Deep Learning (COSC 2779) – Assignment 2 – 2023

Wing Hang Chan(s3939713)

1 Problem Definition and Background

The task of stance classification in tweets involves categorizing expressions into "favor," "against," or "neutral" stances regarding specific topics known as "targets." This is distinct from sentiment analysis, which assesses sentiments like "positive" or "negative." Notably, a single tweet can express different stances toward different topics, while sentiment analysis often evaluates sentiments in the context of a single statement. This task was part of the Semantic Evaluation Exercises in 2016[1], where several teams participated. The winning team, MITRE[2], achieved a macro F1 score of an average target at 67.82. They had their own pre-trained model with a substantial number of tweets. Therefore, setting a target of 65 for this task is reasonable, as it exceeds the performance of many participants in the 2016 competition.

In tweets, hashtags, mentions, and retweets are common elements. Hashtags, denoted by the "#" symbol, are word or phrase tags without spaces and often use camel case to differentiate words. Mentions are created by using "@" followed by a username [4]. Retweets involve reposting others' content, and it's vital to note that there isn't an official command [5], such as starting a tweet with "RT," to indicate a retweet; doing so without adding personal opinion can lead to data leakage.

The dataset used for this classification task comprises 2,914 records in the training set and 1,956 records in the testing set, each with five data columns. Two of these columns, "opinion towards" and "sentiment," are excluded from the classification. The training dataset includes five targets (figure 1): Atheism, Climate Change is a Real Concern, Feminist Movement, Hillary Clinton, and Legalization of Abortion. An additional target, Donald Trump, is introduced in the test set (figure 2). Notably, the dataset exhibits an imbalance in the number of tweets for each target, with "Climate Change" having the fewest tweets. The distribution of stances toward each target is also significantly skewed (figure 3), with some targets having a large majority of tweets leaning toward a particular stance. This bias poses a considerable challenge for the classification problem, making it difficult to address through class weight balancing or downsampling. As a result, oversampling with augmentation is being considered as a potential solution to tackle this issue.

2 Evaluation Framework

Before applying data augmentation, it is essential to perform data pre-processing. In the dataset, there are over 90 Retweets (RTs). We discovered one RT in the test set and three in the training set that reposted tweets. Instead of removing all RTs, it's necessary to retain them for comparison with original tweets in both the training and testing sets. In this process, those four RTs were removed.

For data augmentation, it's advisable to split the training and validation sets before pre-processing. While there is another dataset for testing, it's worth considering a larger validation set due to the bias in the training data. However, given the data bias in the training set, a 70-30 split between the training and validation sets is preferred.

To prepare the text data for machine learning, tokenization is essential. Several tokenizers are available, including TweetTokenizer from NLTK and Tokenizer from Keras. We chose to compare the two and found that Tokenizer from Keras, which filters out all punctuations by default, is more suitable. However, it also removes important symbols like "#" and "@." While you can customize the filter, it treats symbol combinations like ":-)" as separate tokens, and it includes commas or periods within tokens. TweetTokenizer, on the other hand, retains all punctuations and treats hashtag and mention symbols as tokens. It also groups symbol combinations into single tokens, making it more intuitive. We chose TweetTokenizer and manually filtered out single punctuation tokens.

Hashtags can provide valuable information for stance classification, as people often express their emotions and opinions using hashtags. However, hashtags are single phrases without spaces, making them challenging for machines to interpret. While character-based learning is possible, it can be resource-intensive. Therefore, we applied Pascal Case pre-processing to hashtags.

Mentions in tweets typically do not carry a direct meaning for stance classification. For example, tweets mentioning a user like "@HillaryClinton" could express either support or opposition, depending on the surrounding words, rather than the mention itself. Therefore, retweet symbols and mentions are removed as they

do not significantly influence the stance classification. Retweet symbols do not convey additional meaning to the tweet, and usernames are not a primary factor in determining stance. The goal is to classify stance based on the content of a single tweet.
Data augmentation is a valuable technique for model regularization and addressing data imbalance through oversampling. nplaug offers various augmentation methods, including synonym augmentation, contextual word embeddings augmentation, and back-translation augmentation. We chose SynonymAug as it applies semantic-based augmentation to textual input, selecting words with similar embeddings. While it may run out of synonyms and repeat words for frequent sentences, it is computationally efficient. ContextualWordEmbsAug, based on contextual word embeddings, allows word substitution or insertion. However, it requires substantial processing time and resources due to the use of transformer models and it performs not very well as it is based on GPT2, not the latest GPT4. BackTranslationAug translates text to another language, such as German or Japanese, and back-translates it to English. It may also face limitations in terms of available languages for translation. Given our computational resources, we opted for synonym augmentation.
Further pre-processing steps involve converting text to lowercase and creating a word index for the model.
For NLP tasks, there are various model choices, including transformer models like Bidirectional Encoder Representations from Transformers (BERT) and recurrent neural networks (RNN). BERT is designed for handling long-range dependencies and can outperform RNN, but it requires significantly more computational resources. Considering that tweets are relatively short (limited to 280 characters [3]) and our available computing resources, we opted for an RNN with word embeddings.
There are several word embeddings available, such as GloVe, Word2Vec, and FastText. While GloVe and Word2Vec are memory-efficient, they may struggle with handling out-of-vocabulary (OOV) words. In contrast, FastText can provide embeddings for OOV words, which is particularly advantageous for social media data that often includes slang, new words, and misspellings. GloVe and Word2Vec cannot provide representations for words not in their corpus, which can lead to difficulties in generating meaningful representations for such words, potentially impacting model performance. Another solution is training custom word embeddings.

3 Approach & Justifications

The model consists of three inputs: "Tweet," "Hashtag," and "Target," forming a multi-input-single-output architecture.
In the preprocessing stage, input Tweet has mentions and hashtags removed, after which it is embedded using FastText and passed through two bi-directional LSTM layers. Input Hashtag involves special preprocessing, where all hashtags are combined as additional information. This approach aims to enhance the model's analytical capabilities. It uses both a bi-directional LSTM and a standard LSTM. For input Target, which consists of single words or phrases, as there are only five targets in the training set and they are not full sentences, a single LSTM layer is considered sufficient (figure 5).
Since this is a model with distinct characteristics for each input, LSTM and bidirectional LSTM layers are applied to each input. These three sub-models are then combined, and two fully-connected dense layers with dropout regularizations are used for further processing.
To prevent overfitting, various regularization techniques are employed, particularly due to the relatively small dataset. Each LSTM layer includes two dropout mechanisms: one from its input (dropout) and the other from its previous output (recurrent_dropout), both set at 80%. Additionally, two dense layers utilize 50% dropout.
For the task of multi-class classification, a sigmoid activation function is used. The Adam optimizer, with a default learning rate of 0.001, is chosen as it is considered safe and effective for NLP tasks. The loss function is categorical cross-entropy, which suits multiclass classification. Despite the balanced training dataset achieved through augmentation, the model is evaluated using the macro F1 score since the test set is imbalanced. This approach ensures that all stances are equally important during evaluation.
The base model delivers a respectable performance with a macro F1 score of 58.96 on the validation set. However, it's plagued by an overfitting issue, as evidenced by the increasing gap between training and validation loss, as illustrated in the loss graph (figure 7). While the gap is not substantial, further regularization techniques may be needed to address it.

4 Experiments & Tuning

Addressing the overfitting and suboptimal performance observed in the base model, we have introduced fine-tuned models as a solution. The first fine-tuned model primarily targets the overfitting issue. In this iteration, the dropout rates in the LSTM layers, recurrent dropout, and dropout in the fully-connected layers have been increased by 10%. However, a 90% dropout rate might be excessively high, which appears to hinder the model's ability to achieve a good F1 score. This model exhibits early stopping at epoch 27 (figure 6) and achieves the highest result of 52.04. Although this model demonstrates some improvement in addressing the overfitting issue (indicated by the reduced loss gap compared to the base model), the F1 score is significantly lower than that of the base model.

To further enhance model performance, a second fine-tuned model with a more complex architecture is introduced. Building upon the base model, this second model incorporates an additional fully connected layer with 32 units to extract higher-level features. Furthermore, it includes one more bidirectional LSTM with 32 units for input Tweet and changes the LSTM layer for input Hashtag to a bidirectional LSTM. The sub-model for input Target remains unchanged, as the data for this component is relatively simple. The increased complexity of this model may require additional computational resources and time. Therefore, the number of training epochs is reduced to 20. This model has a performance of macro F1 score 48.59 (figure 7) and its training and validation loss do not have an increasing gap (figure 6), indicating that there is no overfitting issue for the first 20th epoch. However, it takes more time and resource to train. Therefore, base model is the most suitable model although there is sign of overfitting. The loss and score of bases are better than fine tune model version 1 & 2. Base model early stops at epoch 49 to prevent further overfitting.

5 Ultimate Judgment, Analysis & Limitations

The test result for all targets is 45.11 (figure 8 & 12), which falls far short of the target of 65. There are several reasons for this. Firstly, there are tweets targeting topics that do not exist in the training set. If we exclude the target "Donald Trump," the macro F1 score improves to 50.85 (figure 9 & 12). While this is an improvement, it still does not reach the target goal. It could be guessed that the target "Donald Trump" has the lowest F1 score, which is 31.69, because it does not appear in the training set. There is no word index for this target as word indexes are built from the training set inputs. Additionally, his opponent in the 2016 presidential election, Hillary Clinton, has the second-lowest F1 score of 37.94. The model predicts many "against" labels for Hillary Clinton. This is because there are many "against" references to Hillary Clinton in the training set, causing the model to skew against her (figure 10). For the "against Climate Change is a Real Concern" stance, there is only 1 true positive, with 5 true negatives and 10 false positives (figure 11). The model's performance is very poor because there are very few real data points in the training set. Even though they have been augmented, it appears to be insufficient to handle this biased data.

Apart from the issue of unseen targets, another problem is that the model struggles with new words. If there are any new tokens in the test set for instance target Donald Trump, the model cannot recognize them, as the word index is based on the training set alone. Even though we use Fasttext for the embedding, any tokens beyond the training set are treated as 0. Consequently, the model does not understand the meaning of these new words. Using other word embedding could be improve the model performance but cannot solve OOV issue.

Add more training data must help but data has cost. Putting mention could provide insight for the model. Another solution to this problem is to use more advanced augmentation techniques. nplaug provides sentence-level augmentation based on GPT-2, which has shown poor performance. It tends to generate meaningless words at the end of sentences without making meaningful changes to the existing words. The model could significantly benefit from the application of a transformer model like BERT. It's worth noting that the latest technologies, such as GPT-4, and developments by companies like Google, are also leveraging transformer models for improved performance. This advanced augmentation with transformer models could be a promising avenue to address the challenges faced by the current model.

Finally, there are 25 data points that have been manually collected from Twitter, with 5 data points for each target. While these data points do not come with labelled stances, it's interesting to analyse some of these samples. For the target "Hillary Clinton," there is a tweet saying, "We still love you #HillaryClinton!" is predicted as "none" with a confidence of 63.59%, "against" at 55.54%, and "favor" at only 37.38%. This indicates that the model has significant room for improvement.

6 References:

[1] S. M. Mohammad, "SemEval-2016 Task 6: Detecting Stance in Tweets,"
[2] G. Zarrella, “MITRE at SemEval-2016 Task 6: Transfer Learning for Stance Detection”
[3] X Corp. - "Counting characters", online available: https://developer.twitter.com/en/docs/counting-characters
[4] X Corp. - "Help with username registration", online available: https://help.twitter.com/en/managing-your-account/x-username-rules
[5] X Corp. - "RePost FAQs", online available: https://help.twitter.com/en/using-x/repost-faqs

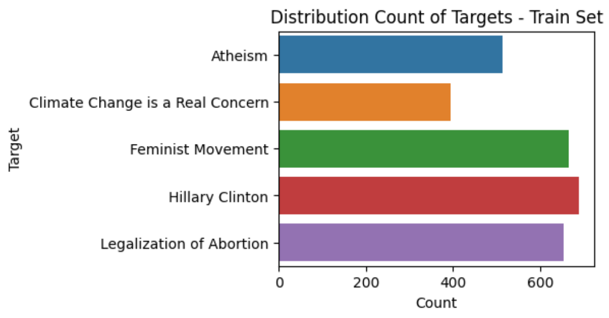


Figure 1 Distribution Count - Train Set

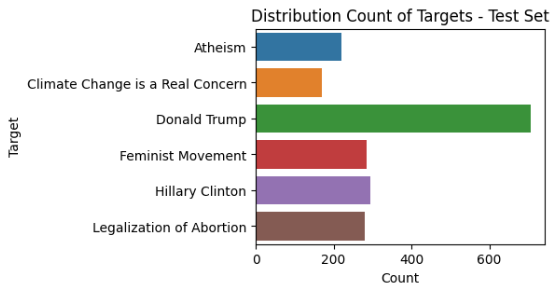


Figure 2 Distribution Count - Test Set

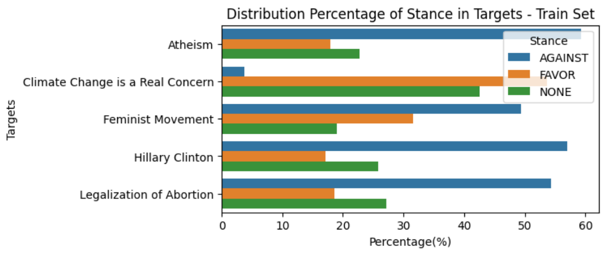


Figure 3 Distribution Count by Targets - Train Set

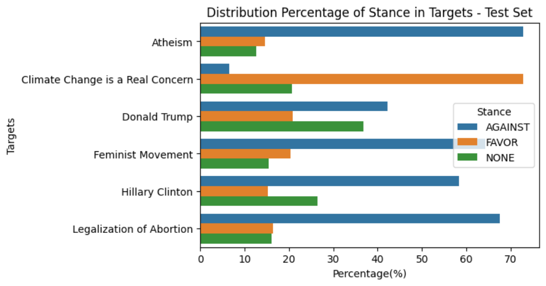


Figure 4 Distribution Count by Targets - Train Set

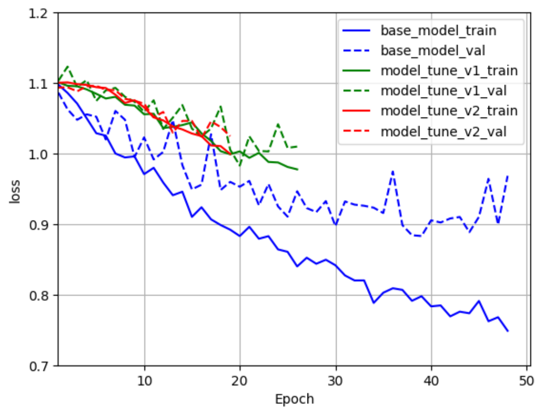
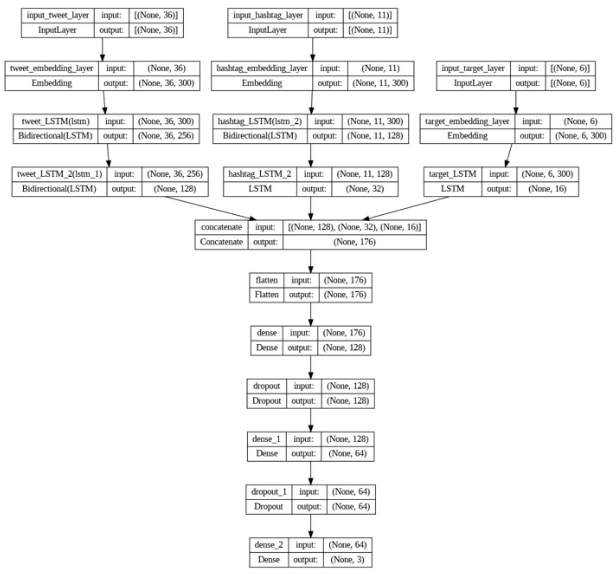


Figure 6 Categorical Crossentropy loss for 3 models



input_tweet_layer InputLayer input: [(None, 36)] output: [(None, 36)]
input_hashtag_layer InputLayer input: [(None, 11)] output: [(None, 11)]
tweet_embedding_layer Embedding input: (None, 36) output: (None, 36, 300)
hashtag_embedding_layer Embedding input: (None, 11) output: (None, 11, 300)
input_target_layer InputLayer input: [(None, 6)] output: [(None, 6)]
tweet_LSTM(lstm) Bidirectional(LSTM) input: (None, 36, 300) output: (None, 36, 256)
hashtag_LSTM(lstm_2) Bidirectional(LSTM) input: (None, 11, 300) output: (None, 11, 128)
target_embedding_layer Embedding input: (None, 6) output: (None, 6, 300)
tweet_LSTM_2(lstm_1) Bidirectional(LSTM) input: (None, 36, 256) output: (None, 128)
hashtag_LSTM_2 LSTM input: (None, 11, 128) output: (None, 32)
target_LSTM LSTM input: (None, 6, 300) output: (None, 16)
concatenate Concatenate input: [(None, 128), (None, 32), (None, 16)] output: (None, 176)
flatten Flatten input: (None, 176) output: (None, 176)
dense Dense input: (None, 176) output: (None, 128)
dropout Dropout input: (None, 128) output: (None, 128)
dense_1 Dense input: (None, 128) output: (None, 64)
dropout_1 Dropout input: (None, 64) output: (None, 64)
dense_2 Dense input: (None, 64) output: (None, 3)

Figure 5 Base Model Architecture

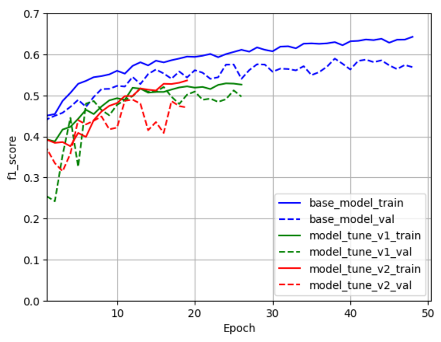


Figure 7 F1 Score for the 3 models

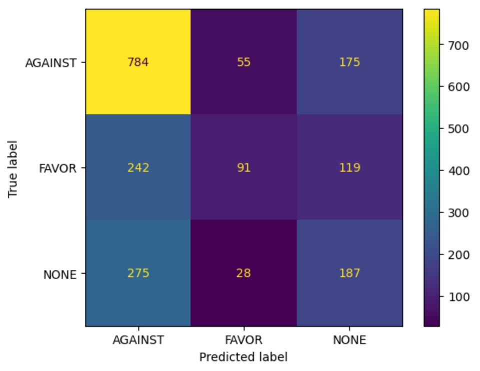


Figure 8 Confusion Matrix of All test results

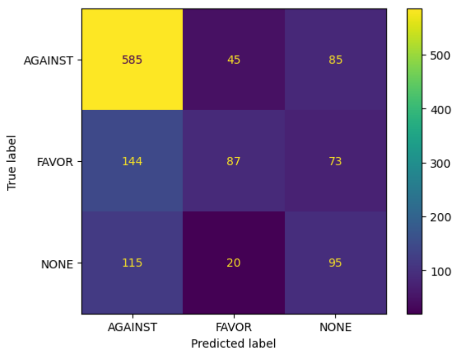


Figure 9 Confusion Matrix of test results without Trump

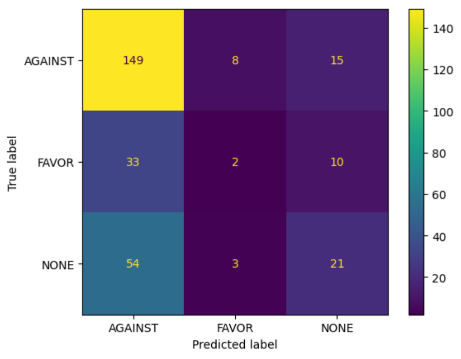


Figure 10 Confusion Matrix of test results Hillary Clinton

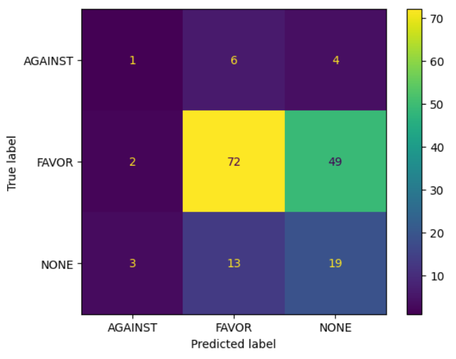


Figure 11 Confusion Matrix of test results Climate Change

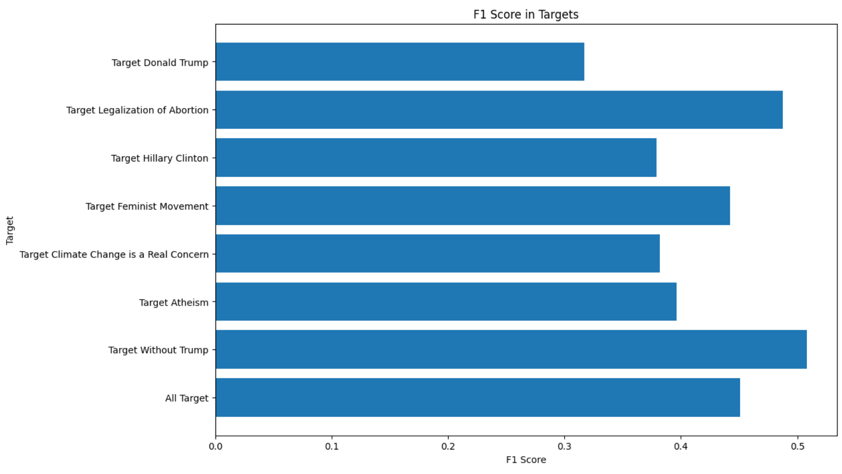


Figure 12 Marco F1 Score for Different Targtes