Computing semantic textual similarity through transformer-based encoders and combining multiple content similarity measures

Stanford CS224N Custom Project

Kayson Hansen
Department of Symbolic Systems
Stanford University
`kayson@stanford.edu`

Peng Hao Lu
Department of Computer Science
Stanford University
`penglu@stanford.edu`

Ethan Ko
Department of Computer Science
Stanford University
`ethanko@stanford.edu`

Abstract

Semantic textual similarity task is an extremely relevant area of natural language processing that we aim to explore in this study. We reproduced a basic miniBert model for sentiment analysis, paraphrase detection, and semantic textual similarity, specifically aiming at exploring possible improvements to this base model for simultaneous textual analysis and improved text embeddings. Our goal is to build upon the work of Bär et al. (2012) by utilizing techniques developed in the field since Bar et al. published their paper in 2012. Namely, we added an embedding generated from a transformer-based encoder model as another feature to see if this will achieve scores better than those achieved in the paper. Additionally, we generated sentence embeddings through the FlagEmbedding model for a total of approximately 770 features. Due to difficulties with efficiency, computing power, and poor NumPy implementation of our features, our accuracies for the full model were suboptimal.

1 Key Information

Custom Project

Mentor: Anirudh Sriram

Team Contributions:

- Kayson: Wrote code for Bert baseline model and adapted HuggingFace transformer-based embeddings.
- Peng: Set up the Google Colab and version control, worked on debugging generally, implement multiclass classifier, and helped write the final report.
- Ethan: Wrote code for the multiclass classifier and additional features such as GST, LCS and Character/Word n-grams. Helped finish the final report.

2 Introduction

Our team felt particularly attracted to semantic textual similarity (STS) due to the wide variety of approaches it could be applied to. STS has many applications such as "question answering, text reuse detection or automatic essay grading" (Bär et al., 2012). The task of not only determining if sentences are paraphrased but also ranking multiple sentences in similarity seemed like a difficult one: one that initially seemed innately human. The tasks requires bidirectional similarity between pairs of sentences. However, previous approaches which implement surface level or semantic features are limited because measures were used exclusively and did not consider any other features beyond only the textual content. Previous attempts to tackle these limitations use a supervised machine learning approach to utilize combined measures, which our team felt that we could build upon.

3 Related Work

Our primary source of inspiration was (Bär et al., 2012). Our team felt that the paper provided many great initial strategies, upon which we can improve. We plan to adopt many of their text similarity measures while also using our own transformer-based encoder model to generate additional features. Bar was limited by the methods available at the time, and we believe that with the new methods that have surfaced today, we can achieve better results using the same approach of combining multiple content similarity measures. Their study also notes a large limitation in the space, which is that content similarity measures are often context-blind. In our approach, we will choose similarity measures to try to approximate context, such as through n-grams, but capturing the full contextual meaning is difficult and may involve things such as time period and culture. Thus, our study is also limited by a lack of "full" context.

4 Approach

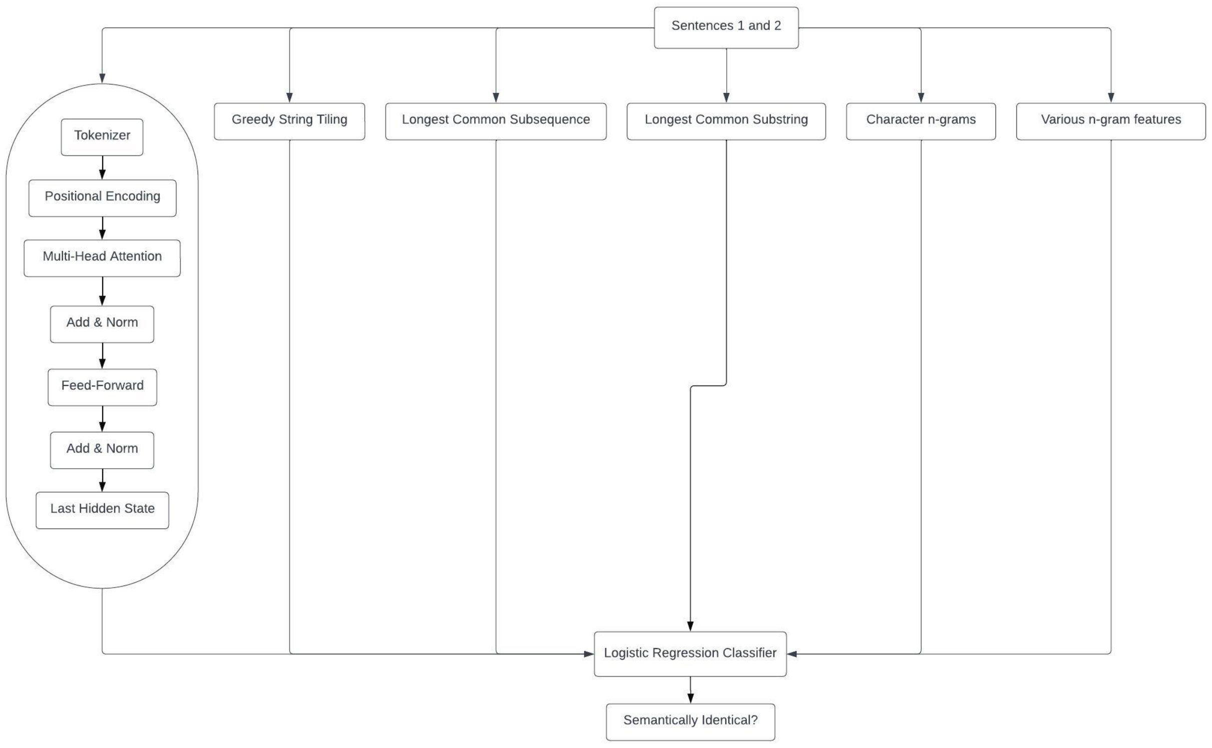
Our main novel contribution in this paper is combining the features used in the original paper Bär et al. (2012) with the new features we have available using transformer-based encodings. From the original paper, we've implemented the Greedy String Tiling, Longest Common Subsequence, Longest Common Substring, Character 2-, 3-, and 4-grams, Word 1- and 2-grams (using containment scores, w/o stopwords), Word 1-, 3-, and 4-grams (using Jaccard similarity), and Word 2- and 4-grams (using Jaccard similarity, w/o stopwords). Unfortunately, we weren't able to use all of the original features from the paper, because the remaining features (Distributional Thesaurus, Pairwise Word Similarity, Explicit Semantic Analysis) all required more datasets that we weren't able to locate due to vagueness in the original paper.

We implemented the following features: Greedy String Tiling, Longest Common Subsequence, Longest Common Substring, Character 2-, 3-, and 4-grams, Word 1- and 2-grams (using containment scores, w/o stopwords), Word 1-, 3-, and 4-grams (using Jaccard similarity), and Word 2- and 4-grams (using Jaccard similarity, w/o stopwords).

Jaccard Similarity:

$$J(s_1, s_2) = \frac{|s_1 \cup s_2|}{|s_1 \cap s_2|} \tag{1}$$

Our model architecture is as follows:



With the implemented features and transformer-based embeddings, we use a logistic regression classifier to determine semantical identity.

5 Experiments

5.1 Data

For sentiment classification, we used the Stanford Sentiment Treebank(SST) developed by Richard Socher and Pott (2013) , which contains 11,855 single sentences and was parsed with the Stanford parser developed by Chen and Manning (2014). This dataset has 215,154 unique phrases which were classified by 3 humans as either negative, somewhat negative, neutral, somewhat positive, or positive. Additionally, we used the CFIMDB dataset , which contains 2,434 movie reviews, each labeled with a binary classification of either negative or positive. For the task of paraphrase detection and exploring transformer based embeddings, we used the first Quora dataset by Shankar Iyer and Csernai (2017) release which contains over 400,000 question pairs each labeled with a binary classification regarding their duplicity.

5.2 Evaluation method

Using the accuracy for the 3 tasks of sentiment analysis, paraphrase detection and semantic textual similarity against a test set, we compared our performance to others in the class(leaderboard).

5.3 Experimental details

We generated sentence embeddings using the FlagEmbedding model from Hugging Face (https://huggingface.co/BAAI/bge-small-en-v1.5). These embeddings were roughly 380-dimensional, meaning that after we concatenated the embeddings for the first and second sentences in the Quora dataset, as well as the additional semantic textual features we generated, we had approximately 770 features.

Our model used a Sigmoid activation layer, standard MSE loss and logistic regression ADAM optimizer techniques. Training was done with 10 epochs on a virtual Google Colab T4 GPU environment.

5.4 Results

Report the quantitative results that you have found. Use a table or plot to compare results and compare against baselines.

Task	Accuracy
Sentiment Classification Test set	0.288
Paraphrase Detection Test set	0.372
Semantic Textual Similarity Test set	0.107
Baseline Transformer embeddings Dev set	0.62

6 Analysis

A drawback of our system was not weighing the semantic features more heavily than the 360-dimensional embeddings. Given how few semantic features there are (about 11), their impact could be overshadowed during training time by the sentence embeddings. Concatenating the embeddings instead of averaging the two also resulted in high dimensionalities. Reducing these dimensionalities or upgrading hardware would allow for faster training and allow for the implementation of more features. We had previously achieved 62% accuracy with only the transformer-based embeddings, suggesting that if we had used matrices or NumPy functions with parallel processing, we would've been able to avoid long generation times for learning the embeddings($\approx$ 2 hours per run).

semantic textual features

7 Conclusion

In this paper, we improved upon a baseline miniBert model using transformed-based encodings and other features. Our implementation achieved significant improvement. A future area of research we would've liked to pursue would be to implement the weighings. Additionally, we would like to use different datasets in order to be able to implement more features, namely Distributional Thesaurus, Pairwise Word Similarity, and Explicit Semantic Analysis. Making our implementation of the semantic textual features more efficient by using NumPy and matrices would improve our debugging times and accuracy.

References

Daniel Bär, Chris Biemann, Iryna Gurevych, and Torsten Zesch. 2012. Ukp: Computing semantic textual similarity by combining multiple content similarity measures. In ** SEM 2012: The First Joint Conference on Lexical and Computational Semantics–Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 435–440.

Danqi Chen and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of EMNLP*.

Jean Y. Wu Jason Chuang Christopher D. Manning Andrew Y. Ng Richard Socher, Alex Perelygin and Christopher Pott. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)*.

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