation coefficient as in Equation 4. The values of X_i and Y_i represent individual instances while the values $\bar{X}$ and $\bar{Y}$ represent the mean of X and Y. The higher the coefficient, the better the model evaluation. The value of the coefficient ranges between -1 to 1 where 1 represents a higher correlation between the inputs. The metric value is generated with submissions being uploaded through CodaLab (Pavao et al., 2023). CodaLab is a platform for competitions and research purposes. The platform returns the scores for development and testing.

$$r = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2 \sum_{i=1}^{n}(Y_i - \bar{Y})^2}} \tag{4}$$

In addition to the datasets, the organisers provided a baseline for all languages based on LabSE (Feng et al., 2022), which provides sentence embeddings. Scores were calculated based on cosine similarity between these embeddings.

4 Results

The system would initially be trained on the training set and evaluated with the development set, and the official results would be produced by applying the model to the testing sets. This work's approach is incremental and experimental. So, variations of models and adapters would be tested, and the models achieving better metrics would be selected.

4.1 Experimental Development Results

All the initial experiments were conducted in English. Then, the best approach was applied to Moroccan Arabic. The experiments carried out in the development phase were incremental in 10 epochs. Firstly, several models (BERT, ALBERT, RoBERTa, DeBERTa, and T5) were fine-tuned with the Pfeiffer adapter and the model with the best-yielding scores was selected. The models and their respectable Spearman scores can be seen in Table 3. The model with the highest score was RoBERTa, with a development score of 0.8155. The exact model names in Huggingface (Wolf et al., 2019) are available in Appendix A. Huggingface is a platform with a large number of pre-trained models. The initial selected models were general and not domain-specific. Therefore, a model trained for similarity could be investigated. The chosen model was a cross-encoder trained with the STS-B dataset. The results were better than the base RoBERTa model, reaching a development score of 0.8296. So, the system continued using this model.

Model	Score
BERT	0.8023
ALBERT	0.7755
DeBERTa small	0.8088
T5 small	0.8003
RoBERTa base	0.8155
RoBERTa STSB-CE	0.8296

Table 3: Selected models and their development scores.

Then, the impact of merging several single adapters (Houlsby, Pfeiffer and LoRA) on the cross-encoder was studied. Table 4 shows the scores with the attachment of the three adapters. The LoRA adapter scored the highest with a Spearman coefficient of 0.8343. To further improve the accuracy, the training epochs were increased to 30 to improve the generalisation of the model, and this was achieved by producing a final development score of 0.8417.

Adapter	Score
Pfeiffer	0.8296
Houlsby	0.8309
LoRA	0.8343

Table 4: Impact of different adapters on the RoBERTa STSB-CE in the development phase.

The same configuration was used for the Moroccan Arabic language, resulting in a development score of 0.8577.

To see whether an Arabic model would perform differently in an Arabic pre-trained model, another experiment in Moroccan Arabic was conducted in the development set using the CAMeLBERT model (Inoue et al., 2021). This model was trained on a large corpus of Arabic for different tasks with several dialects (modern standard Arabic, dialectal Arabic, and classical Arabic). There is also a version of the CAMeLBERT model trained on the combination of the three dialects. This version produced a higher score of 0.871 on the Moroccan Arabic development set. However, this result was not submitted for the official competition.

4.2 Official Results

The same configuration used in the development phases was applied in the test set. The model used for both languages was the cross-encoder RoBERTa trained on STS-B. The adapter was LoRA and the training epochs were 30. The rest of the parameters can be seen in Appendix B. The official results reported can be seen in Table 5. The scores for English and Moroccan Arabic were 0.8425 and 0.8163 respectively. Both exceed the baseline scores of 0.83 and 0.77 respectively.

System	Baseline	Our System
English	0.83	0.8425
Morrocan Arabic	0.77	0.8163

Table 5: Official submitted scores of the competition in the English and Morrocan Arabic.

5 Discussion

5.1 Error Analysis

One interesting negative result was found during development. The developed system was applied to Algerian Arabic, which yielded a low evaluation metric of 0.54. This could be attributed to the fact that this dialect had majorly unseen data for the model. The STS-B dataset has some overlap with the two languages (English and Moroccan Arabic) worked on in this paper.

The development set with labels was not utilised in the testing phase. Moreover, the differences between applying the adapters on the cross-encoder are minor. So, applying these models to the testing set may yield different results.

The results on CAMelBERT on Moroccan Arabic were better than the cross-encoder in the development phase, but this was not considered in the system's official results to maintain the consistency of the used model (cross-encoder). The usage of CAMelBERT yielded a metric value of 0.8347, which is higher than the cross-encoder. This was noticed in the post-evaluation phase and, therefore, not reported in the official results. This indicates that using a pre-trained model on a large corpus of Arabic yields a better result than the cross-encoder trained on the STSB dataset.

5.2 Limitations

Due to time constraints, this work has several limitations. Firstly, the models were not fully fine-tuned with the training data. It would be comprehensive to compare the full fine-tuning to the adapter tuning and discuss the obtained results. Secondly, there was no hyper-parameter optimisation was done in this system except for the number of epochs. The effect of applying different parameters to different models would yield a better overview of the effect of these hyper-parameters on the training process. Thirdly, full training of the cross-encoder would be an interesting scope of future work. Lastly, there was no pre-processing, and the sentences were tokenised as they were. The impact of pre-processing could be studied.

5.3 Ethical Statement

This work used only the STS-B and STR datasets, which are both publicly available. Although the dataset has some overlap between the development and testing sets, this work maintains that the development set was not used for training. This is attributed to the ethical convention in machine learning of using the appropriate set for the suitable task.

Conclusion

This work involves participating in the SemEval 2024 STR task in two languages (English and Moroccan Arabic). The regression task is to predict a score relating to two sentences. The developed system consists of a LoRA adapter trained on the given training instances and then attached to a cross-encoder previously trained on the STS-B dataset. The system achieved excellent performance, exceeding the baseline and ranking seventh in Moroccan Arabic and eighth in English, with Pearson Coefficient scores of 0.8163 and 0.8425, respectively.

Acknowledgements

The authors thank the organisers for answering questions about the task and express their gratitude to the anonymous reviewers for their constructive feedback.

References

Mohamed Abdalla, Krishnapriya Vishnubhotla, and Saif Mohammad. 2023. What makes sentences semantically related? a textual relatedness dataset and empirical study. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 782–796.

Shima Asaadi, Saif Mohammad, and Svetlana Kiritchenko. 2019. Big bird: A large, fine-grained, bigram relatedness dataset for examining semantic composition. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 505–516.

Tilman Beck, Bela Bohlender, Christina Viehmann, Vincent Hane, Yanik Adamson, Jaber Khuri, Jonas Brossmann, Jonas Pfeiffer, and Iryna Gurevych. 2022. AdapterHub playground: Simple and flexible few-shot learning with adapters. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 61–75, Dublin, Ireland. Association for Computational Linguistics.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada. Association for Computational Linguistics.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Fangxiaoyu Feng, Yinfei Yang, Daniel Cer, Naveen Arivazhagan, and Wei Wang. 2022. Language-agnostic bert sentence embedding. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 878–891.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Go Inoue, Bashar Alhafni, Nurpeiis Baimukan, Houda Bouamor, and Nizar Habash. 2021. The interplay of variant, size, and task type in Arabic pre-trained language models. In *Proceedings of the Sixth Arabic Natural Language Processing Workshop*, Kyiv, Ukraine (Online). Association for Computational Linguistics.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Nedjma Ousidhoum, Shamsuddeen Hassan Muhammad, Mohamed Abdalla, Idris Abdulmumin, Ibrahim Said Ahmad, Sanchit Ahuja, Alham Fikri Aji, Vladimir Araujo, Abinew Ali Ayele, Pavan Baswani, Meriem Beloucif, Chris Biemann, Sofia Bourhim, Christine De Kock, Genet Shanko Dekebo, Oumaima Hourrane, Gopichand Kanumolu, Lokesh Madasu, Samuel Rutunda, Manish Shrivastava, Thamar Solorio, Nirmal Surange, Hailegnaw Getaneh Tilaye, Krishnapriya Vishnubhotla, Genta Winata, Seid Muhie Yimam, and Saif M. Mohammad. 2024a. Semrel2024: A collection of semantic textual relatedness datasets for 14 languages.

Nedjma Ousidhoum, Shamsuddeen Hassan Muhammad, Mohamed Abdalla, Idris Abdulmumin, Ibrahim Said Ahmad, Sanchit Ahuja, Alham Fikri Aji, Vladimir Araujo, Meriem Beloucif, Christine De Kock, Oumaima Hourrane, Manish Shrivastava, Thamar Solorio, Nirmal Surange, Krishnapriya Vishnubhotla, Seid Muhie Yimam, and Saif M. Mohammad. 2024b. SemEval-2024 task 1: Semantic textual relatedness for african and asian languages. In *Proceedings of the 18th International Workshop on Semantic Evaluation (SemEval-2024)*. Association for Computational Linguistics.

Adrien Pavao, Isabelle Guyon, Anne-Catherine Letournel, Dinh-Tuan Tran, Xavier Baro, Hugo Jair Escalante, Sergio Escalera, Tyler Thomas, and Zhen Xu. 2023. Codalab competitions: An open source platform to organize scientific challenges. *Journal of Machine Learning Research*, 24(198):1–6.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021. AdapterFusion: Non-destructive task composition for transfer learning. In *Proceedings of the 16th Conference of the European Chapter of the Association* *for Computational Linguistics: Main Volume*, pages 487–503, Online. Association for Computational Linguistics.

Clifton Poth, Hannah Sterz, Indraneil Paul, Sukannya Purkayastha, Leon Engländer, Timo Imhof, Ivan Vulić, Sebastian Ruder, Iryna Gurevych, and Jonas Pfeiffer. 2023. Adapters: A unified library for parameter-efficient and modular transfer learning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 149–160, Singapore. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

A Model Names

The exact model names in Huggingface[2] used in this work are available in Table 6.

Table 6: Model Names in Huggingface.

Model	**Model Name**
BERT	google-bert/bert-base-uncased
RoBERTa	FacebookAI/roberta-base
DeBERTa	microsoft/deberta-v3-small
ALBERT	albert/albert-base-v2
T5	google-t5/t5-small
RoBERTaCE	cross-encoder/stsb-roberta-base
CAMeLBERT	CAMeL-Lab/ bert-base-arabic-camelbert-mix

[2]`https://huggingface.co/`

B Hyper-parameters Configuration

The hyper-parameters used to develop the system are available in Table 7. The number of epochs during the development phase was 10, while in testing, it was increased to 30.

Parameter	**Value**
Epoch	10 - 30
Learning rate	5×10^{-4}
Batch size	30
LoRA (r and alpha)	8
GPU	Tesla A100 (40GB)

Table 7: Parameter configuration.

C Used Libraries

This section includes the Python libraries used in the code and their versions according to Table 8.

Library	**Version**
adapters	0.1.1
transformers	4.35.2
datasets	2.17.0
sklearn	1.2.2
torch	2.1.0
accelerate	0.27.0
pandas	1.5.3
numpy	1.23.5

Table 8: Used Python libraries.