**CSCE 5290: Natural Language Processing**

**Project Increment**

Group 15: Nikita Lokhmachev, Paul Phillips, Likhitha Gullapalli, Leela Dodda

Project Name: Semantic Textual Similarity Comparison

• **GitHub**

The project GitHub link is: <https://github.com/n1kkqt/nlp_text_similarity>

• **Introduction**

In Natural Language Processing, the task of Semantic Textual Similarity plays a significant role as it has many applications in the field of AI. It has made a large contribution to search engines, anti-plagiarism systems, conversational and commercial chatbots as well as document clustering.1

However, while humans can tell two texts how similar two texts are based on their context, for computers, it is a difficult task. The similarity can be measured by comparing word-level and context-level relationships of texts or sentences. The former can be achieved relatively easily by incorporating heuristics and machine learning algorithms, whereas the latter requires supervised, self-supervised, or unsupervised deep learning models that require a lot of computational resources.

• **Motivation**

The number of heuristics, machine learning approaches, and deep learning approaches that can be combined in different ways to achieve reasonable and satisfactory results is practically limitless, which makes the problem of text similarity a large field of research. Therefore, it is vital to compare several existing methods with each other and potentially come up with our own by combining them.2

• **Significance**

This project has significance in labeling and organizing text documents. If a program can read in multiple text documents and recognize which ones are making similar points, then it can organize these documents under a single label. Recognizing textual similarity is also significant in search engines. If a person wants to use a search engine to look for the answer to a question, then textual similarity can be used to identify documents will answer it, even if the documents word the question in a different way. This project has significance in improving the capabilities of conversational chatbots. People in ordinary conversations can say the same thing in multiple different ways. A program that is made to simulate a conversation will need to be able to do this if the same point needs to be made more than once. Repeating the same point, in the same way, would make the conversation seem mechanical, so the ability to do a semantically similar response with different words would make the conversation feel more authentic.

• **Objectives**

One of the main objectives of the project is to try and improve the model accuracy on the MRPC dataset (mentioned below) in particular and compare the results with other solutions that incorporate state-of-the-art models such as T5, XLNet, and [ERNIE](https://paperswithcode.com/paper/ernie-enhanced-language-representation-with). We would like to see if the ensembling of BERT and different machine learning approaches can improve its accuracy.

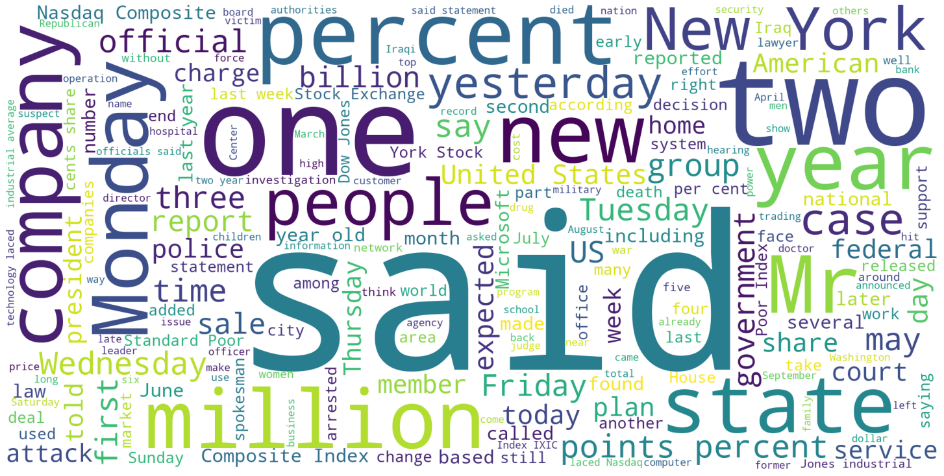
Our team is planning to split into two groups: those who use standard machine learning approaches and those who incorporate deep learning algorithms. Ultimately, since there are 4 members in our team, we should get 3-4 models and try to stack them together to achieve even better results.

• **Dataset**

We chose a relatively small dataset called Microsoft Research Paraphrase Corpus.6 It consists of 5800 pairs of sentences that have been extracted from news sources on the web. Each pair is manually labeled as either having paraphrased or semantic equivalence. Since the dataset is small, it will be easy to test different models on. The training set has 2753 true paraphrase pairs and 1323 false paraphrase pairs, whereas the test set has 1147 true and 578 false pairs.

• **Analysis**

To analyze and visualize the information within the dataset, we created a word cloud of the combination of all texts in it. We removed all stop words beforehand.



• **Related work**

We analyzed 3 papers that evaluated their model performance on the MRPC dataset.

T5 is a transformer that is trained to perform text-to-text transfer. For every task it can perform (such as question answering, translation, and classification), it is trained to receive some text data as an input and output some target text data. This approach enables the model to be trained for a vast set of tasks.3

ERNIE is an unsupervised deep learning model that is trained for word-aware, semantic-aware, and structure-aware text tasks, thereby enhancing its existing knowledge base.4

XLNet is an autoregressive model in which each token is dependent on its previous tokens. The model is generalized since it captures context in two directions using a permutation language modeling mechanism.5

|  | **T5-Small** | **ERNIE** | **XLNet** |
| --- | --- | --- | --- |
| **Accuracy** | 86.6 % | 88.2 % | 90.8 % |

Table 1. State-of-the-art model MRPC accuracy

• **Feature preprocessing**

All the text data has been preprocessed the following way:

1. Remove extra spaces between words and at the ends of sentences
2. Make each word lowercase
3. Convert words/subwords (depending on the algorithm) to their numeric representations
4. In the case of BERT, add [SEP] tokens between texts as well as token type ids and attention masks. Attention masks serve as an indicator what word ids the model has to concentrate on after padding, whereas token type ids are a mask vector that separates two encoded text sequences from each other.7

• **Implementation**

On a high level, the pipeline of our work is the following. For each model, we have to:

1. Get sentence embeddings

2. Train a binary classifier with a sigmoid activation function and a binary cross-entropy loss based on the embeddings (the labels are 0 and 1, texts are not similar and texts are similar respectively)

3. Compare the performance of all 4 models

4. Ensemble all 4 models to potentially increase the algorithm performance as each of the algorithms looks at different text features and requires different preprocessing which might make it more diverse

For our text similarity algorithm, we chose 4 different machine learning/deep learning models to get text embeddings from:

* Term frequency–inverse document frequency (TF-IDF), an algorithm that encodes each word within a document based on its frequency
* Word2Vec, an algorithm that creates an embedding vector for each word within a sentence. To get a sentence embedding, we either sum/average all the words out or use TF-IDF to perform a weighted average. We expect these word embeddings to give better model accuracy compared to TF-IDF
* Doc2Vec, a modified version of Word2Vec that creates embeddings from entire sentences
* BERT, a deep learning transformer-based language model that has already been trained on vast text data. Since it has already been pretrained, we will add a linear classification layer with a sigmoid function to the head of the model and fine-tune it on our dataset. It is expected that this model will give the best results compared to the other methods

After obtaining the text feature embeddings, we will train a simple fully-connected neural network to classify whether two encodings are similar or not. We will also compare and combine the models to see which approach works the best.

• **Preliminary results**

So far, we have fine-tuned the BERT model on the MRPC dataset and trained it to classify whether two texts within the dataset are similar or not. The model validation accuracy is 78.2% so far. We will also consider adding a hard example mining technique.8 It is expected that the model accuracy will increase after ensembling of all 4 models. We have also obtained the sentence embeddings for each of the aforementioned algorithms.

|  | **TF-IDF** | **Word2Vec** | **Doc2Vec** | **BERT** | **Model Ensemble** |
| --- | --- | --- | --- | --- | --- |
| **Accuracy** |  |  |  | 78.2% |  |
| **Confusion Matrix** | |  |  | | --- | --- | |  |  | | |  |  | | --- | --- | |  |  | | |  |  | | --- | --- | |  |  | | | 319 | 259 | | --- | --- | | 117 | 1030 | | |  |  | | --- | --- | |  |  | |

Table 1. Preliminary research results (the table will be updated in the further project increments)

**• Project Management**

* Implementation status report

▪ Work completed:

• Description

We wrote code to get sentence embeddings for all 4 models. Based on the embeddings, we can calculate sentence similarities by computing cosine similarities between them and comparing them with each other. However, to get better results, we will train fully connected neural networks for each algorithm in order for them to classify whether two sentences are paraphrased (similar) or not. We have already trained the BERT Model to perform this task and computed its accuracy and confusion matrix.

• Responsibility (Task, Person)

Nikita Lokhmachev: Get the embeddings of the BERT model, train a binary classifier based on the embeddings

Paul Phillips: Get the embeddings of the Doc2Vec model

Likhitha Gullapalli: Get the embeddings of the Word2Vec model

Leela Dodda: Get the embeddings based on the TF-IDF approach

• Contributions (members/percentage)

Nikita Lokhmachev: 30%

Paul Phillips: 23.3%

Likhitha Gullapalli: 23.3%

Leela Dodda: 23.3%

▪ Work to be completed

• Description

Having written the code to get the embeddings of all 4 models, all we have left to do is train each model to predict whether two sentences are paraphrased or not (similar or not). After that, we will compare the performance of each model with each other as well as with the aforementioned state-of-the-art models.

• Responsibility (Task, Person)

Nikita Lokhmachev: Ensemble all 4 models together and compare their performance with each other as well as with the state-of-the-art models

Paul Phillips: Train a binary classifier based on the Doc2Vec model embeddings

Likhitha Gullapalli: Train a binary classifier based on the Word2Vec model embeddings

Leela Dodda: Train a binary classifier based on the TF-IDF approach embeddings

• Issues/Concerns

One of the main concerns about this project that we have is that the ensembling of all 4 models might not be able to improve the overall accuracy of the system. However, this experiment needs to be conducted regardless.

• **References**

[1] Ganesan, K. (2021). What is text similarity? *The Business Case for AI.* Retrieved from <https://kavita-ganesan.com/what-is-text-similarity/#.YUTYsbhKhPY>

[2] Sieg, A. (2018, Jul 4). Text Similarities: Estimate the degree of similarity between two texts. *Medium.* Retrieved from <https://medium.com/@adriensieg/text-similarities-da019229c894>

[3] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J. Liu: “Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer”, 2019; [arXiv:1910.10683](http://arxiv.org/abs/1910.10683). <http://arxiv.org/abs/1910.10683>

[4] Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, Hua Wu: “ERNIE: Enhanced Representation through Knowledge Integration”, 2019; [arXiv:1904.09223](http://arxiv.org/abs/1904.09223). <http://arxiv.org/abs/1904.09223>

[5] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, Quoc V. Le: “XLNet: Generalized Autoregressive Pretraining for Language Understanding”, 2019; [arXiv:1906.08237](http://arxiv.org/abs/1906.08237). <http://arxiv.org/abs/1906.08237>

[6] (2005, Mar 3). Microsoft Research Paraphrase Corpus. *Microsoft*. <https://www.microsoft.com/en-us/download/details.aspx?id=52398>

[7] (2020). Huggingface Glossary. *Huggingface.* <https://huggingface.co/transformers/glossary.html>

[8] Chengyue Gong, Tongzheng Ren, Mao Ye, Qiang Liu (2020). MaxUp: A Simple Way to Improve Generalization of Neural Network Training. *Arxiv.* <https://arxiv.org/pdf/2002.09024.pdf>