Analysis of Semantic Similarity between Sentences Using Transformer-based Deep Learning Methods

Igor Onyshchenko, Anatoly Anisimov, Oleksandr Marchenko and Mariam Isoieva

Taras Shevchenko National University of Kyiv, 60, Volodymyrska Str., Kyiv, 01033, Ukraine

Abstract

This paper presents an analysis of modern Transformer-based approaches to the semantic modelling of words and sentences. It covers the research and design of semantic similarity and paraphrase identification methods, as well as experimental evaluation of their performance.

Metric learning approach and Transformer-based models are analysed as a basis for possible applications for solving tasks related to semantic similarity estimation. Experimental results for Siamese and triplet networks are presented along with a comparison of various aggregation functions.

Experiments demonstrate that the considered deep language models based on the Transformer architecture can be used to obtain efficient latent words' features and to analyse their connections within a sentence and links between sentences. The proposed combined approach, which is based on using the BERT-like models fine-tuning, has shown significant improvements to the various popular strategies.

Keywords [1]

Deep Language Model, Metric Learning, Attention Mechanism, Neural Networks, Paraphrasing, Semantic Similarity

1. Introduction

Analysis of semantic similarity of sentences is one of the main tasks in the field of Natural Language Processing. It is crucial for clustering, information retrieval, summarization, plagiarism detection, etc. In general, the task of measuring semantic similarity consists in assigning some value, which represents similarity, to a pair of sentences. In this paper, we mostly consider the task in a binary setting, in other words, we try to solve the problem of identifying whether two sentences are semantically identical, i.e., are paraphrases. The considered solution is based on some continuous measure of similarity and some threshold value. The problem of semantic similarity analysis and identification is usually considered as a task of classification or logistic regression. Although the task of paraphrase identification is formulated in semantic terms, approaches to solving this problem are often based on statistical classifiers that use shallow lexical and syntactic features.

Usually, models based on bag-of-words, n-grams, and TF-IDF [1] are used to form a representation of a sentence for such approaches, followed by some methods of similarity estimation (such as Levenstein's editing distance, longest common substring, Jaccard coefficient, and cosine distance) [1] for measuring similarity between two sentences. However, paraphrasing is usually done by replacing words with their synonyms/antonyms, syntactic modifying, shortening the sentences, combining, reorganizing, mixing words, generalizing the mentioned concepts, which allows changing the original text, while maintaining the semantics of the sentence. This fact makes such approaches inefficient.

Other approaches leverage the usage of syntactic features, i.e., take into account the structure of the sentence [2, 3], with an assumption that similar sentences have similar syntactic structures. However, this assumption cannot solve the problem of "the same semantics but different syntactic structures".

Information Technology and Implementation (IT&I-2022), November 30 - December 02, 2022, Kyiv, Ukraine
EMAIL: igorko323@gmail.com (A. 1); anatoly.v.anisimov@gmail.com (A. 2); omarchenko@univ.kiev.ua (A. 3), isoyevamaryam@gmail.com (A. 4)
ORCID: 0000-0002-1467-2006 (A. 2); 0000-0002-5408-5279 (A. 3)


In recent years, traditional methods for modeling the semantics of words and sentences have given way to approaches based on artificial neural networks. Such models can learn to find hidden features and, most importantly, to identify the relationships between both words and sentences, which is crucial for the task of measuring semantic similarity. Deep neural models, in contrast to traditional approaches, model the contextual representation of words; this means that two identical words, but used in different contexts, have different vector representations. English sentences and the relationship of semantic similarity between them are the object of this research. Research methods and tools include deep learning natural language models based on the Transformer architecture, some deep learning methods, and text data corpora for neural network training.

The following section presents an overview of Transformer-based models, their limitations, and opportunities for using them to solve tasks related to semantic similarity estimation. Section 3 covers aspects of semantics modeling, methods of sentence representations formation, and usage of metric learning approach for similarity measuring. Experimental results are presented in Section 4.

2. Deep language models

2.1. Transformer deep neural network architecture

The Transformer architecture proposed in [4] has become, in a sense, a golden standard in the modern field of natural language processing. The ideas and approaches described in the original paper are the basic elements for numerous modern deep learning language models.

However, the mechanism of attention described by the authors is considered to be the most significant contribution. The attention function can be specified as a mapping of input data consisting of queries and sets of key-value pairs to a set of outputs, where queries, keys, values, and the output are represented as vectors. The result is calculated as a weighted average of the values, where the weight assigned to each value is set using the relevance function for the query and the corresponding key.

The attention mechanism proposed by the authors is called "Scaled Dot-Product Attention", it takes queries, dimension keys d_k, and dimension values d_v as an input. The attention function is calculated for a set of queries simultaneously, which form a matrix Q. The keys and values are also represented by some matrices K and V. The output matrix is calculated as follows:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

For large values of d_k, the absolute value of the scalar product of the query and key vectors becomes greater, displacing the softmax function [5] in the parts where it has very small gradients. To overcome this effect, the scalar product is scaled by $\frac{1}{\sqrt{d_k}}$.

For the proposed architecture, not only the attention function is calculated with the keys, values, and dimensional queries d_{model}, but the queries, keys, and values are projected onto the dimensions d_k, d_k, and d_v correspondingly with different trained linear projections for h times. After that, the attention function is computed in parallel for all the projected queries, keys, and values. Then the results are concatenated, there is another linear projection and, as a result, the final values are obtained. The described mechanism is called the multi-head mechanism of attention. It allows models to receive information from different subspaces in different positions simultaneously.

2.2. BERT-based models

The authors of [6] presented the language model BERT (Bidirectional Encoder Representations from Transformers). The BERT architecture consists of a multilayer bidirectional Transformer encoder. BERT is designed for pre-training of deep bidirectional word embeddings by considering the right and left contexts in all layers of the model. Such representations are then used for the final training in specific NLP tasks. Recent empirical improvements, based on transfer learning for language models, have shown that prior unsupervised training is an integral part of many natural language understanding systems. BERT is an example of a pre-trained model with a bidirectional architecture that can successfully solve a wide range of problems. The BERT model has catalyzed the emergence of new

models based on its architecture and the principles proposed in the original article. Training of such models requires significant computational resources, huge corpora of text data and many hours of experiments for successful selection of hyperparameters. The authors of [7] researched the impact of all these factors on the effectiveness of the original BERT model and proposed an approach to model training based on its architecture, which they called RoBERTa (Robustly optimized BERT approach).

This approach is based on relatively simple modifications: longer training time with larger batch size and bigger text corpus; omission of the NSP target function; training on longer input sequences; dynamic masking of the training data. A new large CC-News text corpus has also been proposed to control the effects of training data size. It is worth mentioning that the authors of RoBERTa increased the size of the token dictionary from 30 000 to 50 000, using a slightly modified version of tokenization. Thus, the implementation of all the mentioned techniques in RoBERTa helped to improve the efficiency of the original BERT model. Model size increase during pre-training often improves the accuracy for specific tasks. However, at some point, further enlargement of the model becomes more difficult due to the limitations of memory in graphics cards and training time. To overcome these limitations, the authors of the ALBERT model (A Little BERT) [8] proposed two techniques that reduce the number of parameters of the BERT model and speed up the training process, while maintaining and even improving its accuracy: factorization of the embeddings' weights matrix and usage of the same parameters in different layers. Both techniques significantly reduce the number of parameters of the BERT model but maintain its accuracy and increase the efficiency of each parameter. These techniques also play the role of additional regularization, which stabilizes training and improves generalization.

3. Sentence semantics modeling

3.1. Problem setting

For modeling the sentence semantics, we consider the construction of a finite-dimensional vector space $S \subseteq R^n$, in which each sentence is represented by some element. The main feature of the resulting space is that two semantically similar sentences are close to each other in terms of some similarity measure. Thus, it would be sufficient to compare the embeddings of each sentence using the defined similarity measure to identify semantic similarity. Figure 1 shows the general scheme of constructing vector representations for two sentences with their subsequent comparison.

For the construction of this space, we use contextual word embeddings obtained by using deep language models based on Transformer, namely BERT, RoBERTa and ALBERT. We use cosine similarity as a similarity measure. We also propose some other measures based on neural networks. Apart from that, to obtain the desired property of the space (semantically similar sentences being close), we consider certain models' training strategies, which intensify this discriminatory effect.

3.1. Deep language models’ outputs aggregation

We use the aforementioned Transformer-based language models as a basic element for modeling sentence semantics. Each such model takes a sequence of words, which form a sentence, as an input. The attention mechanism, which is fundamental for such a model, is used for building contextual representations of each word. On the one hand, this representation encapsulates the statistical features of the word within a particular language and, on the other hand, it takes into account the word’s role in the context of the sentence. Unlike basic recurrent models, the Transformer encoder unit, which is the basis of BERT, RoBERTa and ALBERT, processes both the left and the right context of each word.

Thus, the deep language model transforms the input sequence of words into a sequence of contextual representations of the corresponding words. To obtain the sentence embedding, we need to aggregate this sequence of words' embeddings. Obviously, sentences, in general, have different lengths, so it is important that the vector representation of each sentence has some fixed size, which will determine the dimensionality of the vector space of sentences. Averaging of all the contextual word embeddings is the most obvious option (here, 'MEAN' denotes this method). In this case, the dimensionality of the sentences space is the same as the dimensionality of the words space. For this aggregation method, each word is equally important for constructing a sentence representation. The maximum function (MAX) is another method of aggregation, for which the largest value of the vector components is taken.

A special CLS token was introduced in the original BERT article. This token is always added in front of the input sequence. The vector representation corresponding to the CLS token is used as an embedding of the whole sentence. For BERT and other models, this embedding is used for the subsequent fine-tuning for specific tasks. We also consider this aggregation method.

3.1. Similarity measure based on a neural network

Despite the widespread popularity of cosine similarity for different tasks of natural language processing, other sentence similarity measures are also worth considering. Today, neural networks are a powerful and effective tool in many applications, so in the scope of this paper, we also analyze and build some methods for similarity measuring based on neural networks.

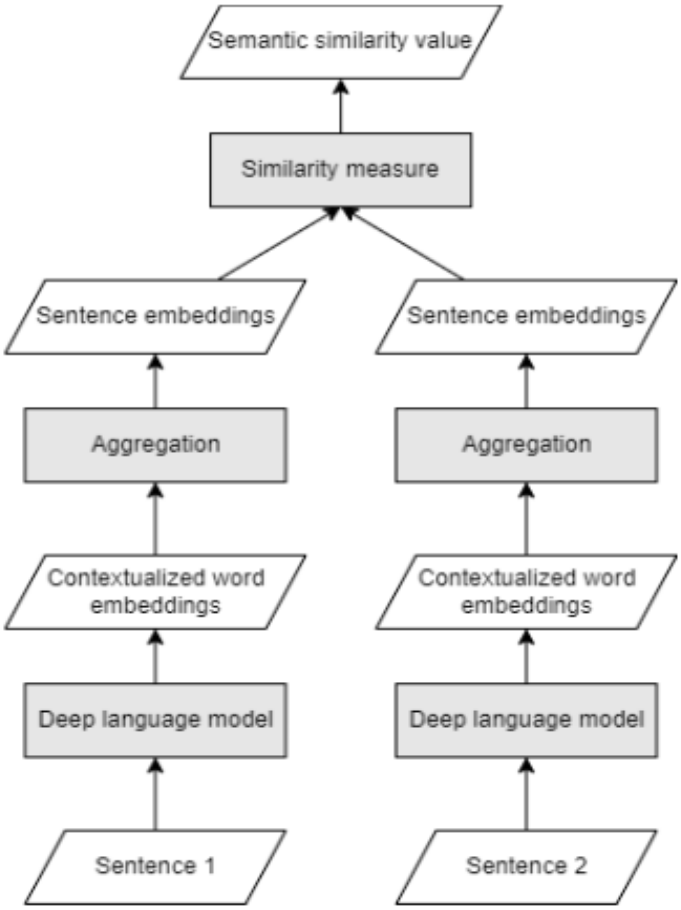

Figure 1: Semantic similarity identification scheme

In general, measuring the degree of similarity of two sentences can be reformulated into a regression or classification problem. The regression problem consists in finding a real number, which corresponds to the similarity value. And the classification problem is considered when some class label is assigned to a pair of sentences, e.g., for the task of paraphrase detection, where there are two classes. Binary classification is often formulated as the problem of logistic regression with the subsequent introduction of some threshold value. Thus, having two sentence embeddings, we will solve the problem of logistic regression to identify their semantic similarity. Since logistic regression is a special case of classification, here, we name the neural network for similarity measuring a classifier.

Some properties of the classifier required for practical applications are the following. First of all, the number of classifier parameters must be much smaller than that of the language model. The required computation time and the amount of memory are also important.

A 'heavy' and slow classifier will be inefficient in practice. Two sentences' embeddings, a and b, are the input for the classifier. The next step is the transformation of these vectors into a single representation that is fed to the neural network. Authors of [9] proposed to use the combined vector of

the a and b together with the results of operations on them, such as component-wise summation, subtraction, multiplication, etc. E.g., $(a, b, |a - b|, a * b)$ is one of the possible input vectors.

In this paper, we consider the effectiveness of different representations. The resulting combined vector representation of the sentences is then fed to the input of an artificial neural network. For the case of logistic regression, the output of the neural network is always one number, i.e., a one-dimensional vector. To get the similarity measure value, we use a sigmoid, which translates its argument into a number from the interval $(0; 1)$. We also use a specific threshold value to identify the class.

3.2. Metric learning. Siamese and triplet networks

Neural network training consists in the optimization of a specific objective function, also called the loss function. The goal is usually to minimize the network's inference error with respect to the correct label of the input object. For identifying semantic similarity, we need to evaluate the relative distance or similarity between the two input sentences. This training strategy is widespread in other areas of artificial intelligence, where there is a need to compare different objects. This concept is called metric learning, and the target function used in this case is called contrastive loss.

The task of identifying the semantic similarity of sentences conforms to the principles of metric learning. We first get the sentences' embeddings by using a deep language model. Then we choose the comparison function, i.e., the similarity function. After that, we train a deep language model to generate similar vector representations for semantically similar sentences and distant vectors for dissimilar ones.

There are two general approaches to constructing the contrastive loss function: using pairs of training objects (pairwise loss function) and using triplets of training objects (triplet loss function).

3.2.1. Pairwise loss function. Siamese neural network

The first approach involves the usage of positive and negative pairs of training objects. Positive pairs are formed from an anchor object x_a and a positive object x_p that is close to x_a in terms of the defined similarity measure, and negative pairs are formed by an anchor object x_a and a negative object x_n that is dissimilar to x_a. The objective of the training is to form vector representations, for which the distance is as small as possible for the positive pairs and is greater than a certain margin for the negative pairs. Let r_a, r_p and r_n denote the corresponding embeddings of the input objects, then the pairwise loss function can be formulated as follows:

$$L = \begin{cases} d(r_a, r_p) & if \quad PositivePair \\ max(0, m - d(r_a, r_n)) & if \quad NegativePair \end{cases}$$

The pairwise loss function is often used when embeddings are formed using identical deep neural networks with common weights. In this case, training is based on *Siamese neural networks*.

3.2.2. Triplet loss function. Triplet neural networks

The contrastive loss function based on triplets of training objects in many cases shows better results than the pairwise loss function. Triplets are formed by the anchor x_a, positive object x_p, and negative object x_n. This approach is based on the fact that the distance between the anchor and the negative object is greater (with a margin of m) than the distance between the anchor and the positive object. In the general case, the triplet loss function is formulated as follows

$$L(r_a, r_p, r_n) = max(0, m + d(r_a, r_p) - d(r_a, r_n))$$

Like the pairwise loss function, the triplet loss function is most often used when embeddings are obtained using identical deep neural networks with shared weights. In this case, training is based on the *triplet neural networks*.

3.3. Fine-tuning of the BERT-like language models

The main purpose of BERT is its further fine-tuning for a specific NLP task. BERT is a pre-trained model with a built-in attention mechanism, which is used to extract certain features of words and to

model the relationships between them within a sentence. And it is the process of fine-tuning that allows one to accurately interpret these features and relationships to solve specific problems. This approach demonstrates the best results for different tasks on a variety of corpora. This applies to the problems of semantic similarity and paraphrase identification. Although this approach works well for the pairs of input sentences, it requires both sentences to be fed together to the model. This implies significant computational costs. Authors of [10] investigated that finding the most semantically similar pair among $n = 10000$ sentences requires $n \cdot (n-1)/2 = 49995000$ runs of the BERT model. This task takes approximately 65 hours on a modern V100 graphics card. Such a long search time is inadequate for practical applications in an era when the volume of new information is growing exponentially.

Thus, despite the high accuracy of the BERT model after its fine-tuning for a specific task, this approach cannot be applied to real-world problems such as semantic similarity identification, clustering, information retrieval for the huge streams of input objects. That is why in this paper, we consider methods of constructing a semantic representation of each sentence separately, which can be used for their efficient comparison. However, the method of simultaneous training for a pair of sentences as a single input sequence, which was proposed in BERT, can help to improve the results.

4. Experiments

4.1. Training experiments setting

For training and testing the proposed approaches, we use Microsoft Research Paraphrase Corpus (MRPC), which is classic for the analysis of semantic similarity of sentences [11].

The task of paraphrase identification, as mentioned earlier, is a binary classification problem. In this case, the most popular choices for model quality assessment metrics are accuracy and F1-score. The latter allows one to get a better evaluation of the model quality when the corpus is not balanced, as for MRPC. For paraphrase identification based on metric learning and logistic regression, we need to select a threshold for classification. For fair assessment of the model performance on the test corpus and its ability to generalize, the choice of the threshold value is made using cross-validation.

Python programming language and PyTorch framework have been used for the development of the software system for training experiments. All the experiments have been performed on an Nvidia GeForce GTX 1060 graphics card with 6GB of video memory. HuggingFace library has been used for accessing the deep language models and corresponding weights. Due to memory limitations, the following model versions have been selected for the experiments: $BERTbase$, $RoBERTabase$, $ALBERTbase$. For all experiments, the batch size is 12. Each model has been trained for 30 epochs, each of 80 iterations. For the classification and fine-tuning, the additional block (head) of the neural network has been trained during the first 10 epochs with "frozen" weights of the language model.

A stochastic gradient descent with a Nesterov momentum with a parameter of 0.9 has been chosen as the optimizer. The initial value of the learning rate depends on the specific model and target function, but the overall strategy for changing the step is to reduce it by a factor of 0.5 every 5 epochs.

L2 regularization with parameter 0.01 has been chosen for regularization of the model. For Siamese and triplet networks, the *m* value has been set to 0.5 for all the experiments.

4.2. Results and analysis of the training experiments

4.2.1. Initial results of the language models

The initial results of the models with pretrained weights for different types of outputs aggregation are presented in the Table 1, to give better understanding of how efficient the training and different target functions are. Cosine similarity has been used as a measure of similarity. As we see, types of aggregation MEAN and CLS show slightly better metrics' values.

4.2.1. Metric learning

Deep language models are pre-trained to extract a diverse set of features from the input word sequence. To solve a particular problem, one needs to interpret these features correctly, assign bigger weights to some of them and discard others. Metric learning allows models to better use certain features

for semantic similarity identification. Table 2 shows the results of models' training in the paradigm of the Siamese networks. An improvement for all the models and types of aggregation can be noticed, comparing to the results in the Table 1. Table 3 shows the results of training using a triplet networks-based approach. Based on the comparison with the Table 2, the training strategy based on the triplet networks shows much better results than the one based on Siamese networks. This is also confirmed by research in other fields of artificial intelligence.

Table 1
Initial results (accuracy/F1) of the language models

	MEAN	MAX	CLS
BERT	0.659/0.789	0.659/0.785	0.664/0.797
RoBERTa	0.681/0.791	0.661/0.788	0.664/0.797
ALBERT	**0.690**/0.788	0.683/0.792	0.665/**0.798**

Table 2
Results (accuracy/F1) of the Siamese networks

	MEAN	MAX	CLS
BERT	0.692/0.798	0.684/0.789	0.697/0.801
RoBERTa	0.693/0.797	0.686/0.790	0.701/0.799
ALBERT	0.691/0.799	0.684/0.787	**0.702/0.811**

Table 3
Results (accuracy/F1) for the triplet networks

	MEAN	MAX	CLS
BERT	0.709/0.809	0.692/0.795	**0.729/0.818**
RoBERTa	0.709/0.807	0.691/0.799	0.720/0.812
ALBERT	0.694/0.793	0.676/0.797	0.715/0.795

4.2.2. Logistic regression

Several types of combinations of sentences' vector representations have been analyzed, together with variants of classification networks with different parameters and numbers of layers, to identify semantic similarity using the neural network-based similarity measure.

Based on the results of numerous experiments, which did not show significant differences in the use of many hidden layers, we decided to choose a network with one input layer. We chose binary cross entropy as a loss function. In order to choose the type of embeddings aggregation, we conducted experiments with the BERT model and the CLS aggregation type. The results and types of combinations are shown in Table 4, where u, v are vector representations of sentences, operation " $*$ " denotes the element-wise vector multiplication.

Table 4
Results (accuracy/F1) for different types of aggregation

Type of aggregation	BERT(CLS)
(u, v)	0.671/0.782
(|u - v|)	0.689/0.791
*(u * v)*	0.698/0.795
(u, v, |u - v|)	0.722/0.812
(u, v, u v)*	0.719/0.809
*(u, v, |u - v|, u*v)*	**0.727/0.814**

The best results have been obtained for aggregation *(u, v, |u - v|, u*v)*, which is used for the subsequent experiments. Table 5 shows the results of logistic regression training for all the models and types of aggregation. Comparing the results obtained for triplet networks and logistic regression, we see that the use of the neural network-based measure of similarity shows much better results, especially when using the RoBERTa and ALBERT models. It is also worth noting that sentence embeddings, which have been obtained from logistic regression training, can be efficiently compared using cosine similarity

Table 5
Results (accuracy/F1) for the logistic regression

	MEAN	MAX	CLS
BERT	0.712/0.811	0.709/0.802	0.727/0.814
RoBERTa	0.782/0.839	0.762/0.829	**0.809/0.860**
ALBERT	0.770/0.837	0.751/0.823	0.769/0.843

4.2.3. Conclusions on the aggregation type

Analyzing the outcomes of the experiments, we conclude that the MAX aggregation type shows the worst results. We can assume that the maximum function is quite 'aggressive' and discards some important generalized properties. MEAN and CLS show quite similar results, which motivates the usage of one of these aggregation types for sentence embeddings. Following the original BERT paper and taking into account the mentioned results, we use the CLS aggregation type for the subsequent experiments.

4.2.4. Models fine-tuning

As mentioned above, BERT-like models have certain pre-trained weights. The model must be fine-tuned for any task. This also applies to the problem of paraphrase identification, when two sentences are combined into a single input sequence. Fine-tuning usually increases the quality of results, but this approach is impractical due to significant computational costs. However, in Table 6, we present the results of our own fine-tuning and the results given by the authors in the original articles. It is worth noting that our results have been obtained for the base versions of all models. The results of RoBERTa, presented in the article, are given for the large version and of ALBERT - for the xxlarge version. Both of these versions have significantly more parameters than the base version.

Table 6
Results (accuracy/F1) after the fine-tuning

	Obtained result	Presented in the article
BERT	0.864/0.898	-/0.889
RoBERTa	**0.885/0.915**	-/0.909 (large)
ALBERT	0.857/0.894	-/0.909 (xxlarge)

4.2.5. Combined approach

All the considered models have a certain initial set of weights, used as the starting point for further training. Despite the inability to apply the approach of semantic similarity identification, which was proposed in the original finetuning method, i.e., combining two sentences into one sequence, which is used as an input to the language model, we can use the weights obtained in the process of such tuning as the initial ones for the aforementioned metric training and logistic regression. We will call this two-step method of training a combined approach. Table 7 presents the results of the language models that were initially fine-tuned (Table 6), using the considered training strategies and the type of aggregation CLS, where Baseline is the starting result of the models with cosine similarity, Siam - Siamese learning strategy, Triplet - triplet networks learning, LR - logistic regression.

Table 7
Results (accuracy/F1) based on the combined approach

	Baseline	Siam	Triplet	LR
BERT	0.708/0.790	0.751/0.828	0.743/0.825	0.796/0.851
RoBERTa	0.729/0.807	0.721/0.811	0.725/0.817	**0.798/0.855**
ALBERT	0.692/0.799	0.730/0.822	0.724/0.819	0.779/0.843

The results show that model fine-tuning helps to improve results when used with further training strategies. Thus, the fine-tuning approach, proposed by the authors of the original BERT model for

tasks on the level of pairs of sentences, can also be suitable for sentence semantics modeling, i.e., in constructing a vector space of sentences, in which semantically similar sentences are represented by the close elements (in terms of some similarity measure). The obtained training results show that the standard usage of the outputs of the considered language models for getting the sentence embeddings is inefficient. The strategy of metric learning and logistic regression can be used to obtain better, more 'discriminative' sentence embeddings for more accurate semantic similarity identification.

5. Conclusions

In this paper, efficient approaches to the analysis of semantic similarity of sentences using deep learning methods were proposed and investigated. The considered deep language models based on the Transformer architecture can be used to obtain efficient latent words' features and to analyse their connections within a sentence, as well as connections between sentences. However, to model the semantics of a sentence, i.e., to construct a vector space where semantically similar sentences represented by points in that space are close, we need an appropriate interpretation of the features and connections provided by the language model. We researched and applied corresponding training strategies, which allowed us to obtain more discriminative features. Metric learning based on Siamese and triplet networks with a cosine degree of similarity allowed to improve the initial results of the considered language models. Logistic regression as a comparison measure has shown that this approach is very promising.The proposed combined approach, which is based on using the BERT-like models fine-tuning, has demonstrated significant improvements to the previously discussed training strategies. The results indicate the efficiency of the proposed and researched approaches.

6. References

[1] Jurafsky, D., Martin, J. H.: Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. 2nd edn. Pearson Education (2009).

[2] Elhadi, M., Al-Tobi, A.: Duplicate detection in documents and webpages using improved longest common subsequence and documents syntactical structures. In: 2009 Fourth International Conference on Computer Sciences and Convergence Information Technology, pp. 679-684. IEEE (2009).

[3] Elhadi, M., Al-Tobi, A.: Use of text syntactical structures in detection of document duplicates. In: 2008 Third International Conference on Digital Information Management, pp. 520-525. IEEE (2008).

[4] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Polosukhin, I. (2017). Attention is all you need. In: Advances in Neural Information Processing Systems, vol. 30. Curran Associates (2017).

[5] Bishop, C. M.: Pattern recognition and machine learning. Springer, New York (2006).

[6] Devlin, J., Chang, M. W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, vol. 1, pp. 4171–4186. Association for Computational Linguistics, Minneapolis, Minnesota (2019).

[7] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Stoyanov, V. (2019). RoBERTa: A robustly optimized BERT pretraining approach. arXiv preprint, http://arxiv.org/abs/1907.11692.

[8] Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., Soricut, R. (2019). Albert: A lite BERT for self-supervised learning of language representations. In: Proceedings of the Eighth International Conference on Learning Representations. ICLR (2020).

[9] Conneau, A., Kiela, D.: SentEval: An evaluation toolkit for universal sentence representations. In: Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018). European Language Resources Association (ELRA), Miyazaki, Japan (2018).

[10] Reimers, N., Gurevych, I.: Sentence-BERT: Sentence embeddings using Siamese BERTnetworks. 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), pp. 3982–3992. Association for Computational Linguistics, Hong Kong, China (2019).

[11] Dolan, W., Brockett, C.: Automatically Constructing a Corpus of Sentential Paraphrases. In: Proceedings of the Third International Workshop on Paraphrasing (IWP2005). Association for Computational Linguistics (2005).