Zhestyatsky at SemEval-2021 Task 2: ReLU over Cosine Similarity for BERT Fine-tuning

Boris Zhestiankin
Moscow Institute of Physics and Technology
Moscow, Russia

Maria Ponomareva
ABBYY
HSE University
Moscow, Russia

zhestyankin.ba@phystech.edu maria.ponomareva@abbyy.com

Abstract

This paper presents our contribution to SemEval-2021 Task 2: Multilingual and Cross-lingual Word-in-Context Disambiguation (MCL-WiC). Our experiments cover English (EN-EN) sub-track from the multilingual setting of the task. We experiment with several pre-trained language models and investigate an impact of different top-layers on fine-tuning. We find the combination of Cosine Similarity and ReLU activation leading to the most effective fine-tuning procedure. Our best model results in accuracy 92.7%, which is the fourth-best score in EN-EN sub-track.

1 Introduction

The increasing progress in Natural Language Processing is closely related with development of word representations. The context-independent word embeddings, such as word2vec (Mikolov et al., 2013) and fastText (Bojanowski et al., 2017) brought the idea of measuring the relatedness of the meanings as the distance between the vectors encoding them. The introduction of the methods of pre-training context dependent embeddings, such as ELMo (Peters et al., 2018), ULMFit (Howard and Ruder, 2018), and BERT (Devlin et al., 2018) made the next crucial breakthrough overcoming the shortcomings of previous methods to encode the meaning. Despite the fact that the primal objective of word embeddings is to encode the meaning of words, it is not obvious how to evaluate them directly. While common manner to examine the superiority of particular type of embeddings is to look at their performance on some downstream tasks, the more direct way to evaluate their ability to represent semantic is challenging.

SemEval-2021 Task 2: Multilingual and Cross-lingual Word-in-Context Disambiguation (MCL-WiC) (Martelli et al., 2021) presents a new framework to evaluate embeddings. In this paper we present our contribution for the task. We explore the potential of different pre-trained context-dependent embeddings based on pre-trained language models. We find that the Cosine Similarity can produce fruitful results when used for fine-tuning the weights of the pre-trained models, while adding linear layers to learn the similarity from the limited data leads to instant overfitting.

2 Background

The traditional approach to evaluate the ability of embeddings to catch the meaning of words is Word Sense Disambiguation (WSD) task (Navigli, 2009). WSD is defined as classification problem, when a given word is classificated between its predefined senses. WSD by design comes with an important limitation, being connected directly with predefined sense inventories such as WordNet[1] (Fellbaum, 2005).

The Word in Context (WiC) benchmark (Pilehvar and Camacho-Collados, 2019) addresses these limitations. The task proposes a binary classification setting for English, when, given two sentences s_i and s_k and two words w_i and w_k in them, the system needs to decide whether the word w_i in s_i and w_k in s_k have same or different meanings. The main advantage of WiC task is a possibility to expand its consideration to the languages that lack such sense inventories.

MCL-WiC extends the WiC approach to new senses and new languages, covering data in five languages: Arabic, Chinese, English, French and Russian. The task provides data of two types: in the multilingual setting one needs to predict the label to the pair of sentences in one language (AR-AR, ZH-ZH, EN-EN, FR-FR, RU-RU sub-tracks), in the cross-lingual setting the first sentence is in English and the second one is in one of the four

[1] https://wordnet.princeton.edu

Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021), pages 163–168
Bangkok, Thailand (online), August 5–6, 2021.

other considered languages (EN-AR, EN-ZH, EN-FR, EN-RU sub-tracks).

After preliminary experiments we decided to focus our efforts on the only sub-track with training data, namely the English sub-track from the multilingual setting. Our solution[2] is fourth placed in the EN-EN leaderboard with 92.7% accuracy and is 0.6% behind the winner.

3 System overview

Approaching the task we conduct multiple experiments with a variety of architectures, however all of them are deeply based on contextual embeddings fine-tuning. For our experiments we use pre-trained embeddings from BERT and XLM-RoBERTa (Conneau et al., 2020) models and fine-tune them for our task.

3.1 Target word embeddings

Design of BERT and XLM-RoBERTa models assumes that text is first split to tokens and embeddings for these tokens are evaluated. Therefore we define our technique to obtain the embeddings, representing target words in the sentences.

For a single sentence we take embeddings of all sub-tokens corresponding to the target word in it and max pool them into one embedding. Repeating this procedure for both sentences in each pair we obtain two embeddings as the result: first — corresponding to target word in the first sentence and second — corresponding to target word in the second sentence.

3.2 Multilayer Perceptron Architecture

In our initial setup we build a system based on Multilayer Perceptron neural network. The purpose of this approach is to train the system to predict that target words have the same meaning in both sentences.

This model calculates embeddings of the target word in both sentences of the pair and concatenates them together, taking the result as an input layer. The model contains one hidden layer with 100 neurons, ReLU activation before it and an output layer, activated by sigmoid.

Interpreting the model output as the probability that target words have the same meaning in both of the sentences, we predict True if the output turns out to be greater than 0.5 or we predict False otherwise.

To enrich the knowledge of the model about the task we also experiment with a slightly different input, making use of [CLS] tokens. Each [CLS] token represents the whole sentence. Taking [CLS] tokens embeddings for each sentence in a pair we concatenate them together and afterwards concatenate the result with an input layer (consisting of target word embeddings concatenation) defined above. We use the resulting embedding as an input layer for our model and do not change other parameters in the setup.

3.3 Cosine Similarity Architecture

As an alternative to Multilayer Perceptron approach we define a Cosine Similarity approach, illustrated on Figure 1. which proves to be our best system for the task. The purpose of this approach is to train the system to predict the probability that the target word has the same meaning in both sentences.

During training our system takes embeddings of the target word in each sentence in a pair and calculates Cosine Similarity between them. It activates the similarity through ReLU layer. The result value is considered the output of the model.

After the training is finished we have to make predictions, which is achieved by defining the probability threshold as a hyperparameter. In this way we predict True if the output of the model is greater than the threshold or False otherwise.

To maximize the accuracy of the model we calculate the probability threshold by building the Receiver Operating Characteristic (ROC) curve and choosing the value corresponding to the maximum difference between true positive and false positive rates.

We note that in this approach no new weights are introduced in contradistinction to Multilayer Perceptron approach. Therefore only pre-trained weights of BERT and XLM-RoBERTa models are fine-tuned.

To provide a comparison option for Cosine Similarity approach we also try applying sigmoid as an activation instead of ReLU.

3.4 Datasets

Speaking about the datasets for training and validation we fully utilize train and development English data provided by the competition organisers for the EN-EN sub-track. However, to achieve the

[2] Source code, experiments, requirements and results can be found at https://github.com/zhestyatsky/MCL-WiC

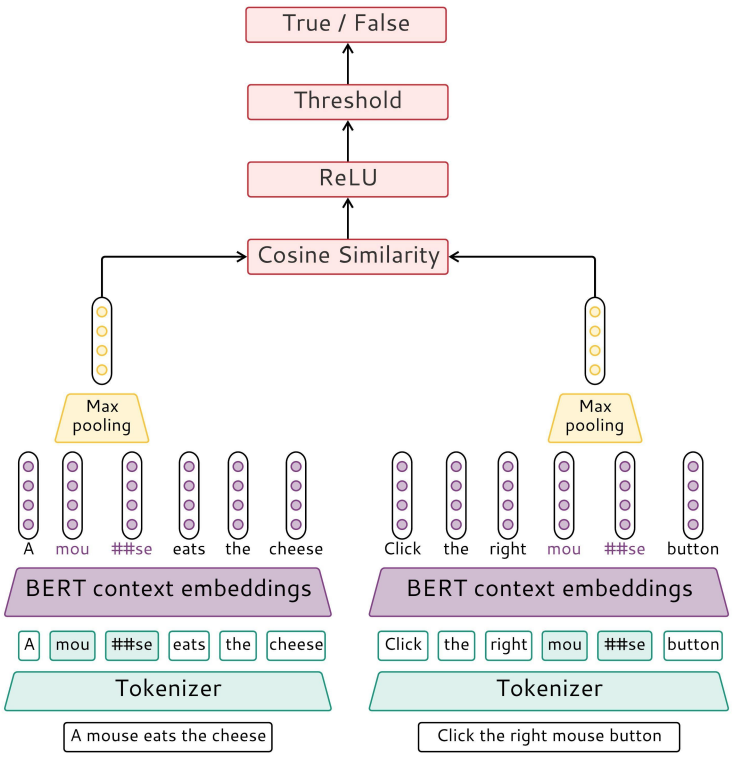

Figure 1: The scheme presents Cosine Similarity Architecture, which was used in the model achieving the best performance in our experiments.

best possible results we extend our train and development datasets with WiC dataset (Pilehvar and Camacho-Collados, 2019)[3], included to SuperGLUE (Wang et al., 2019) benchmark, for English sentence pairs.

We also conduct an experiment with our best model, using only default datasets provided by competition organisers. This experiment will be described at the end of Results section.

4 Experimental setup

In our setup we mix train and development data and split it randomly by unique lemmas in proportion 97.5% to 2.5%. Having 14680 samples in the first chunk and and 386 samples in the second chunk we use the first chunk for training and the second for validation.

During training, the data is processed by batches of size 8. Each sentence is split into 118 tokens maximum. In this way it is guaranteed that the longest sentence in the dataset is not going to be truncated.

Experiments with four different types of embeddings are conducted[4]:

- *bert-base-cased*: 12-layer, 768-hidden, 12-heads, 109M parameters;
- *bert-large-cased*: 24-layer, 1024-hidden, 16-heads, 335M parameters;
- *xlm-roberta-base*: ~270M parameters with 12-layers, 768-hidden-state, 3072 feed-forward hidden-state, 8-heads;
- *xlm-roberta-large*: ~550M parameters with 24-layers, 1024-hidden-state, 4096 feed-forward hidden-state, 16-heads.

We train our models for a maximum of 8 epochs and define an early stopping criteria. Every half of epoch (after training on the half of all the batches) we check if the loss on validation dataset is decreasing. If the loss does not decrease for 2 checks in a row, we stop training.

In all our experiments we use Binary Cross Entropy Loss as the loss function and AdamW optimizer with a learning rate set to 1e-5.

To conduct experiments we use version 1.7.1 of PyTorch (Paszke et al., 2019) together with version 0.8.2 of torchvision[5] and version 0.8.1 of torchtext[6],

[3] https://pilehvar.github.io/wic/
[4] https://huggingface.co/transformers
[5] https://github.com/pytorch/vision
[6] https://github.com/pytorch/text

version 1.1.6 of PyTorch Lightning[7] framework and version 4.2.2 of HuggingFace's Transformers (Wolf et al., 2020). From the latter we obtain BERT and XLM-RoBERTa model implementations.

As we define a probability threshold as a hyperparameter in Cosine Similarity approach, we provide its values for all experimental configurations in the Table 1.

embeddings	activation	threshold
xlm-roberta-large	**sigmoid**	0.680
xlm-roberta-base		0.632
bert-large-cased		0.609
bert-base-cased		0.678
xlm-roberta-large	**ReLU**	0.638
xlm-roberta-base		0.642
bert-large-cased		0.519
bert-base-cased		0.509

Table 1: Probability thresholds for Cosine Similarity Architecture. Abbreviations used: **activation** stands for *activation function* used, **threshold** stands for *probability threshold* of the model.

5 Results

In the Table 2 the results of the fine-tuning of language models with Multilayer Perceptron on top are presented. During the experiments we found out that for this dataset not only additional linear layers can not learn to measure the distance effectively, but they lead to overfitting in a few epochs. It is seen by the number of the passed epochs before the early stopping.

As [CLS] token is designed to accumulate sentence meaning we expected it to make the representations for each instance in a pair more complete. The results in the Table 2 show that the usage of [CLS] tokens give a moderate improvement to all models except for one with *xlm-roberta-large* embeddings.

Pre-trained language models, like BERT and XLM-RoBERTa, have the property of associating close vectors with similar words. Therefore to provide a baseline for the model described in Cosine Similarity approach we measure the accuracy of it without additional fine-tuning. Due to the technique used to evaluate the probability thresholds, the accuracies for configurations with different activations are identical in this case. Accuracies for different embeddings and thresholds for sigmoid and ReLU activations can be found in Table 3. Viewing the results on validation dataset we can estimate the quality of the approach and the results on test dataset confirm its relevance. Best accuracy on validation dataset is provided by *bert-large-cased* embeddings. In addition, the thresholds in Table 3 show how differently the vector spaces are arranged for BERT and XLM-RoBERTa models: for the second, a threshold of about 0.99 distinguishes vectors of words with different meanings from words with the same meanings.

embed	add cls	epochs	val	test
XLMR-l	**yes**	2.5	0.585	0.579
XLMR-b		2.5	0.580	0.580
BERT-l		3.5	0.585	0.548
BERT-b		2.5	0.588	0.565
XLMR-l	**no**	2	0.484	0.519
XLMR-b		2.5	0.590	0.611
BERT-l		3	0.598	0.583
BERT-b		2.5	**0.601**	**0.592**

Table 2: Accuracy of models with Multilayer Perceptron Architecture. Abbreviations used: **embed** stands for *embeddings*, **add cls** defines if *[CLS] token embedding was used*, **val** stands for *accuracy on validation dataset*, **test** stands for *accuracy on test dataset*. We refer to *xlm-roberta-large* as **XLMR-l**, to *xlm-roberta-base* as **XLMR-b**, to *bert-large-cased* as **BERT-l** and to *bert-base-cased* as **BERT-b**.

embed	sigm thld	ReLU thld	val	test
XLMR-l	0.73	0.995	0.645	0.659
XLMR-b	0.72	0.994	0.666	0.719
BERT-l	0.66	0.64	**0.710**	**0.780**
BERT-b	0.69	0.77	0.690	0.780

Table 3: Accuracy of models with Cosine Similarity Architecture without fine-tuning. Abbreviations used: **embed** stands for *embeddings*, **sigm thld** stands for *probability threshold of model using sigmoid activation*, **ReLU thld** stands for *probability threshold of model using ReLU activation*, **val** stands for *accuracy on validation dataset*, **test** stands for *accuracy on test dataset*. As models are not fine-tuned, accuracies on validation and test datasets are independent of the activation function. We refer to *xlm-roberta-large* as **XLMR-l**, to *xlm-roberta-base* as **XLMR-b**, to *bert-large-cased* as **BERT-l** and to *bert-base-cased* as **BERT-b**.

[7] https://github.com/PyTorchLightning/pytorch-lightning

Finally, Table 4 presents results of the experimental setup when the language models are fine-tuned using Cosine Similarity measure. It is worth mentioning that in such a setup there are no additional weights and only the layers of the language model are changing. It can be seen that such an architecture allows th model not to overfit for longer epochs.

embed	activ	epochs	val	test
XLMR-l	**sigm**	6	0.661	0.748
XLMR-b		3	0.679	0.746
BERT-l		3	0.728	0.823
BERT-b		3	0.676	0.727
XLMR-l	**ReLU**	5.5	0.785	0.876
XLMR-b		2	0.730	0.769
BERT-l		**4.5**	**0.808**	**0.927**
BERT-b		4	0.790	0.889

Table 4: Accuracy of models with Cosine Similarity Architecture. Abbreviations used: **embed** stands for *embeddings*, **activ** stands for the *activation function* used, **sigm** stands for *sigmoid activation function*, **val** stands for *accuracy on validation dataset*, **test** stands for *accuracy on test dataset*. We refer to *xlm-roberta-large* as **XLMR-l**, to *xlm-roberta-base* as **XLMR-b**, to *bert-large-cased* as **BERT-l** and to *bert-base-cased* as **BERT-b**.

While conducting the experiments, we judged the models by their performance on the validation dataset, not being able to check how representative it is. According to the obtained scores, the validation dataset is representative enough and is more challenging for the models than the test dataset.

To provide a convenient report we conduct an experiment with our best model (using *bert-large-cased* embeddings together with Cosine Similarity Architecture, using ReLU activation), which only uses data provided by organisers. We perform no further processing with the data and use it as is: train dataset is used for training and development dataset for validation. Being trained for 4 epochs the model in the experiment demonstrates 0.886 accuracy on validation dataset and **0.913** accuracy on test dataset. This result shows that using additional data leads to better performance.

6 Error analysis

Our best model leads to accuracy 92.7%. It means that our model has erroneously labeled 73 sentences in the 1000-sentence testset. The error analysis revealed that our model is not biased towards one or another class, it produced 37 false negative predictions and 36 false positives predictions. The next observation is related to the construction feature of the dataset. The dataset is organized in the following manner: for each combination of lemma and POS-tag there are two instances in the dataset. All three possible combinations of labels are presented, with prevalent case when one pair is labeled False and second True. The peculiarity of the dataset is that both instances have the same first sentence. We found that 20 out of 73 errors have these repeating first sentence. In other words, if the model produces incorrect prediction for one instance for lemma it tends to make a mistake for the second instance in the dataset. Due to the described peculiarity of the data, we can not speculate that certain lemma is a stumbling block for the model or it is just a context of the first sentence, that for example differs by genre or thematically from second sentence and complicates the prediction. The manual analysis of the errors has not revealed instances that could be considered hard and unclear for human assessment.

In order to reveal objectively hard instances among the errors of the best model, we have intersected the mislabeled pairs for all the models fine-tuned with Cosine Similarity. The intersection indicated that all but two instances were predicted correctly by at least one of the models. We can conclude that no objectively hard instances were presented in the erroneously labeled pairs by the best model. Additionally, the possible conclusion could be that an ensemble of our models could result in even more powerful solution for the task.

7 Conclusion

We have provided an overview of different approaches to fine-tune pre-trained language models for the task that is naturally suitable for them – detecting the distance between representations of the words.

We have showed that, for the data of given amount and type, learning distance between words in context with Multilayer Perceptron neural network is not applicable and generally leads to overfitting.

Using Cosine Similarity to predict probability during pre-trained embeddings fine-tuning leads to much more promising results, when activated with ReLU layer.

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. Cite arxiv:1810.04805Comment: 13 pages.

Christiane Fellbaum. 2005. WordNet and wordnets. In *Encyclopedia of Language and Linguistics*, pages 665–670, Oxford. Elsevier.

Jeremy Howard and Sebastian Ruder. 2018. Fine-tuned language models for text classification. *CoRR*, abs/1801.06146.

Federico Martelli, Najla Kalach, Gabriele Tola, and Roberto Navigli. 2021. SemEval-2021 Task 2: Multilingual and Cross-lingual Word-in-Context Disambiguation (MCL-WiC). In *Proceedings of the Fifteenth Workshop on Semantic Evaluation (SemEval-2021)*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Neural and Information Processing System (NIPS)*.

Roberto Navigli. 2009. Word sense disambiguation: a survey. *ACM COMPUTING SURVEYS*, 41(2):1–69.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Mohammad Taher Pilehvar and Jose Camacho-Collados. 2019. WiC: the word-in-context dataset for evaluating context-sensitive meaning representations. 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 1267–1273, Minneapolis, Minnesota. Association for Computational Linguistics.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. SuperGLUE: A stickier benchmark for general-purpose language understanding systems. *arXiv preprint 1905.00537*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.