Neural Networks Theory and Implementation: Language Project

Sergey Berezin
Universität des Saarlandes
sebe00003@teams.uni-saarland.de

Laurent Hug
Universität des Saarlandes
s8lahugg@stud.uni-saarland.de

Abstract

The problem of automatically figuring out how closely two sentences are related to one another in their meaning, also known as semantic textual similarity, has been extensively researched in recent years. Great results have recently been achieved by models such as sentence-BERT Reimers and Gurevych [2019] and SimCSE Gao et al. [2021]. In this project we explore three different Siamese network architectures and their results in the SICK data set.

1 Introduction

The task of determining semantic similarity in text is one that finds application in many fields. Measuring the distance between sentences based on their meaning is an important step in allowing AI to understand language. As such many attempts at creating and optimizing machine learning models to complete this task have been made. Within this research project, our goal was to read about and implement some of these models in order to gain a deeper understanding into natural language processing, as well as machine learning as a whole.
The first step was to combine the ideas proposed by Mueller and Thyagarajan [2016] and by Lin et al. [2017]. The result was a Siamese architecture that uses bidirectional LSTMs (biLSTM) and self-attention to learn sentence embeddings. The biLSTM layer allows the model to learn long-range dependencies, while the self-attention layer is able to focus on different aspects of meaning within the sentences.
In the second step, we then enhanced the previous architecture by including a transformer encoder layer based on the work of Vaswani et al. [2017]. This encoder layer works by leveraging the power of multi-head attention, giving it the ability to develop an even richer understanding of sentence semantics.
Lastly, we further improved the sentence embeddings by completely doing away with the biLSTM as well as the final self-attention layer. This last architecture now only relies on the transformer encoder, its multi-head attention and its positional embedding. As we will show throughout this paper, using this simpler model actually improves performance on the SICK data set considerably.

2 Methodology

The provided SICK data set was preprocessed according to instructions. That is, all words in the sentences were set to lower case, all punctuation was removed, and, for task 1 and task 2, the stop words were removed. Tokenization and vocabulary building was done using spaCy Honnibal and Montani [2017] torchtext and the pretrained fasttext vectors Bojanowski et al. [2016].
As a measurement of the model accuracy we chose to use Spearmans rank-order correlation. Another popular choice is to use Pearson's correlation, but it can be sensitive to outliers and can only measure linear relationships Reimers et al. [2016].

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

2.1 Hyperparameter tuning and Genetic Algorithm

The first step for each task was always to tune the model hyperparameters. For this, we developed a genetic algorithm in an attempt to automatically find the best possible hyperparameter combination. A genetic algorithm, like neural networks, takes its inspiration from nature, i.e. the process of evolution. First, a set of models is created from randomly sampled hyperparameter combinations. Each one of these was then trained for a certain number of epochs, and its maximum validation accuracy achieved during this training was recorded. After all models are finished training, they are sorted based on this validation accuracy and only the top performing models are kept in the list while the worse models are pruned. For the next generation, a new set of child models is created from the remaining models by combining their hyperparameters. These children are then also trained and the process is repeated for a set number of generations.
When creating the child models, there is also a chance that a hyperparameter can randomly "mutate", that is change its value. These mutations also occur naturally, and they allow the system to randomly generate a new model hyperparameter combination that might reach even better performance. If it does, its hyperparameters will be used to create further children, thus continuously improving the best overall accuracy. This, of course, introduces new hyperparameters that must be set. In our case we chose to train each model for 25 epochs and to run the genetic algorithm for 20 generations with a mutation rate of $m = 0.25$. Choosing just 25 epochs as a cutoff favors models that reach good accuracy quickly, but there might be some models that could reach even better accuracy if trained for more epochs.

2.2 Model Training

After tuning the optimal hyperparameters using our genetic algorithm, we trained a final model in each of the tasks. For this we chose a sightly longer training time of 50 epochs, in case there were further improvements to be made by training longer. In order to avoid overfitting, we applied early stopping by only saving the model that had the best validation set accuracy. After training, the best model was always loaded from the disk and the final accuracy values were determined on the unseen test set.

3 Siamese BiLSTM with Transformer Encoder (Task 2)

While tuning the hyperparameters for the task 2 model, we noticed that the genetic algorithm would always prefer a model that either has a very low self-attention penalty, or very few self-attention vectors. To investigate what the reason for this might be, we trained two different models, one with a penalty of $p = 0.9$ and $r = 10$ self-attention vectors (active), and one with $p = 0.0$ and $r = 1$ self-attention vector (inactive).

3.1 Self-Attention Layer Active

Figure 1 shows the training and validation accuracy during training of the model with $r = 10$ vectors of self -attention and a penalty of $p = 0.9$. Both the training and validation accuracy initially fall sharply before recovering after approximately 15 epochs. The reason for this behavior is not entirely clear, but we have some possible explanations. It might be, that the transformer encoder attention and self-attention layer are working against each other at first. Since the self-attention penalty causes the model to have a large initial loss even though the Spearman correlation might actually be quite high. But, because the criterion used for backpropagation was MSELoss, the model first decides to unlearn the embeddings provided by the pretrained fasttext vectors.

In that case, the model decides to disregard the fasttext embeddings and relearn them basically from scratch. After some epochs, this leads to a steep rise in accuracy. One advantage of this approach is that the self-attention layer can actually contribute to the model as shown in Figure 2. We can see that the self-attention penalty works in forcing the vectors to focus on different aspects of the sentence. Some vectors focus on the object, some on the subject and some on the interaction between subject and verb.
Still, this approach can not leverage the advantage provided by the pretrained fasttext vectors. The results also show that this approach is not very good, achieving a Spearman correlation on the test set of only $\rho = 0.65$.

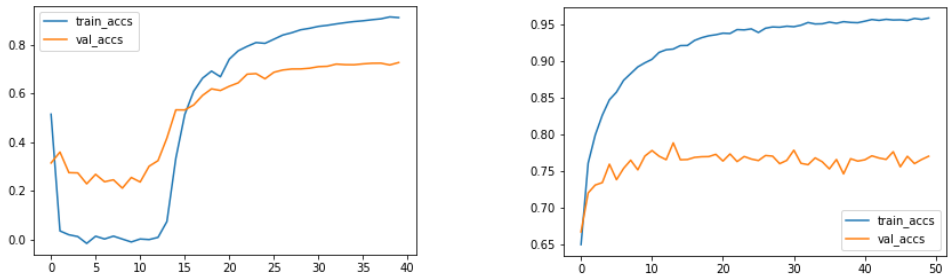

Figure 1: **Left**:Training and validation accuracy accuracy for the model with multiple self-attention vectors and large penalty. The accuracy initially falls steeply, as the self-attention layer seems to clash with the transformer encoding before learning new embeddings after about 15 epochs.
Right: Training and validation accuracy accuracy for the model with only one self-attention vector and no self-attention penalty. The dip in accuracy visible on the left is no longer present.

3.2 Self-Attention Layer Inactive

The genetic algorithm hyperparameter tuning always tended toward models with very few self-attention vectors $r = 1$ or very low self-attention penalty $p = 0$. Figure 1 shows the training and validation accuracy over 50 epochs of training. In contrast to the left side of Figure 1, the model no longer has the initial accuracy decline. Due to the insignificant self-attention penalty added to the loss, the model no longer tries to train the word embeddings from scratch. It can now take advantage of the fasttext embeddings to start training from a good initialization. The effect this has on the self-attention vectors can be seen in Figure 2. The attention is now spread evenly over every word of the sentences. It therefore seems like almost all of the learning is taking place in the transformer encoder layers.

The result is much better than before with $\rho = 0.72$. This marks an even more significant improvement over the best task 1 model with $\Delta\rho = 0.11$. The transformer encoder multi-head attention must therefore be much better suited at learning semantic textual similarity. The task 1 model had to rely entirely on the self-attention layer for its attention scores, while the task 2 model wants to disregard this part as much as possible. An overview of all model results can be seen in Table 1.

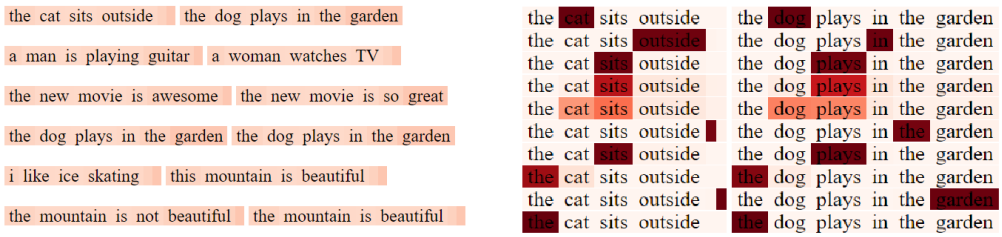

Figure 2: Left: Self-attention layer vectors r visualized over sentences when only one vector exists and the penalty is set to 0. We can see that attention is just evenly spread across words. Right: self-attention vectors when $r = 10$ and p$= 0.9$. We can see that different layers pay attention to different aspects of the sentences, e.g. "cat sits", thanks to the self-attention penalty outlined in

4 Sentence-BERT using simple Transformer (Task 3)

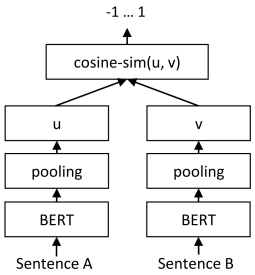

Figure 3: The sentence-BERT Siamese architecture. Image taken from Reimers and Gurevych [2019].

Due to the insights gained during task 2, we decided on removing the self-attention layer entirely for task 3. Since the best results were achieved when the self-attention was as small as possible, this should lead to even better results. Reading through the literature, we came across a paper about expanding the BERT architecture to include sentences Reimers and Gurevych [2019]. The so-called sentence-BERT now only relies on BERT and a pooling layer to calculate embeddings. Instead of using BERT, we then decided to use our own simple transformer encoder implementation and initialize the embeddings using the fasttext vector like in tasks 1 and 2. As a pooling layer we chose to implement mean pooling.
We also changed the similarity score calculation to match the sentence-BERT paper. Using cosine similarity instead of Manhattan distance is found in many recent papers on semantic sentence similarity and seems to improve results.
Lastly, we decided to no longer remove the stop words from the input sentences before tokenization. We found that the stop words were too aggressive, removing some important context in the process. For example, the sentences ”The black dog is running through the grass” and ”The black dog is not running through the grass” both turn into the same sentence after removing stop words: ”black dog running grass”. The SICK data set has many such sentences with contradictory meaning and they are mostly rated between 3.0 and 3.5 in gold label similarity. Yet the transformer will generate the same embedding for both sentences leading to a predicted score of 5.0. Obviously, some important context is lost when removing the stop words. The results corroborate this intuition, showing a large increase in accuracy when using the entire sentence.

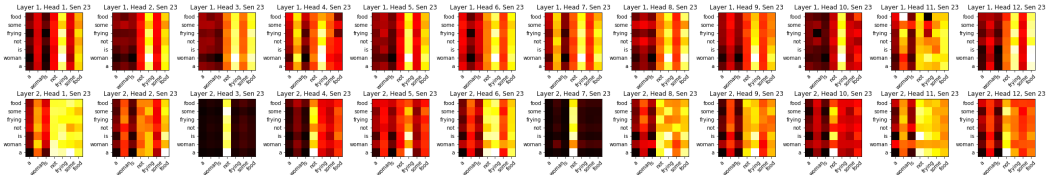

Figure 4: Multi-Head attention visualized for the first two layers of the transformer encoder. The columns show attention heads, the rows show attention layers. Different heads and layers pay attention to different aspects of the sentence. For example heads 3 and 6 in layer 2 pay a lot of attention to the word ”not”. The model can use this to learn that the sentence might indicate a contradiction.

Initially we experimented with no longer using the fasttext embedding vectors and instead allowing the transformer to learn an embedding by itself from scratch. But, as this turned out to produce worse results than using the fasttext initialization, we quickly reverted that decision.
After hyperparameter tuning using our genetic algorithm, we settled on a transformer implementation using 12 encoder layers and 12 heads of multi-head attention. The final model was trained for 50 epochs, keeping the model with the highest validation set accuracy. The result was a model with test set Spearman correlation of $\rho = 0.79$, beating all other models except for the MaLSTM. The intuition gained in task 2 was therefore shown to be correct,a s doing away with the self-attention layer and biLSTM considerably improved model performance.

In Figure 4 we can gain some idea on why the multi-head attention works so well. Here we can see the attention scores the words in a sample sentence over all other words in the sentence after the transformer attention softmax. The sentence is "a woman is not frying some food" and depicted are all attention heads in the first two layers.
The multiple layers and heads of the transformer attention allow it to extract a rich variety of meaning from a sentence. For example in layer two of Figure 4 we can find some attention heads that focus entirely on the word "not", allowing them to understand that the meaning of this sentence might be inverted. In layer one, many heads pay attention to the words "frying" and "food". It seems like they are trying to focus on the action that is happening in the sentence. Combining all of these aspects allow the transformer to create a meaningful sentence embedding.

5 Results

The final accuracy result on the SICK data test set of all of our models can be seen in Table 1. This table also includes the accuracy obtained by the sentence-BERT model as well as the original MaLSTM model for comparison.
The results show that the transformer encoder multi-headed self-attention layers represent a large improvement over the simple self-attention layer with penalty. Just by adding the transformer encoder into the architecture we observed an improvement in Spearman correlation of $\rho = 0.05$. Minimizing the self-attention layer as much as possible, by reducing the amount of vector to $r = 1$ and the self-attention penalty to $p = 0$, further improves these results. By relying more on the transformer encoder, as well as the pretrained fasttext embeddings, we observe an improvement of $\rho = 0.12$ as compared to the original task 1 implementation.
Completely removing the self-attention layer and replacing the biLSTM layer with a mean pooling layer as shown in Figure 3 once again increases performance. The final model was able to increase Spearman correlation by $\rho = 0.06$ over the best task 2 model, even beating the sentence-BERT model by $\rho = 0.04$. It was able to get quite close to the performance described original MaLSTM paper by Mueller and Thyagarajan [2016], but did not quite reach the same performance. Possible improvements that could still be done include data augmentation and transfer learning. Both of these techniques were used by Mueller and Thyagarajan [2016], but left out in our implementations.

Table 1: Results of the model accuracy on the SICK data set. Task 1 refers to the initial biLSTM with self-attention, Task 2 active refers to the task 2 model with self-attention penalty, task 2 inactive the task 2 model with minimal self-attention and task3 refers to our implementation of Reimers and Gurevych [2019].

Model	Pearson r	Spearman ρ
Task 1	0.72	0.60
Task 2 active	0.75	0.65
Task 2 inactive	0.78	0.72
Task 3	0.82	0.78
MaLSTM	0.88	0.83
s-BERT	-	0.74

6 Conclusion

Overall, this software project was a success. The trained models showed an improvement in both Pearson and Spearman correlation with each iteration of the network architecture. The last model was even able to beat the state of the art sentence-BERT implementation it is based on. Of course, our model was specifically trained on the SICK data set while the BERT model used in sentence-BERT is much more general. We were not able to beat the original MaLSTM by Mueller and Thyagarajan [2016]. This might be because they used data augmentation methods like synonym replacement and transfer learning while we did not. One question that remains is how well the final models perform on different data sets. Are they able to infer sentence semantics when the sentences have a different structure? Perhaps here the decision to not remove stop words would lead to bad performances.
In the end, our results show that Vaswani et al. [2017] were right in their iconic paper when they said "Attention is All You Need".

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. Simcse: Simple contrastive learning of sentence embeddings. *CoRR*, abs/2104.08821, 2021. URL `https://arxiv.org/abs/2104.08821`.

Matthew Honnibal and Ines Montani. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear, 2017.

Zhouhan Lin, Minwei Feng, Cícero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. *CoRR*, abs/1703.03130, 2017. URL `http://arxiv.org/abs/1703.03130`.

Jonas Mueller and Aditya Thyagarajan. Siamese recurrent architectures for learning sentence similarity. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, page 2786–2792. AAAI Press, 2016.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *CoRR*, abs/1908.10084, 2019. URL `http://arxiv.org/abs/1908.10084`.

Nils Reimers, Philip Beyer, and Iryna Gurevych. Task-oriented intrinsic evaluation of semantic textual similarity. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 87–96, Osaka, Japan, December 2016. The COLING 2016 Organizing Committee. URL `https://aclanthology.org/C16-1009`.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL `http://arxiv.org/abs/1706.03762`.