NLP Project
Sentence Similarity using PAWS dataset

Anirudh Krishna Lakshmanan, Sagar Seth, Zhemin Zhang
{alaksh19, sseth9, zzhan215}@asu.edu

Abstract

A paraphrase is a rewording of a sentence(s) conveying the same meaning. However, rewording of sentences can result in sentences with different meanings but with high overlap. Current semantic similarity models fail due to limitations of many datasets in this regard, a lack of paraphrase adversaries. PAWS (Paraphrase Adversaries from Word Scrambling) addresses this exact problem by including challenging sentence pairs that are generated by word scrambling. This project aims to create an efficient model for the sentence pair identification task, which performs well on the PAWS dataset.

1 Introduction

Sentence similarity is a significant issue in the natural language processing tasks, and one key point is to find the proper representation of a sentence. A widely used method is the Term Frequency Inverse Document Frequency (TF-IDF) (Robertson and Walker, 1994), which uses the term frequency to asses the word relevance. Later, Latent Semantic Indexing (Deerwester et al., 1990) and Latent Dirichlet Allocation (Blei et al., 2003) was proposed considering the semantic meaning of the words where TF-IDF ignores. In the recent decade, vector space models are proved to represent the textual information very well, such as word2vec (Mikolov et al., 2013), a simple two-layer neural net yet efficient.

(Achananuparp et al., 2008) evaluated different similarity between sentences according to three classes of measures, which are word overlap measures, TF-IDF measures, and linguistic measures. Overall, linguistic measures outperform the other two classes of measures in TREC9 question variants key (TREC9), Microsoft Paraphrase corpus (MSRP), and recognizing textual entailment challenge (RTE3) three data sets. Linguistic measures discussed by (Li et al., 2006) proposed a method to compare two sentences from semantic and syntactic information.

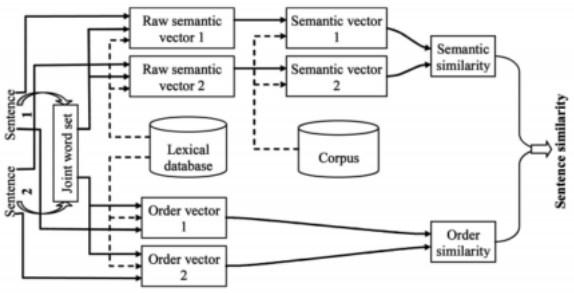

Figure 1: Sentence Similarity Computing Procedure (Li et al., 2006)

Figure 1 shows the procedure for computing the sentence similarity between two candidate sentences. First, a synset for a word in WordNet was used to compute semantic similarity between words by considering path length between two words and depth of each word, denoted as $s(w_1, w_2) = f(l, h)$. Second, the semantic similarity between sentences proposed to form the semantic vectors based on compared sentences dynamically. Weight of each word derived from the information content of a word by computing the probability from Brown Corpus. Furthermore, word order also plays a role in distinguishing two sentences. Thus, a formula for word order similarity measure between two sentences given below:

$$S_r = 1 - \frac{\| r_1 - r_2 \|}{\| r_1 + e_2 \|}$$

where r_1 and r_2 are word order vectors of sentences s_1 and s_2, respectively. Altogether, this method performs well at 0.816 to distinguish two sentences if we take the performance of human 0.825 as the upper bound.

Many machine learning approaches have been explored for sentence semantic similarity. (Viswanathan et al., 2019) compares different traditional methods for semantic similarity on QQP dataset, such as logistic regression, SVM, KNN *etc.*. (Abishek et al., 2019) describes the use of GLOVE embeddings with CNNs to determine the similarity of two sentences. All these methods, however, pale in comparison to the use of Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2018) and its modifications (DistilBERT (Sanh et al., 2019), RoBERTa (Liu et al., 2019)). BERT is used to generate contextual embeddings of sentences, which can be used to determine similarity scores. In benchmarks, a classification layer is attached to the token output, with the two sentences being fed together into the network. There have also been cases where the embedding output of BERT is used to compare sentences through the use of Siamese networks and cosine similarity measures (Reimers and Gurevych, 2019).

2 Dataset Description

There are many datasets to evaluate the performance of sentence semantic similarity models, such as SNLI/ NLI, SentEval, QQP *etc.*. The primary focus will be on QQP, since that is the primary source used in PAWS.

2.1 Quora Quesstion Pair (QQP)

A prominent dataset is QQP dataset (Csernai et al., 2017). This dataset contains over 400k pairs of questions from quora which have been manually labeled. However, a deeper look into the dataset yields some interesting conclusions. A naive way to compare two sentences can be through Bag of Words (BOW) overlap. As noted in (Zhang et al., 2019), the overlap for negative samples is very low. But, there are many examples of sentences where a re-order of words can generate two completely different sentences. A simple example can be the pair *"A goes from X to Y"* and *"A goes from Y to X"*, which have very high BOW overlap but have completely different meanings. This can lead to models trained on QQP to have a very high bias towards word overlap when making a decision.

2.2 Paraphrase Adversaries from Word Scrambling (PAWS)

PAWS (Zhang et al., 2019) dataset contains pairs of phrases generated with high overlap. The dataset is generated from word swapping, which involves rearranging of words, and back translation technique, which involves fixing the grammatical inconsistencies obtained due to word swapping. The whole process is highlighted in Figure 2. Labeling of these examples is done through a combination of manual and language models. This can also assist in generating a dataset that is more balanced between the two classes and can result in a much lower bias for the overlap of words. PAWS also contains pairs generated from Wikipedia sentences. In total, after performing filtering, the dataset contains over 12k pairs generated from QQP and 43k pairs generated from Wikipedia.

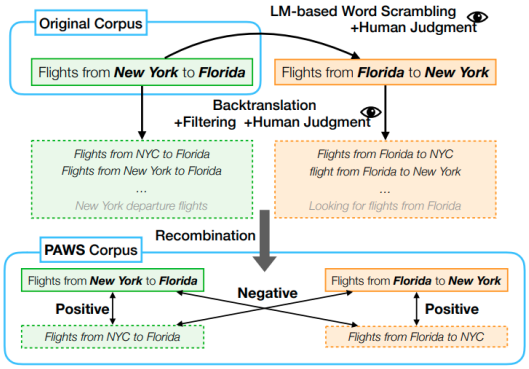

Figure 2: Sample generation from PAWS (Zhang et al., 2019)

3 Method & Results

3.1 Baseline

In the work (Zhang et al., 2019), the performance of different semantic similarity models is compared. Bidirectional Encoder Representations from Transformers (BERT) consistently has the highest performance for the different datasets. The authors use a classification layer on top of the CLS token output to predict the label. To determine the baseline, a similar model provided by HuggingFace transformers library is used (BERT for sequence classification) is used. Further, to shorten the runtime, DistilBERT, which is a smaller and faster version of BERT, is used.

The validation and training splits is obtained from GLUE benchmark website. A tokenizer is

Error Type	Sentence1	Sentence2	Label	Prediction
1	What is the weirdest thing about you? Are you proud of it?	What is the weirdest thing about you?	N	Y
1	Is Jesus the only way to God?	Do Christians believe that Jesus is the only way to reach God?	N	Y
2	Which is more common, a liberal Democrat or a conservative Republican? Why so?	Which is more common, a conservative Democrat or a liberal Republican? Why so?	N	Y
2	What are the benefits of doing an MBA after getting an undergraduate degree in geography?	What are the benefits of getting an MBA after doing an undergraduate degree in geography?	N	Y
3	What is the maximum file size that can be uploaded in Whatsapp?	What is the maximum file size on WhatsApp?	Y	N
3	How do landlords qualify tenants?	How do landlords conduct background checks on tenants?	Y	N

Table 1: Different types of misclassification errors observed.

used before using the data to train the model. The training is stopped when the difference in the accuracy of the validation set between two iterations is minimal. The validation accuracy is checked every 500 iterations, with the model loss being monitored every 100 iterations. A batch size of 32 is used at each iteration. Adam optimizer with the initial learning rate of 10^{-5} is used. The model has been trained for two epochs.

Train vs Test	QQP	$PAWS_{QQP}$
QQP	88.58	31.76
$QQP + PAWS_{QQP}$	88.10	80.50

Table 2: Baseline performance results

Table 2 shows the baseline results generated for the different combinations of datasets. The reported accuracies for QQP was around 90%. The drop in accuracy can be associated with the use of DistilBERT as compared to the full BERT model described in (Zhang et al., 2019). Further, there is a significant improvement in $PAWS_{QQP}$ validation set when trained with $QQP + PAWS_{QQP}$ compared to only QQP in training.

Table 1 shows some of the incorrect classification examples generated by the model. Most of the incorrect classifications can be classified into three categories. The first error type shown in the table is a partial match error type. In this type, there is one of the phrases that overlap between sentences. This error, in general, causes the model to classify two sentences as the same while the two sentences convey different meanings. Second type error happens when one or two keywords in sentences are swapped, and that changes the meaning of two sentences completely. In general, the model will classify the two sentences to have the same meaning, while they are different. The third type of error is that the meaning of sentences are the same, but expressed in different ways. In this case, some additional knowledge beyond the sentences is required.

3.2 Proposed Changes & Analysis

3.2.1 Generating sentence embeddings

We adopt a similar approach to (Reimers and Gurevych, 2019) to generate sentence embedding. Due to the lack of modularity of their codebase, we built our own layers which mimic the actions performed in the work. An average pooling layer is applied to each sentence embeddings, which reduces the size of an embedding from $d_{sentence} \times 768$ to 768. The mask filter is used to ensure that the padding entries do not contribute to this pooling result.

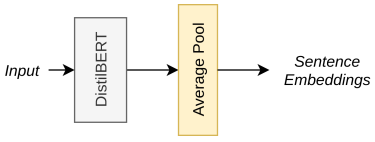

Figure 3: Sentence Embedding Generation

3.2.2 Generating phrase embeddings

Spacy framework is used to generate noun phrase locations in a sentence. Up to 10 phrases in each sentence are considered. This is used to create a filter that filters out certain portions of the word embeddings. These portions are then pooled to form an embedding that represents the phrase. Pooling is performed similarly to how the sentence pooling is done.

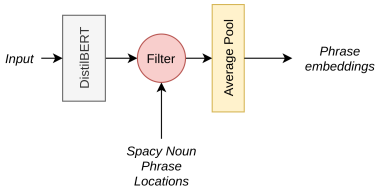

Figure 4: Phrase Embedding Generation

3.2.3 Siamese comparison layers

To compare the embedding results of two sentences, a siamese architecture is adopted. The overall network involves three sections. The first section combines the result from two sentence embeddings, for which the inputs are the two sentence embeddings u and v. Further, as per (Reimers and Gurevych, 2019), we also include $|u - v|$, which is the euclidean distance between the two sentence embeddings. The cosine similarity between the two embeddings is also included.

The second sections involve comparing two phrase embedding sets. A fully connected layer is adopted due to the size of each section and to reduce the impact of empty phrases before merging the two embeddings.

Finally, the two sections are merged in the final section, which has another fully connected layer before a softmax layer. To train the model, the loss is computed using Binary cross entropy loss function, which is prominently used in the case of probabilistic outputs.

3.2.4 Introducing attention

Attention is introduced by adding a multi-head attention layer after the DistilBERT embeddings are generated. The idea is to have a specialized attention layer that is able to specifically direct the comparison to the different phrases.

For sentence embeddings, Figure 6 shows how the new embeddings are generated. The embedding from the encoder is multiplied by linear layers (weights W_k, W_v) to get K and V, respectively.

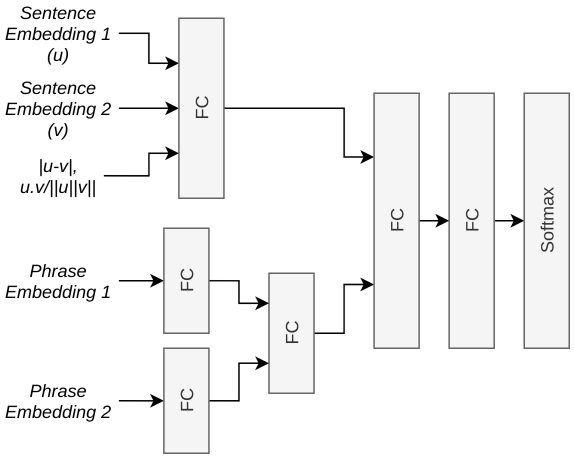

Figure 5: Comparison Network

Two variations of Q are experimented with. One method to generate Q is multiplying the embeddings of the same sentence by W_q. The other option is to use the other sentence embedding for generating Q. The intuition is that the first case would highlight what defines a sentence, whereas the second case can learn to focus on different phrases.

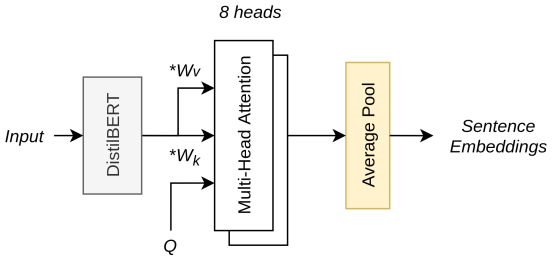

Figure 6: Sentence Embedding Generation with Attention

A similar approach is adopted for phrase embeddings, with filtering being applied to the input corresponding to V. Further, the output is also filtered. The same analysis is also performed for Q.

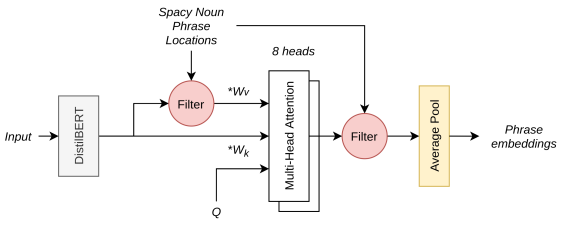

Figure 7: Phrase Embedding Generation with Attention

3.2.5 Analysis of modifications

Table 3 shows the results obtained from training the custom models. Although the performance is

No	Description	QQP	$PAWS_{QQP}$
1	Attention on both sentence and phrase; Cross-Attention; residual skip connection included	87.26%	57.75%
2	Attention on both sentence and phrase; Cross-Attention; No residual skip connection	85.63%	55.83%
3	Attention on both sentence and phrase; Self-Attention; No residual skip connection	84.94%	56.28%
4	Attention on phrase only; Self-Attention; No residual skip connection	85.49%	53.91%
5	No attention; No residual skip connection	86.08%	54.06%

Table 3: Model variations tested

quite close in terms of QQP dataset alone, the performance on $PAWS_{QQP}$ is much lower than the baseline. This could be due to a couple of reasons.

First, when pooling, we are reducing the amount of information that we are dealing with. This could be one of the reasons behind the lower result. A different pooling strategy or making a comparison prior to pooling could be a better approach. However, we will have to compare two 2D representations. For this, we were unable to figure out a suitable method.

Second, the attention layer is designed to capture similarities between two sentences, not the difference. If the layer could be modified to include this, this attention layer can perform a lot better than the original.

Finally, there is a huge disparity between the number of samples between the QQP and $PAWS_{QQP}$ dataset. Due to this, the training time needed for learning the optimal strategy could be much larger than what we used, which was around $2 - 3$ epochs.

4 Issues Faced

- Due to the sheer volume of data, training the model was a lengthy process. Further, there were many instances where the loss was decreasing, but the validation accuracy remained constant, and cases of the other way as well.
- Due to the disparity between the quantity of data the two datasets and among the two classes, the training process was hard to monitor, and the accuracies were not so representative of the performance gains.
- There was also a challenge of figuring out the optimal learning rate, which is able to account for the new layers training as well as not disrupting the pre-trained BERT models.

5 Conclusion

In this work, a baseline has been generated for sentence comparison tasks using DistilBERT. We also build a custom model that aims to address some of the issues faced by the original model. However, the approaches explored do not resolve but only magnify these issues. Hence, a different approach is needed to resolve some of the identified problems.

References

K Abishek, Basuthkar Rajaram Hariharan, and C Valliyammai. 2019. An enhanced deep learning model for duplicate question pairs recognition. In *Soft Computing in Data Analytics*, pages 769–777. Springer.

Palakorn Achananuparp, Xiaohua Hu, and Xiajiong Shen. 2008. The evaluation of sentence similarity measures. In *International Conference on data warehousing and knowledge discovery*, pages 305–316. Springer.

David M Blei, A Ng, and M Jordan. 2003. Latent dirichlet allocation journal of machine learning research (3).

Kornél Csernai, Shankar Iyer, and Nikhil Dandekar. 2017. First quora dataset release: Question pairs.

SC Deerwester, ST Dumais, TK Landauer, et al. 1990. Indexing by latent semantic analysis journal of the american society for information science.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Yuhua Li, David McLean, Zuhair A Bandar, James D O'shea, and Keeley Crockett. 2006. Sentence similarity based on semantic nets and corpus statistics. *IEEE transactions on knowledge and data engineering*, 18(8):1138–1150.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Stephen E Robertson and Steve Walker. 1994. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *SIGIR'94*, pages 232–241. Springer.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Sujith Viswanathan, Nikhil Damodaran, Anson Simon, Anon George, M Anand Kumar, and KP Soman. 2019. Detection of duplicates in quora and twitter corpus. In *Advances in Big Data and Cloud Computing*, pages 519–528. Springer.

Yuan Zhang, Jason Baldridge, and Luheng He. 2019. Paws: Paraphrase adversaries from word scrambling. *arXiv preprint arXiv:1904.01130*.