Sentence Similarity on Quora Question Pairs

1st Ayca Meric Celik
Hacettepe University, Departmen of Computer Engineering
Ankara, Turkey
aycameric.celik@gmail.com

2nd Okan ALAN
Hacettepe University, Departmen of Computer Engineering
Ankara, Turkey
okanalan97.14@gmail.com

***Abstract*—Semantic similarity is of the most popular NLP topics of recent years. After the great success of word embeddings and their applications, we made great progress in understanding natural languages. However, the challenge of evaluating the relations between sentences remains. There are a lot of promising approaches to this problem, and we aim to investigate them. In this paper, we will examine the task of "sentence pair classification" on Quora Question Pairs dataset with transformer-based models BERT and ALBERT.**

***Index Terms*—Quora Question Pair, Sentence Similarity, BERT, ALBERT**

I. Introduction

FOr our experiments, we decided to use the "Quora Question Pairs"(QQP) [10] dataset, and we focused on the subproblem of semantic similarity named "Semantic Question Matching"[1].

Quora is one of the most popular question-answering platforms of our days. So it is not surprising that we observe many duplicate questions on that website. Different users ask the same question in different ways, but the overall intent and the meaning of the entry are the same. Thus, the expected answers would be the same. Detecting these duplicate questions can help both the users and the platform in many ways:

- The users could be redirected to the previously asked questions to reduce the waiting time for the answers.
- The platform could eliminate duplicate questions for the sake of compactness and simplicity.
- Various answers from different users would not be scattered to different entries.

This task is also a good and challenging example of semantic similarity measurement problem. Researchers have proposed different results from various approaches, such as Word2Vec [5], LSTM [6], etc. In our experiments, we fine-tuned transformer-based models for our pair classification task. We also examined the similarities between question vectors by using cosine similarity. In the end, transformer-based models produced pretty promising results.

II. Related Works

In recent years, the natural language processing tasks have gained importance and continue to increase. Sentence similarity is one of the most popular topic in natural language processing area.

We worked on the "Quora Question Pairs (QQP)" dataset. Considering the publication date of our data set, the growth rate of the natural language processing field had just begun to increase. At that time, since popular deep learning techniques have not yet published, we found plenty of machine learning based works that are dealt with this dataset. Therefore, we can gather under two main headings which are worked with machine learning methods and deep learning techniques.

In the initial works, different machine learning models are used, such as SVM, decision tree, logistic regression, etc. ([11], [12], and more). The best scores on the test set differs between 65% and 80%.

In further researches, the deep learning based model such as FNN, LSTM, GRU are used. ([7], [9], [13], [14], [15],etc.) These models performed much better than the machine learning based model on this task.The best scores on the test set differs between 80% and 90%.

III. Dataset

For our experiments, we decided to use the "Quora Question Pairs" (QQP) dataset, and we focused on the subtask of semantic similarity task, named "Semantic Question Matching" [1].

Quora is one of the most popular question-answering platforms of our days. So it is not surprising that we observe many duplicate questions on that website. Different users ask the same question in different ways, but the overall intent and the meaning of the entry are the same. Thus, the expected answers would be the same. Detecting these duplicate questions can help both the users and the platform in many ways: The users could be redirected to the previously asked questions to reduce the waiting time for the answers. The platform could eliminate duplicate questions for the sake of compactness and simplicity. Various answers from different users would not be scattered to different entries.

In 2017, Quora released the dataset with more than 400,000 pairs of questions. They also started a competition, and it attracted a lot of attention. Researchers, NLP enthusiasts, and students had a great opportunity to work with organic data on semantic similarity tasks and compare the results with others easily. The dataset is still widely used and included in NLP benchmarks, such as QLUE.

For our research, we chose the GLUE benchmark [2] version of the dataset. It consists of three .tsv files as "train", "dev", and "test". There are 363192, 40372, and 390965 rows in the files respectively. Train and dev sets have 5 columns representing the first question's id, second question's id, the

full text of the first question, full text of the second question, and the label indicating whether the questions are duplicate or not. The test set only contains two columns representing the full texts of first and second questions.

The best accuracy and F1 score obtained in GLUE benchmark are 86.5 and 66.1. [2]

IV. Approach

We decided to use BERT-based transformer models for this task since we believe that these would produce good baseline scores for us. Our aim is to decide whether the two pairs of questions have the same intent or not, so we decided to use these models as a sentence-pair classifier for our task.

A. BERT

BERT (Bidirectional Encoder Representations from Transformers) made a huge impact on NLP history. [7] Compared to previous models, we can produce state-of-the-art results in various sentence-level and token-level tasks by just fine-tuning the model. It enables pre-trained deep bidirectional representations by using masked language models, so it is quite robust. These are all due to the model's architecture, which is a multi-layer bidirectional Transformer encoder. [3]

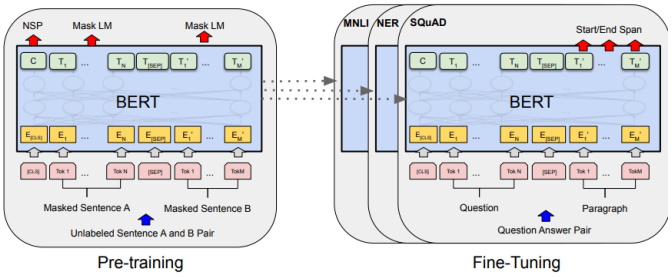

Fig. 1. Overall pre-training and fine-tuning procedures for BERT [7]

B. ALBERT

A Lite BERT(ALBERT) [8] was developed to increase the speed of BERT and decrease memory usage. Even though ALBERT-base model has only 12M parameters, an 89% less than BERT-base model, it is as robust as BERT!

C. SpanBERT

SpanBERT [15] is a pre-training method that is designed to better represent and predict spans of text. This approach extends BERT by masking contiguous random spans, rather than random tokens, and training the span boundary representations to predict the entire content of the masked span, without relying on the individual token representations within it.

D. DistilBERT

DistilBERT [26] is a pre-trained smaller general-purpose language representation model, which can then be fine-tuned with good performances on a wide range of tasks like its larger counterparts. The researchers leverage knowledge distillation during the pre-training phase and proved that it is possible to reduce the size of a BERT model by 40%, while retaining 97% of its language understanding capabilities and being 60% faster. To leverage the inductive biases learned by larger models during pre-training, they introduce a triple loss combining language modeling, distillation and cosine-distance losses.

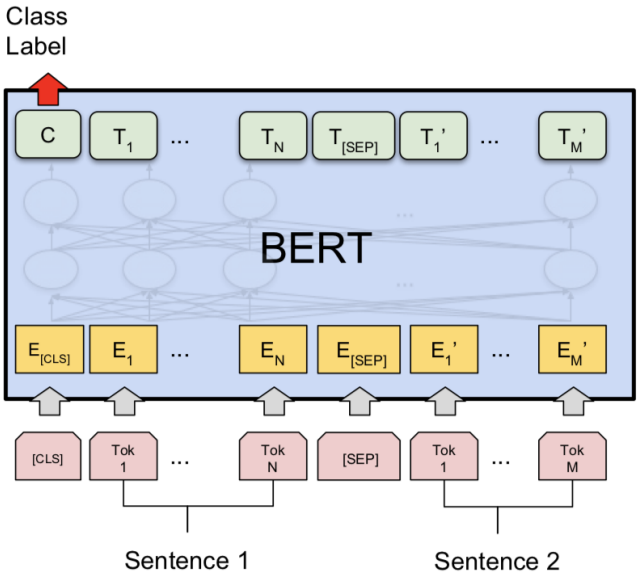

Fig. 2. Sentence Pair Classification tasks in BERT [7]

E. Custom SNLI

We decided to use some other models that are not transformer-based. For this purpose, we selected a custom Stanford Natural Language Inference benchmark implemented in Keras. [25]

In this implementation, each question in the pair is represented as simple summation of GloVe word embeddings. Instead of softmax layer of the original SNLI model, a dense layer with sigmoid activation is used as the final layer. Another difference is that max operator is used instead of sum to combine GloVe embeddings into a question representation. Finally, cross-entropy loss and Adam optimizer is used.

V. Experiment

A. Preprocessing The Data

Question 1	Question 2	Duplicate?
How is air traffic controlled?	How do you become an air traffic controller?	No
Why doesn't everyone I ask to answer a question-answer it on Quora?	Why doesn't anyone here answer my questions?	Yes

TABLE I
THE RAW EXAMPLES FROM THE DATASET ARE AS FOLLOWS

Fig. 3. Model Architecture of Custom-SNLI

We followed the steps below to prepare our dataset:

1) Dropped the faulty lines (3 lines in total).
2) Dropped "qid1" and "qid2" columns in the train and dev set.
3) Reset the indexes in three sets.
4) Converted all texts to lowercase. (We used uncased models for the training).
5) Expanded the English contractions: As an example, "couldn't" becomes "could not". This is quite important since we later insert spaces before and after the punctuation marks. With this, we were able to protect the contractions from deformation.
6) Punctuation isolation: We insert a space before and after each punctuation. We prefer to keep the punctuation marks since BERT performs better with them. Instead, we separated them from words to be correctly tokenized by BERT TableII.

Question 1	Question 2	Duplicate?
how is air traffic controlled ?	how do you become an air traffic controller ?	No
why does not everyone i ask to answer a question answer it on quora ?	why does not anyone here answer my questions ?	Yes

TABLE II
THE EXAMPLES AFTER PREPROCESSING

B. Approach #1 - Thresholding The Similarity Scores of Vectors

In order to review the quality of BERT sentence vectors, we encode text pairs without any fine-tuning. Later, we computed the cosine similarity of these two sentence vectors and compared them with the ground-truth label. We used Pytorch implementation of BERT from HuggingFace [4] for this task.

Since getting the sentence embedding is relatively slow, and our main goal is to train a classifier, we did not spend so much time on this part. We only encoded the dev set and calculated the accuracy by defining a threshold value for similarities by hand. TableIII

Threshold	Accuracy
0.70	0.624
0.75	0.668
0.80	0.704
0.85	0.728
0.90	0.728
0.95	0.697

TABLE III
THRESHOLD - ACCURACY

The threshold of 0.85/0.90 gave the best result. However, there are some problems. Let us review these false positive examples TableIV:

Question 1	Question 2	Similarity	Duplicate?
why is my life getting so complicated ?	why is my life so complicated ?	0.989	No
why are african - americans so beautiful ?	why are hispanics so beautiful ?	0.835	No

TABLE IV
FALSE POSITIVE EXAMPLES

Although some question pairs have very high similarity scores, it does not necessarily mean that these two sentences are duplicate. The questions can have similarities in terms of meaning, however the intent of the question may differ. So, the answers of these questions may differ. This means that these two questions are not duplicates, but this method cannot differentiate that, and produce false-positive resultsTableV.

Question 1	Question 2	Similarity	Duplicate?
what are your views about demonetisation in india ?	what do you think about the ban on 500 and 1000 denomination notes in india ?	0.595	Yes
how can i get rid of a bad habbit ?	what are good strategies for getting rid of a bad habit ?	0.642	Yes

TABLE V
FALSE NEGATIVE EXAMPLES

Although similarity scores are generally high and the model tends to produce false positives, it can produce false negatives as well. Sometimes, the model cannot capture the desired relation between questions.

As we can see, this method is not powerful enough to classify the pairs correctly. Additionally, thresholding the outputs

manually is not the optimal way to handle the predictions. We need to automate this task by training a classifier.

C. Approach #2: Fine-Tuning BERT-LIKE MODELS

To train a sentence-pair classifier, we fine-tuned BERT, ALBERT and DistilBERT and SpanBERT on our train set, and calculated accuracy on the dev set. We also got the test set prediction for the future evaluation on the GLUE benchmark. We used "SimpleTransformers"[5] module, which is a high-level implementation built on HuggerFace Transformers, for the sake of simplicity and efficiency. We trained our model in the free servers of Google Colab (Tesla P100).

We got our best results with the batch size as 16 and epoch count as 3. We experimented with different parameters, and the results as follows:

- **Learning Rate**: Default $4e^{-5}$. Increasing the learning rate (as $4e^{-4}$) decreased the accuracy significantly, so we left it as the default value.
- **Epoch:** We trained our model in 3 and 2 epochs, and as we expected, higher epoch gave higher accuracy.
- **Batch Size:** Since models are quite large, the highest batch size that we can use was 16. We were able to use batch size of 32 during the training of "albert-base-v2" only.
- **Models:** We experimented with "bert-base-uncased", "bert-large-uncased", "albert-base-v2" and "albert-large-v2". The larger models such as "albert-xlarge-v2" and "albert-xxlarge-v2" either did not fit into memory, or did not produce good results. Our guess on the reason that the bigger models performs poorer is that it is unnecessarily complex to be trained on our dataset.

Epoch	Method	Accuracy	Eval Loss
3	albert-base-v2	0.798	0.509
2	albert-base-v2	0.785	0.444
3	bert-base-uncased	0.765	0.503
2	bert-base-uncased	0.740	0.511

TABLE VI
COSINE SIMILARITY & THRESHOLDING RESULTS.

1) Experiment #1: Only 1000 Training Examples: As you can see, it outperformed our previous thresholding method with only 1000 training examples and 2 epochs. It convinced us that we are on the right track. We also observed that in the 3rd epoch, the model continues to train. In our further experiments, we aim to increase the epoch number.

2) Experiment #2: Different Model Versions: Since the training takes a quite long time, we experimented with different BERT/ALBERT versions with 10.000 training examples at maximum. We trained "albert-xlarge-base" and "bert-large-uncased" in this stage.

Epoch	Method	Accuracy	Eval Loss
3	albert-large-v2	0.631	0.650
3	bert-large-uncased	0.631	0.663

TABLE VII
TRAINING RESULTS WITH ONLY 1000 EXAMPLES.

As you can see, since the models are larger, it is unable to converge with few training examples. Let us examine all the models trained with 10.000 examples: (TableVIII)

Epoch	Method	Accuracy	Eval Loss
3	bert-base-uncased	0.837	0.650
3	bert-large-uncased	0.836	0.504
3	albert-base-v2	0.830	0.427
3	albert-large-v2	0.631	0.658
3	spanbert-base-cased	0.832	0.458
3	distilbert-base-uncased	0.814	0.489

TABLE VIII
EVALUATION RESULTS OF THE MODELS TRAINED WITH ONLY 10.000 EXAMPLES.

It seems like larger models did not improve the accuracy scores. For the sake of efficiency, we continued to experiment with the base models.

3) Experiment #3: Full Training Set: The training of each model took 5 hours. The results are as follows:

Epoch	Method	Accuracy	Eval Loss
3	albert-base-v2	0.908	0.278
3	bert-base-uncased	**0.911**	0.344
3	spanbert-base-cased	0.908	0.267
3	distilbert-base-uncased	0.903	0.293

TABLE IX
EVALUATION RESULTS OF THE MODELS TRAINED WITH FULL TRAIN SET.

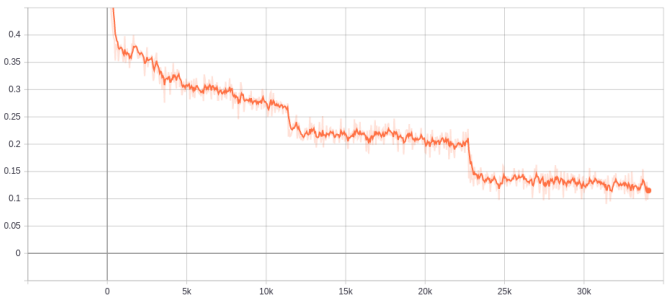

Fig. 4. ALBERT's Loss Graph

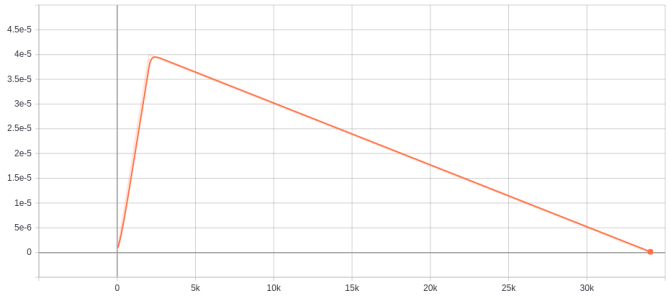

Fig. 5. ALBERT's Learning Rate Graph

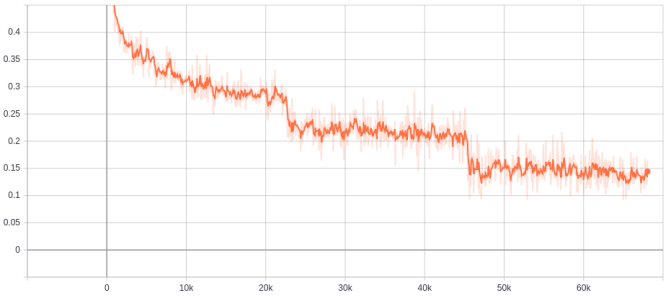

Fig. 6. BERT's Loss Graph

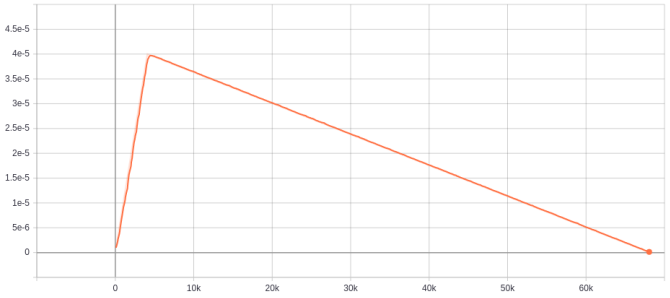

Fig. 7. BERT's Learning Rate Graph

We reached an accuracy score of **%91**! As you can see, more training example yield a more powerful classification model. Since our time was limited due to Google Colab's restrictions, we were not able to train our model with higher epochs, but we are expecting higher accuracies with more epochs as well!

D. Approach #3: Fine-Tuning Modified SNLI Benchmark Model

We used "GloVe.840B" embeddings with 300 dimensions to obtain question vectors. Then, we trained our model with dropout rate of 0.2 and batch size of 512.

We used 20% of the training set as the validations set. Then, we evaluate the model on the dev set.

As you can see, this model did not produce promising results. We finished experimenting with it, and this work proved that transformer models are much suitable for this task compared to LSTM and NN based ones.

Epoch	Method	Accuracy	Eval Loss
25	Custom-SNLI	0.671	2.404
100	Custom-SNLI	0.670	2.304

TABLE X
EVALUATION RESULTS OF THE CUSTOM SNLI MODEL WITH DIFFERENT EPOCHS.

We compared our results with previous scores in below.

Model	Accuracy(%) on Dev Set
Feature Vector	70.46
Tree Kernel	74.05
CNN	77.79
LSTM [6]	77.06
BiMPM [16]	88.69
ABCNN [17]	64,47
pt-DecAtt$_{word}$ [18]	88.44
pt-DecAtt$_{char}$ [18]	88.89
REGMAPR [19]	89.05
DIIN [20]	89.44
MwAN [21]	89.60
DIIN (Ensemble) [20]	90.48
MFAE (BERT Ensemble) [9]	90.61
STILT-BERT [13]	91.5
ALBERT	0.908
BERT	**0.911**
SpanBERT	0.908
DistilBERT	0.903
Custom SNLI	0.671

TABLE XI
COMPARISON WITH OTHER WORKS

VI. CONCLUSION

We observed that transformer-based models such as BERT and ALBERT perform quite well in the task of sentence pair classification. By just fine-tuned the model's classifier on our dataset, we were able to get the accuracy of %91. Both BERT and ALBERT produced better results on fewer examples than the thresholding method we tried earlier. This proves two things: fine-tuning really makes difference, and deep neural networks outperforms simple manual approaches -as we expected-. We also saw that the larger versions of BERT and ALBERT are unnecessary to use in our dataset since they increase the training time and resource need quite a lot without improving the scores much.

REFERENCES

[1] Nikhil Dandekar: February 13, 2017 *Semantic Question Matching with Deep Learning*

[2] Alex Wang1, Amanpreet Singh1, Julian Michael2, Felix Hill3, Omer Levy, Samuel R. Bowman *GLUE: A MULTI-TASK BENCHMARK AND ANALYSIS PLATFORM FOR NATURAL LANGUAGE UNDERSTANDING*

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin *Attention Is All You Need*

[4] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Jamie Brew *HuggingFace's Transformers: State-of-the-art Natural Language Processing*

[5] Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean *Efficient Estimation of Word Representations in Vector Space*

[6] Felix A. Gers, Jurgen Schmidhuber, Fred Cummins *Learning to Forget: Continual Prediction with LSTM*
[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*
[8] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, Radu Soricut *ALBERT: A LITE BERT FOR SELF-SUPERVISED LEARNING OF LANGUAGE REPRESENTATIONS*
[9] Rong Zhang, Qifei Zhou, Bo Wu, Weiping Li, Tong Mo *What Do Questions Exactly Ask?MFAE: Duplicate Question Identification with Multi-Fusion Asking Emphasis*
[10] Z. Chen, H. Zhang, X. Zhang, and L. Zhao. *Quora question pairs, 2017*
[11] Udi Bhaskar *Quora Question Pair Similarity*
[12] Lakshay Sharma, Laura Graesser, Nikita Nangiann, Utku Evci *Natural Language Understanding with the Quora Question Pairs Dataset*
[13] Jason Phang, Thibault Fevry, Samuel R. Bowman *Sentence Encoders on STILTs:Supplementary Training on Intermediate Labeled-data Tasks*
[14] Zainab Imtiaz, Muhammad Umer, Muhammad Ahmad, Saleem Ullah, Gyu Sang Choi, Arif Mehmood *Duplicate Questions Pair DetectionUsing Siamese MaLSTM*
[15] Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, Omer Levy *SpanBERT: Improving Pre-training by Representingand Predicting Spans*
[16] Z. Wang, W. Hamza, R. Florian *Bilateralmulti-perspective matching for natural language sen-tences, in AAAI, 2017, pp. 4144–4150*
[17] W. Yin, H. Schutze, B. Xiang, B. Zhu *Abcnn: Attention-based convolutional neural networkfor modeling sentence pairs*
[18] G. S. Tomar, T. Duque, O. Tackstrom, J. Uszkoreit, D. Das *Neural paraphrase identificationof questions with noisy pretraining*
[19] S. Brahma, *Regmapr-a recipe for textual matching*
[20] Y. Gong, H. Luo, J. Zhang *Natural languageinference over interaction space*
[21] C. Tan, F. Wei, W. Wang, W. Lv, M. Zhou *Multiway attention networks for modeling sentencepairs*
[22] Antonio Uva, Daniele Bonadiman, Alessandro Moschitti *Injecting Relational Structural Representation in Neural Networks for Question*
[23] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever *Improving Language Understanding by Generative Pre-Training*
[24] Matthew Peters, Mark Neumann, Mohit Iyyer, MattGardner, Christopher Clark, Kenton Lee, Luke Zettlemoyer *Deep contextualized word representations*
[25] Samuel R. Bowman and Gabor Angeli and Christopher Potts and Christopher D. Manning *A large annotated corpus for learning natural language inference*
[26] Victor Sanh and Lysandre Debut and Julien Chaumond and Thomas Wolf *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter*