2022 IEEE 19th India Council International Conference (INDICON) | 978-1-6654-7350-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/INDICON56171.2022.10039840

A Cognitive Study on Semantic Similarity Analysis of Large Corpora: A Transformer-based Approach

Praneeth Nemani, Satyanarayana Vollala
Dept. of Computer Science and Engineering
IIIT Naya Raipur
Raipur, Chhattisgarh, India
praneeth19100, satya@iiitnr.edu.in

***Abstract*—Semantic similarity analysis and modeling is a fundamentally acclaimed task in many pioneering applications of natural language processing today. Owing to the sensation of sequential pattern recognition, many neural networks like RNNs and LSTMs have achieved satisfactory results in semantic similarity modeling. However, these solutions are considered inefficient due to their inability to process information in a non-sequential manner, thus leading to the improper extraction of context. Transformers function as the state-of-the-art architecture due to their advantages like non-sequential data processing and self-attention. In this paper, we perform semantic similarity analysis and modeling on the U.S Patent Phrase to Phrase Matching Dataset using both traditional and transformer-based techniques. We experiment upon four different variants of the Decoding Enhanced BERT - DeBERTa and enhance its performance by performing K-Fold Cross-Validation. The experimental results demonstrate our methodology's enhanced performance compared to traditional techniques, with an average Pearson correlation score of 0.79.**

***Index Terms*—Semantic Similarity, K-Fold Cross Validation, Pearson Correlation, Transformers**

I. Introduction

Semantic similarity is defined as the association between two blocks of text, including sentences, words, and documents. It plays a fundamentally acclaimed role in most of the NLP tasks performed by researchers worldwide today. The dynamic and versatile nature of human language makes it difficult to standardize the process of semantic similarity [1]. As time evolves, finding new semantic analysis techniques is deemed essential due to the exponential rise of textual data generation. The conceptual overview of semantic similarity analysis is depicted in Fig. 1.

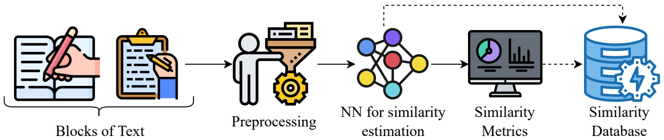

Fig. 1: Conceptual Overview of Semantic Similarity Analysis

As mentioned above, semantic similarity analysis is pivotal in various applications like information recovery, text summarization, speech enhancement, and automatic dialogue generation. In the initial methodologies proposed by researchers worldwide, semantic similarity was calculated based on the number of similar words in two blocks of text. However, this yielded inaccurate results as there were instances where two blocks of text had many similar words but conveyed different meanings. For example, the sentences *Tom and Harry played Badminton and Cricket*, and *Tom played Badminton and Harry played Cricket* are lexically similar. However, the context of these two sentences was not the same. Conversely, the sentences *Jenny knows many languages* and *Jenny is a polyglot*, though lexically dissimilar, convey the same meaning. Similarity analysis based on lexical similarity methods is easy to implement, but they fail in sentences that are lexically dissimilar and semantically similar.

Today's pioneering ML algorithms and their applications use the method of vectorization for the process of feature extraction. The concept of vectorization in NLP is that each word/phrase in a dataset is represented as a vector or an array of numbers, making feature extraction easier owing to today's computers' computational efficiency. Some techniques that use this vectorization concept include Bag of Words [2] and TF-IDF [3]. However, the major limitation of these solutions is that they do not consider the broader context of the text block for computing semantic similarity. The broader context between two blocks of text is inversely proportional to their semantic distance.

Recurrent Neural Networks and Long Short-Term Memory networks, abbreviated as RNNs and LSTMs, respectively, have been considered effective techniques for learning dependencies between blocks of text. While RNNs depend on the recent previous blocks, LSTMs depend on the broader context of the text. However, they are considered inferior to transformers due to their limitation of sequential data processing. The significant advantages of transformers like the training of large corpus, non-sequential data processing, self-attention, and techniques like positional embeddings to replace recurrent have popularized them for performing modern-day NLP tasks. This paper provides an insight into how modern-day transformers can be used for the task of semantic similarity analysis. The major contributions of our paper can be listed below:

- We present a comprehensive study on the different methodologies used for the process of semantic similarity analysis.
- We also perform the state-of-the-art preprocessing tech-

978-1-6654-7350-7/22/$31.00 ©2022 IEEE

niques and exploratory data analysis on the U.S Patent Phrase-to-Phrase matching dataset.
- Exclusive experimentation and analysis of one traditional and four transformer-based techniques is performed to extract context and perform semantic similarity analysis.

II. Related Research Overview

Research on Semantic Analysis has been a topic of interest since the 20th century. Many solutions have been proposed with their implementation on many benchmark datasets. This section presents a comprehensive survey of datasets and different methodologies used for Semantic Similarity Analysis. Table I gives an overview of the widely-used datasets for semantic similarity.

TABLE I: Popular Datasets for semantic similarity

Author	Dataset	Word/Sentence pairs	Similarity Score Range
Rubenstein et al. [4]	R&G	65	0-4
Miller et al. [5]	M&C	30	0-4
Finkelstein et al. [6]	WS353	353	0-10
Agirre et al. [7]	STS2015	3000	0-5
Marelli et al. [8]	SICK	10000	1-5
USTPO	Patent Phrase Matching	33000	0-1

Benajiba et al. [9] proposed a solution involving a Siamese LSTM regression model that is used to predict the similarity of the SQL template of two questions. The authors defined a metric called the SQL structure distance used to estimate the similarity using the proposed methodology. To reduce the computational cost of the solution, the authors have clustered the training set samples with the one-hot lexical representation of the questions. Li et al. [10] proposed another solution involving semantic similarity in biomedical sentences using an SNN approach. The methodology involved the integration of an interactive self-attention (ISA) mechanism and an SNN. The proposed solution was validated on three standard biomedical datasets with an average Pearson score of 0.65. Pontes et al. [11] proposed a Siamese CNN + LSTM model in which the CNN extracts the local context while the LSTM extracts the global context. The proposed methodology was evaluated on the SICK dataset with different combinations of local context and global context.

Quan et al. [12] proposed a framework combining the capability of word embeddings and attention weight mechanism by integrating them into a unified network known as the Attention Constituency Vector Tree (ACVT). The proposed solution was validated on 19 benchmark datasets which include STS'12-STS'15 with a Pearson score of 0.75. Shancheng et al. [13] proposed a double sequential network consisting of identical LSTM layers that simultaneously train two sequences of sentences. The outputs of both the layers were passed through the dense layer and compressed to obtain the semantic similarity. The proposed solution addressed the problem of Chinese characteristics and was compared with the Baidu Semantic Text Similarity model and achieved higher accuracy. Yang et al. [14] proposed a methodology that involved an extensive semantic network known as Probase. From the current weights and parameters of probase, the semantic similarity was performed on the MG, WS353-Sim, and RG datasets.

In recent years, Generative Adversarial Networks (GANs) have gained tremendous popularity in artificial data generation for various tasks, including image sample generation with limited data [15] and text generation. In this view, Liang et al. [16] addressed the generation and identification of similar sentences using a GAN-based approach. The authors proposed a syntactic and semantic long short-term memory (SSLSTM) algorithm for evaluating semantic similarity. Three variations of the sentence similarity generative adversarial network (SSGAN) algorithm were proposed for generating sentences. The state-of-the-art solutions for tasks in natural language processing involve the usage of transformers. Precisely, transformers in NLP are used to solve NLP tasks involving the dependency of long sequences. In this context, Li et al. [17] introduced a hybrid Cross2self attention, Bi-RNN - BERT model to computer semantic similarity in biomedical data. The methodology was validated on the OHNLP2018 baselines with an increase of 0.6% in the Pearson coefficient. Another approach involving the usage of BERT for semantic similarity of outlook emails was proposed by Sanjeev et al [18]. Some of the standard approaches in NLP for semantic analysis include Word2Vec, proposed by Google in 2013 and the Glove model. The related research overview could be summarized in Table II.

TABLE II: Overview of the existing solutions

Author	Methodology	Dataset used
Benajiba et al. [9]	Siamese LSTM Regression	WikiSQL
Li et al. [10]	ISA + Siamese NNs	DBMI , CDD-ful, CDD-ref
Pontes et al. [11]	Siamese CNN + LSTM	SICK dataset
Quan et al. [12]	Attention Constituency Vector Tree (ACVT)	STS'12-STS'15
Shancheng et al. [13]	Double Seq. NN + LSTM	Chinese semantic similarity dataset
Yang et al. [14]	Probase	M&G, WS353-Sim, and R&G
Liang et al. [16]	SSLSTM + SSGAN	SemEval and Quora
Li et al. [17]	Cross2self, BERT	Biomedical Data
Sanjeev et al [18]	BERT	Outlook Emails
Our Work	**DeBERTa + K-Fold Stratified Cross Validation**	**U.S Patent Phrase to Phrase Matching Dataset**

III. Methodology

This section deals with the different techniques used to perform the task semantic similarity analysis. In this work, we compare and analyze the performance of five different techniques used for semantic similarity analysis. These include Levenshtein Metric similarity and four different variants of the DeBERTa model. This section deals with the architecture of each model and how it can be fine-tuned for our dataset to perform the required task.

A. Levenshtein Metric

In Natural Language Processing, the Levenshtein distance between two words is defined as the number of single-character edits required to convert one word from other [19]. It is a string metric used to understand the disparity between two different sequences. Edits can be defined as insertion, replacement, and deletion in this context. Some of the Levenshtein Distance applications include DNA Analysis and Plagiarism Checking. In this task of semantic similarity analysis, we experiment with the approach of the Levenshtein Distance on our dataset.

B. DeBERTa

As mentioned in section II, there has been a remarkable rise in the usage of transformers in many NLP tasks like semantic analysis and dialogue generation. BERT (Bidirectional Encoder Representations from Transformers) has been acknowledged as a recent advancement in transformers by researchers at Google in their work [20]. The concept of BERT lies in the fact that when sequential data is trained in a bi-directional manner, better and deeper inference can be obtained on the data for specific tasks like language understanding. In Machine Vision, transfer learning is a widely used technique by researchers across the globe to perform various tasks rather than training a model from the onset. The idea of transfer learning is that existing deep learning models could be transformed into objective-specific models by fine-tuning the existing model. This approach has gained significance among NLP researchers worldwide, and transfer learning could now be applied to many NLP tasks. BERT employs the usage of a transformer, a mechanism that is based on attention that comprehends the contextual inference between two words in a corpus. The simplest form of BERT has an encoder and a decoder. The purpose of the encoder is to comprehend the input text, while the decoder's purpose is to deliver a prediction.

Since 2018, there has been a rapid rise in the design and development of pre-trained language models like GPT, T5, RoBERTa, StructBERT, and DeBERTa [21]. However, in this work, we emphasize the different versions of DeBERTa and their performance in the U.S Patent to Phrase matching dataset. DeBERTa is a Decoding Enhanced BERT with disentangled attention, which functions based on introducing two novel techniques: Disentangled attention and enhanced masked decoding. The concept of disentangled attention is that each word or token in the input layer is represented by two vectors corresponding to its content and position in the corpus. This is inferred from the fact that the word's position also has significant importance in content extraction. However, though disentangled attention conveys the relative positions of words, it is deemed essential to determine the exact position of words in a corpus to avoid semantic disparity. So to achieve this, DeBERTa integrates the positional word embeddings prior to its softmax layer. Owing to its architecture, DeBERTa is considered significantly superior to its counterparts like RoBERTa [22].

The input to this pipeline is the *anchor phrase + seperated token + the context phrase*. DeBERTa uses a metric known as the cross-attention score to infer the semantic similarity between two blocks of text. Mathematically, the cross attention score of a block *m* with respect to another block *n* can be represented as shown in Eq. 1 where C_m represents the content of the word and $Pos_{m|n}$ represents the position of the word *m* with respect to *n*. The cross-attention score between two blocks *m* and *n* can be categorized into four components: *block value-to-block value*, *block value-to-index*, *index-to-index*, and *index-to-block value.* as shown in Eq. 1

$$\begin{aligned} S_{m,n} &= [C_{m,n}, Pos_{m|n}] \times [C_{n,m}, Pos_{n|m}]^T \\ &= C_m C_n^T + C_m Pos_{n|m}^T + Pos_{m|n} C_n^T + Pos_{m|n} Pos_{n|m}^T \end{aligned} \tag{1}$$

However, there have been recent improvements in the composition of DeBERTa owing to ELECTRA-Style Pre-Training [23]. The version, also known as DeBERTa-V3, has many variants, including DeBERTa-base, DeBERTa-V3-Small, DeBERTa-V3-XSmall, and mDeBERTa-V3-Base. The initial version of DeBERTa uses a mask language modeling (MLM) mechanism, which is now replaced by replaced token detection (RTD), considered to be a sample-efficient pre-training task. The variants of DeBERTa differ in their backbone parameters, vocabulary, hidden size, and layers. The architectural specifications of all the variants have been depicted in Table III. Once we input the data into the model, we now perform the task of Stratified K-Fold cross-validation.

TABLE III: Specifications of the different versions of DeBERTa

Model	**Corpus**	**Backbone Parameters(M)**	**Hidden Size**	**Layers**
DeBERTa-V3-Base	128	86	768	12
DeBERTa-V3-Small	128	44	768	6
DeBERTa-V3-XSmall	128	22	384	12
mDeBERTa-V3-Base	250	86	768	12

C. Stratified K-Fold Cross Validation

It is deemed essential to evaluate our model once trained on our input data. A methodological error is incorporated if the model retains the parameters of a periodic function and is experimented on the same data. The prediction scores would remain perfect on known labels, and the model's performance would still be unsatisfactory on unseen data. This condition is also known as overfitting. So to prevent overfitting, it is always deemed essential to split out a chunk of data into the test/validation set. However, there is a probability of overfitting the test/validation set due to tweaking the existing parameters until the estimator performs correctly. So to address this situation, we perform $K - Fold$ Stratified Cross-Validation [24]. The objective of the $K - Fold$ Stratified Cross-Validation is that the data is split into K folds. Training is performed on $K - 1$ folds while testing is performed on the remaining fold, resulting in a higher performance of the model. Mathematically, the cross-validation estimate CV can be represented in Eq. 2

$$CV = 1/N \sum_{i=1}^{N} L(y_i, f^{K_i}(x_i)) \tag{2}$$

where y_i depicts the actual score, $f^{K_i}(x_i)$ depicts the prediction the on K_ith fold and L is the loss function. Subse-

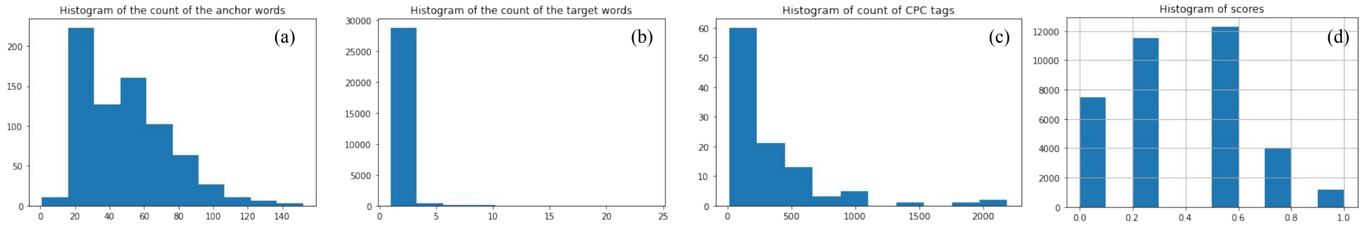

Fig. 2: Distribution of Terms in (a) Anchor Phrases, (b) Target Phrases, (c) Context Tags. (d) Distribution of scores

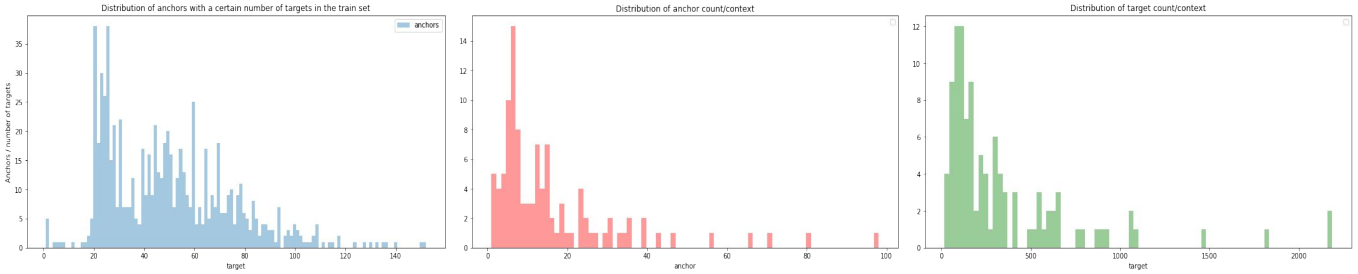

Fig. 3: (a) Distribution of anchors with respect to targets, (b) Distribution of anchor count with respect to context, (c) Distribution of target count with respect to context

quently, we evaluate the model using the Pearson correlation coefficient.

IV. Results and Experimentation

A. Dataset Used

This section deals with the description and exploratory data analysis of the dataset used. We use the U.S Patent Phrase to Phrase Matching dataset to perform semantic similarity analysis in this work. The U.S Patent Phrase to Phrase Matching dataset is derived from the repositories of the U.S. Patent and Trademark Office (USPTO), and its patent archives stand as a rare blend of information volume quality, and variety. The dataset consists of 4 columns: Anchor, Target, Context, and Score. The first phrase is represented by the anchor columns, the second by the target columns, and the context column represents the subject within which the similarity is to be scored. The dataset consists of 733 unique anchor words and 29340 unique target words. The frequency distribution of terms of anchor, target, and context columns is depicted in Fig. 2 (a), 2 (b) and 2 (c) respectively. The distribution of the score column is represented in Fig. 2 (d). Also, we analyze the distribution of anchor phrases with respect to context and target terms.Fig. 3 (a) shows the distribution of anchors with respect to targets, Fig. 3 (b), the distribution of anchor count with respect to context and Fig. 3 (c) depicts the distribution of target count with respect to context. Similarly, the character count and word count distribution of both anchor and target columns are illustrated in Fig. 4 (a) and 4 (b) respectively.

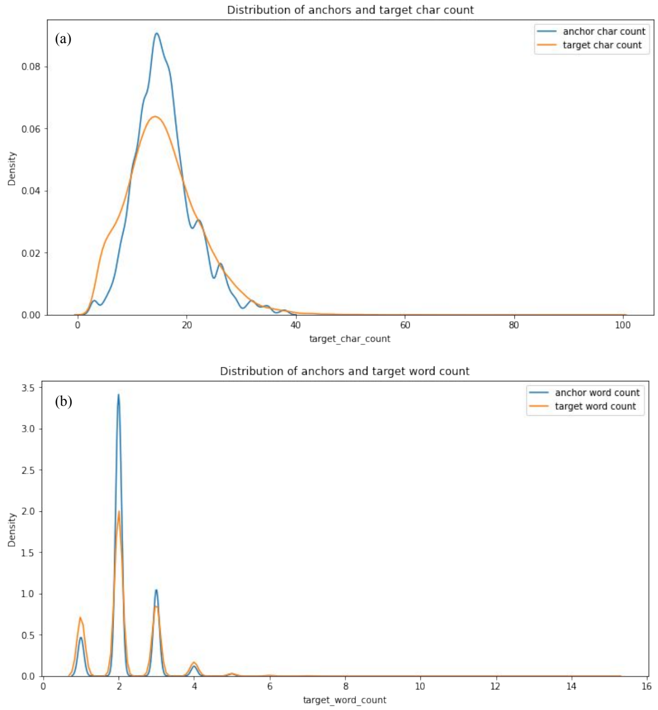

Fig. 4: (a) Distribution of Anchors and Target's character count (b) word count

B. Results of Semantic Analysis by Levenshtein Metric

The below Fig. 7 shows the distribution of the Levenshtein similarity score with respect to the number of anchor-target pairs. The figure shows that maximum pairs are present between 0.1 and 0.2, with the least number of pairs having the Levenshtein similarity score between 0.8 and 1.0. From this experiment, we conclude that the methodology yields us a pearson correlation score of 0.4147.

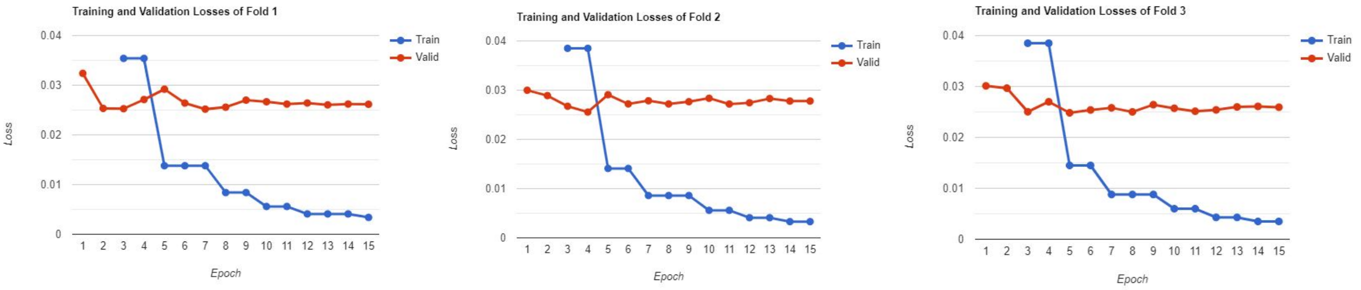

Fig. 5: Training and Validation Losses of DeBERTa-small in (a) Fold 1, (b) Fold 2 and (c) Fold 3

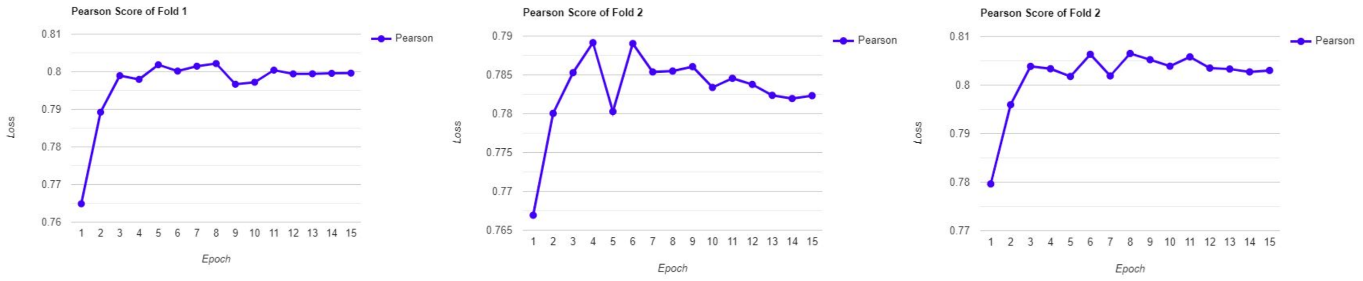

Fig. 6: Variation of pearson scores of DeBERTa-small in (a) Fold 1, (b) Fold 2 and (c) Fold 3

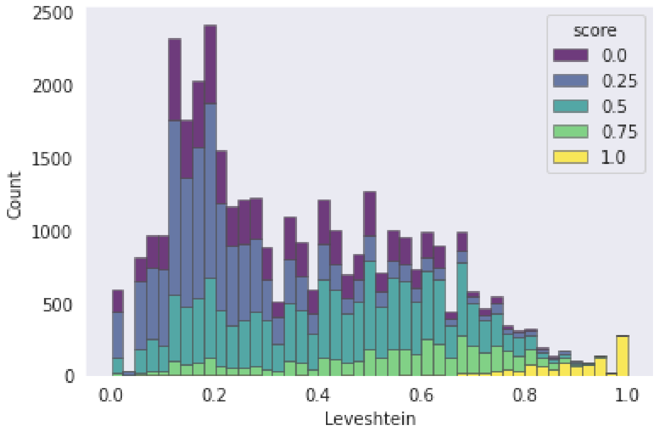

Fig. 7: Distribution of Levenshtein Similarity Scores

C. Performance of DeBERTa-base

In this section, we illustrate the performance of the DeBERTa-base model on the dataset. We use three metrics to evaluate the performance: training loss, validation, and Pearson correlation coefficient. The training loss measures how deep the model fits the training data and how accurate the model's predictions are on the training set. Similarly, we define the validation loss as the model's performance on the validation/test set. The final metric, also known as the Pearson correlation coefficient, gives us a linear measure of the strength of two variables. As depicted in Table IV, DeBERTa-base achieved an average training loss of 0.03, validation loss of 0.026, and Pearson correlation score of 0.74. In the subsequent sections, we analyze the performance of DeBERTa-V3-Small, DeBERTa-V3-XSmall, and mDeBERTa-V3-Base.

TABLE IV: Performance Metrics of DeBERTa-base

Fold	Training Loss	Validation Loss	Pearson Correlation
1	0.030000	0.024792	0.771547
2	0.030700	0.027380	0.760287
3	0.030500	0.024049	0.751671
4	0.029900	0.028247	0.785597

D. Performance of mDeBERTa

In this section, we emphasize the performance of multi-lingual DeBERTa on the training and validation sets. The number of epochs are 5, with the batch size being 128. Despite performing stratified K-Fold cross-validation with the number of folds set as 4, the model showed an inferior performance in terms of similarity prediction with very less Pearson coefficient score in all folds, as depicted in Table V. This can be justified by the fact that mDeBERTa is trained upon the CC100 multilingual data, and the presence of multilingual backbone parameters led to the depreciated performance of the model.

TABLE V: Performance Metrics of mDeBERTa

Fold	Training Loss	Validation Loss	Pearson Correlation
1	0.273200	0.276361	0.116614
2	0.148500	0.141006	0.154153
3	0.147500	0.140739	0.193211
4	0.150100	0.136254	0.175404

E. Performance of DeBERTa-Small

DeBERTa-Small is an abridged version of the DeBERTa-base, keeping in view the critical parameters required for prediction. It is trained on 160GB data as its previous version with 44M backbone parameters and has a hidden size of 768 with the number of layers as 6. The model achieved a cumulative Pearson score of 0.78 after three cross-validation folds. The performance metrics of DeBERTa-Small are depicted in Table VI. From Table VI, we can infer that the best performance is illustrated by DeBERTa-Small. Also, we illustrate the training and validation losses of each fold in Fig. 5 and the variation of the pearson score in Fig. 6.

TABLE VI: Performance Metrics of DeBERTa-v3-Small

Fold	Training Loss	Validation Loss	Pearson Correlation
1	0.003400	0.026166	0.799629
2	0.003300	0.027797	0.782329
3	0.003500	0.025930	0.803020

F. Performance of DeBERTa-XSmall

DeBERTa-XSmall is considered a simplified version of DeBERTa-Small with only 22M backbone parameters which is half in number compared to its counterpart. This model achieved a cumulative Pearson score of 0.765 after four cross-validation folds. However, fewer backbone parameters and hidden size justify its lesser performance than DeBERTa-Small. The performance metrics of DeBERTa-XSmall are depicted in Table VII.

TABLE VII: Performance Metrics of DeBERTa-XSmall

Fold	Training Loss	Validation Loss	Pearson Correlation
1	0.039200	0.030078	0.774637
2	0.039200	0.031391	0.765988
3	0.038800	0.029105	0.780142
4	0.038700	0.031934	0.755139

V. Conclusion

This paper experimented with traditional and transformer-based approaches for semantic similarity modeling on large corpora. We also compared our methodology with existing techniques, and the results demonstrated the improved performance of the model. The proposed methodology also illustrated context extraction and showed its importance in similarity modeling. In the following aspects, the execution time and memory could be optimized, thus leading to enhanced training. Also, the architecture of the existing model could be improved, thus leading to enhanced performance.

References

[1] D. Chandrasekaran and V. Mago, “Evolution of semantic similarity—a survey,” *ACM Computing Surveys (CSUR)*, vol. 54, no. 2, pp. 1–37, 2021.

[2] Y. Zhang, R. Jin, and Z.-H. Zhou, “Understanding bag-of-words model: a statistical framework,” *International Journal of Machine Learning and Cybernetics*, vol. 1, no. 1, pp. 43–52, 2010.

[3] S. Qaiser and R. Ali, “Text mining: use of tf-idf to examine the relevance of words to documents,” *International Journal of Computer Applications*, vol. 181, no. 1, pp. 25–29, 2018.

[4] H. Rubenstein and J. B. Goodenough, “Contextual correlates of synonymy,” *Communications of the ACM*, vol. 8, no. 10, pp. 627–633, 1965.

[5] G. A. Miller and W. G. Charles, “Contextual correlates of semantic similarity,” *Language and cognitive processes*, vol. 6, no. 1, pp. 1–28, 1991.

[6] L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin, “Placing search in context: The concept revisited,” in *Proceedings of the 10th international conference on World Wide Web*, 2001, pp. 406–414.

[7] E. Agirre, C. Banea, C. Cardie, D. Cer, M. Diab, A. Gonzalez-Agirre, W. Guo, I. Lopez-Gazpio, M. Maritxalar, R. Mihalcea *et al.*, “Semeval-2015 task 2: Semantic textual similarity, english, spanish and pilot on interpretability,” in *Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015)*, 2015, pp. 252–263.

[8] M. Marelli, S. Menini, M. Baroni, L. Bentivogli, R. Bernardi, and R. Zamparelli, “A sick cure for the evaluation of compositional distributional semantic models,” in *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC’14)*, 2014, pp. 216–223.

[9] Y. Benajiba, J. Sun, Y. Zhang, L. Jiang, Z. Weng, and O. Biran, “Siamese networks for semantic pattern similarity,” in *2019 IEEE 13th International Conference on Semantic Computing (ICSC)*. IEEE, 2019, pp. 191–194.

[10] Z. Li, H. Lin, W. Zheng, M. M. Tadesse, Z. Yang, and J. Wang, “Interactive self-attentive siamese network for biomedical sentence similarity,” *IEEE Access*, vol. 8, pp. 84 093–84 104, 2020.

[11] E. L. Pontes, S. Huet, A. C. Linhares, and J. Torres-Moreno, “Predicting the semantic textual similarity with siamese CNN and LSTM,” *CoRR*, vol. abs/1810.10641, 2018.

[12] Z. Quan, Z.-J. Wang, Y. Le, B. Yao, K. Li, and J. Yin, “An efficient framework for sentence similarity modeling,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 4, pp. 853–865, 2019.

[13] T. Shancheng, B. Yunyue, and M. Fuyu, “A semantic text similarity model for double short chinese sequences,” in *2018 International Conference on Intelligent Transportation, Big Data Smart City (ICITBS)*, 2018, pp. 736–739.

[14] T. Yang, S. Wu, J. Feng, N. Fu, and M. Tian, “Semantic network based approach to compute term semantic similarity,” in *2019 3rd International Conference on Electronic Information Technology and Computer Engineering (EITCE)*, 2019, pp. 654–658.

[15] P. R. Medi, P. Nemani, V. R. Pitta, V. Udutalapally, D. Das, and S. P. Mohanty, “Skinaid: A gan-based automatic skin lesion monitoring method for iomt frameworks,” in *2021 19th OITS International Conference on Information Technology (OCIT)*, 2021, pp. 200–205.

[16] Z. Liang and S. Zhang, “Generating and measuring similar sentences using long short-term memory and generative adversarial networks,” *IEEE Access*, vol. 9, pp. 112 637–112 654, 2021.

[17] Z. Li, H. Lin, C. Shen, W. Zheng, Z. Yang, and J. Wang, “Cross2self-attentive bidirectional recurrent neural network with bert for biomedical semantic text similarity,” in *2020 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, 2020, pp. 1051–1054.

[18] M. M. Sanjeev, B. Ramalingam, and S. Kumar T.K., “Realtime semantic similarity analysis of bulk outlook emails using bert,” in *2020 International Conference on Advances in Computing, Communication Materials (ICACCM)*, 2020, pp. 89–94.

[19] R. Haldar and D. Mukhopadhyay, “Levenshtein distance technique in dictionary lookup methods: An improved approach,” *arXiv preprint arXiv:1101.1232*, 2011.

[20] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” *Advances in neural information processing systems*, vol. 30, 2017.

[21] P. He, X. Liu, J. Gao, and W. Chen, “Deberta: Decoding-enhanced bert with disentangled attention,” in *International Conference on Learning Representations*, 2021.

[22] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, “Roberta: A robustly optimized bert pretraining approach,” *arXiv preprint arXiv:1907.11692*, 2019.

[23] P. He, J. Gao, and W. Chen, “Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing,” *arXiv preprint arXiv:2111.09543*, 2021.

[24] T.-T. Wong and N.-Y. Yang, “Dependency analysis of accuracy estimates in k-fold cross validation,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 11, pp. 2417–2427, 2017.