For this project, I used the same features as my midterm project, as it was the same task of paraphrase identification. Since I had received good results on the midterm dataset, I used those features for this project. Like the midterm, the majority of the features that I utilized were found in the literature related to this task. The primary article in the literature that I utilized for the midterm was the paper *Re-Examining Machine Translation Metrics For Paraphrase Identification*. In this paper, various metrics were identified, such as BLEU1-BLEU4, NIST1-NIST5, TER, and METEOR. These metrics work to quantify the accuracy of the translation of a sentence, which makes them useful for paraphrase identification. Also, I used a metric known as the Levhstein, or edit distance. Instead, the number of edits it takes to transform one sentence into another works better as an indicator of sentence meaning than sentence length, as two sentences with similar lengths but different words would have an equal distance but a dissimilar edit distance. In the paper *Low-Level Features For Paraphrase Identification,* edit distance was also discussed as a potential metric for paraphrase identification tasks.

In terms of data preparation, I preprocessed the text similar to the midterm project since the final project was the same task. For instance, I corrected some syntactical errors during my preprocessing and eliminated the periods and commas from these sentences before they were processed. However, unlike the midterm project, I did not augment the data distribution. While I experimented with copying the data to be a 50/50 split or a 25/75 split (the same distribution of the test), my dev results received slightly better results when the original training data was used. Due to this, I did not change the original distribution of the testing data. On top of this, another difference occurred in that I used a StandardScaler to scale my data for this project, while I used a MinMaxScaler to scale my data in my midterm project. While I initially started with a MinMaxScaler, the dev results were higher when a StandardScaler was used.

In terms of the libraries that I used, I utilized libraries that I used in the midterm project, but I also used new libraries. Like the midterm project, I used the NLTK, pyter, and Levenshtein libraries to generate my features for the project. The NLTK library generated the majority of the features for the project. In contrast, the pyter library generated the TER metric, and the Levenshtein library generated the edit distance metric. After generating these metrics, I utilized the CSV library to convert these results into CSV files with raw data. After this, the pandas library was used to convert the train, dev, and test CSV files into their respective dataframes. After this point in the project, I extensively used the PyTorch library and its various functions and objects to create datasets that can be used by dataloaders along with building a neural network. This library also assisted in the creation of my loss function and optimizer, along with helping to build training and testing functions. I also used the PyTorch Library to save models for various epochs so they could be reloaded and utilized to generate a text file with my predictions. In terms of the actual model I built, I built a simple Multi-Layer-Perceptron network. It takes 12 inputs (the base features) and transforms them into an additional 12 nodes, then applies the ReLu activation function to these nodes. This linear layer also has a dropout value of 0.25. Then, another linear transformation is done on these 12 new nodes to create an additional 12 nodes to which the ReLu activation function is applied to these nodes. This linear layer also has a dropout value of 0.25. Then these 12 new nodes are placed through another linear layer that results in one node. The optimizer I used was an SGD optimizer with a learning rate of 0.01. For this task, I used a built-in loss function in PyTorch titled BCEWithLogitsLoss, in that this loss function not only computes the binary cross entropy, but it also implicitly passes the result of the network through a sigmoid and computes the loss based on this value. I used my local CPU for all computations, and I used a batch size of 12. However, to get the actual predictions for my network, I had to manually pass the result in the last node through a sigmoid function, and if this new value is greater than 0.5, the predicted value is one. If this new value is less than or equal to 0.5, the predicted value is 0. After running this model through many epochs, the best result on the development set was the 97th epoch with an f1-score of about.803.

From doing this final project, I gained a large amount of experience. For instance, I have never had experience designing a neural network from scratch and implementing it. I also have never had experience with this extensive use of the Pytorch library. I also enjoyed the challenge of working with imbalnced data, as I have worked with imbalanced data before, but this project enhanced my understanding of it. I also enjoyed learning how to improve on a prior machine learning task. For all the other machine learning projects that I have completed, I have never utilized work done in prior projects and furthered my progress on a similar and related task. Utilizing the same features from my midterm project allowed me to do this and I am happy that I had this experience. Also, unlike the sklearn library, I had no very little prior experience with Pytorch, and I uncovered many online tutorials to help me understand the basics of Pytorch. Through the use of this project, I have learned various new resources to help me understand and use Pytorch in the future. In total, this project gave me knowledge with building and generating predictions from a neural network and I am excited to utilize this knowledge in future projects.

Works Cited

Bird, Steven, Edward Loper and Ewan Klein (2009), *Natural Language Processing with Python*. O'Reilly Media Inc. (NLTK library)

Chroma Code ASMR. (2021, December 1). *ASMR coding: Teaching you AI | neural network with PyTorch | Binary Classification*. YouTube. Retrieved November 30, 2022, from https://www.youtube.com/watch?v=G9EcnReMoYM

Madnani, Nitin, Joel Tetreault, and Martin Chodorow. "Re-examining machine translation metrics for paraphrase identification." *Proceedings of the 2012 conference of the north american chapter of the association for computational linguistics: Human language technologies*. 2012.

Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011.

Pronoza, Ekaterina, and Elena Yagunova. "Low-level features for paraphrase identification." *Mexican International Conference on Artificial Intelligence*. Springer, Cham, 2015.