LING 573: Affect Classification Report

Ben Cote
University of Washington
bpc23@uw.edu

Madhav Kashyap
University of Washington
madhavmk@uw.edu

Lindsay Skinner
University of Washington
skinnel@uw.edu

Keaton Strawn
University of Washington
kstrawn@uw.edu

Allan Tsai
University of Washington
yltsai@uw.edu

Abstract

This paper describes our system for Task 5 of SemEval-2019: HatEval: Multilingual Detection of Hate Speech Against Immigrants and Women in Twitter. The main purpose of this shared task is to conduct hate speech detection on tweets, which mainly targets two specific groups of people: immigrants and women. This takes place across two tasks: a binary classification of hate speech (Task A), and a classification of hate speech tweets as targeted and aggressive (Task B). To address these tasks, we developed a system that utilized word embeddings and various other features with random forest, logistic regression, and SVM classifiers, ensembled to allow weighted voting, as well as one neural method. In this paper, we present the results obtained for both Subtask A and Subtask B in English and Spanish data.

1 Introduction

With the growing popularity of social media, microblogging platforms like Twitter provide a medium for people to communicate with each other using short texts. While these platforms can be used to share users' memorable personal events or constructive opinions on certain topics, some people may use them to propagate their hatred against an individual, a group, or a race. Hence, it becomes crucial to come up with automated and computational methods to identify hate speech on social media platforms.

While there is an increasing number of research dedicated to combatting the issue of hate speech on social media platforms, there are still a lot of problems that remain unsolved. This is mainly caused by the fact that there is no widespread agreement on what constitutes hate speech. One definition considers hate speech as "language that is used to express hatred towards a targeted group or is intended to be derogatory, to humiliate, or to insult the members of the group" (Davidson et al., 2017). Another describes hate speech tweets as those containing racist or sexist comments (Waseem, 2016; Waseem et al., 2017).

The shared task 5 of the SemEval-2019 workshop (Basile et al., 2019) defines two subtasks for detecting hate speech against immigrants and women on Twitter. Both subtasks contain tweets in English and Spanish. In Subtask A, the system has to predict whether a tweet, with a given target, is hateful or not. In Subtask B, the system has to determine whether a given hateful tweet is aggressive or not and whether it targets an individual or a group.

The rest of the paper is structured as follows. After this introductory section, we provide a brief description of the subtasks in Section 2. In Section 3, we give an overview of our system. In Section 4, we provide a detailed approach to developing our system. In Section 5, we show the results of our system. Section 6 contains the discussion and summary. We conclude in Section 8.

2 Task Description

Drawing from the Semeval-2019 shared task 5 [1], this project develops a binary affect classification system aimed at Multilingual detection of hate speech against immigrants and women in twitter (Basile et al., 2019). The data is text-based [2], and comes from shared task 5 in the form of pre-scraped and pre-labeled tweets, classified with either a 1 or a 0 on three metrics: Hate Speech, Personal Target, and Aggression. Hate Speech (0 - hate speech not present, 1 - hate speech present) refers to whether

[1]The shared task CodaLab page can be found here or at this URL: https://competitions.codalab.org/competitions/19935#learn_the_details

[2]The data is not available on the shared task CodaLab page, but it is available on GitHub here or with this URL: https://github.com/cicl2018/HateEvalTeam

or not the target tweet disparages a person or a group on the basis of some characteristic such as race, color, ethnicity, gender, sexual orientation, nationality, religion, or something else. Personal Target (0 - general group, 1 - specific individual) refers to tweets that do have hate speech, and reflects whether the hateful tweet is targeting a group of people generally, or whether there is a specific person being attacked. Aggression (0 - no aggression, 1 - aggression) refers to tweets that do have hate speech, and reflects whether the tweeter is aggressive or not. For our primary task, our system will only be using the English data to classify English tweets on the three binary metrics listed above. For our adaptation task, we adjust our system in order to classify tweets in both English and Spanish on the three binary metrics listed above. In both cases, evaluation is done by comparing our system's classification of each tweet with the prelabeled classifications. Precision and recall of the classification will be amalgamated into a macro-averaged F1-score. Additionally, we calculate an Exact Match Ratio (EMR), the metric used by the shared task to rank submissions from most- to least-effective at hate speech detection.

3 System Overview

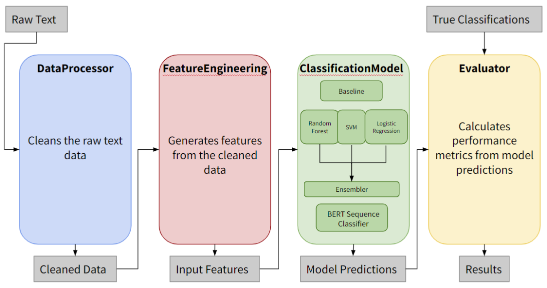

Figure 1: A graphical depiction of our Hate Speech Classification system

Figure 1 shows the flow of our system, from input configuration through the Hate Speech system, outputting both a dataframe of features and classification predictions, and an evaluation of those predictions using precision, recall, accuracy, and F1 scores. In this system, once the input tweets have been processed, each tweet is treated as the input for the Hate Speech Classification System. Features are selected from these tweets to use as the basis for classification, the model is trained, and then the development data is classified using the trained model according to the methods specified within the configuration file. The classification predictions are then used to evaluate the model's performance. The current implementation of this system involved pre-training the model such that the run time is significantly reduced. Training the model takes approximately four and one half hours. Running the test data through the model takes between four minutes and two hours, depending on the combination of features used and the classifier model. The file size of the pickled training data is 1.3 GB, and the size of the pickled validation data is 1.7 MB.

4 Approach

Our approach has four major components: (1) Data Processing; (2) Feature Engineering; (3) Classification; and (4) Evaluation. This document reflects the state of the system as of its current implementation (D2). Each component is detailed below.

4.1 Data Processing

The DataProcessor is used to pull and clean the raw data. The data is read in from .csv files that are either saved locally, or downloaded from GitHub. The raw data comes from two separate .csv files, separated into training and validation data. Each file has five columns: id, text, HS, TR, AG. 'id' contains the unique identification numbers for each tweet; 'text' contains the raw text of the tweet (this includes URLs, hashtags, emojis, references to twitter accounts, slang and misspellings, etc.); HS is a binary (0 or 1) tag that indicates whether (1) or not (0) the tweet is classified as hate speech; TR is a binary tag that indicates whether the tweet is targeted at an individual (1) or a general group (0); AG is a binary tag that indicates whether (1) or not (0) the tweet is aggressive.

The cleaning process separates URLs and hashtags from the text and replaces emojis with English descriptions. It also calculates and adds a feature indicating what percentage of the text is capitalized, before lower-casing the cleaned text. Similarly, the cleaning process generates counts for special punctuation symbols, before removing all punctuation from the cleaned text. Additionally, Twitter ID references are stored in a separate feature and replaced with the string 'user' in the cleaned text. We also implemented spellchecking with the python spellchecker library, in order to replace typos with best guesses of correctly spelled words, as this is twitter data and includes typos and common abbreviations which can be expanded via spellchecking. This library worked for both English and Spanish data, by tweaking a parameter.

4.2 Feature Engineering

The FeatureEngineering class takes in the output of the data cleaning process and creates an expanded dataframe with additional columns for each new feature that is generated. This class contains two main methods. First is the fit_transform method, which intakes the training data in order to train the feature-generating helper methods, wherever such training is required. The second main method is the transform method, which uses the trained helper methods from an earlier call to fit_transform in order to transform the validation and test datasets to include the complete list of generated features for each dataset. The outputs of the fit_transform and transform methods will be transformed dataframes that contain additional fields for each of the features mentioned below.

4.2.1 NRC Counts and Extension

This feature uses the NRC Lexicon (NRCLex) of emotional words (Mohammad, 2018). This feature involves the binary classification of English words across eight emotional dimensions (anger, anticipation, disgust, fear, joy, sadness, surprise, trust), in addition to a positive dimension and a negative dimension. Raw counts are normalized across tweets. For the Spanish adaptation, two methods were implemented. First, a roughly-equivalent sentiment word lexicon for Spanish was used, we saw little overlap in the content of Spanish training data and the words in the lexicon, leading to poor performance. Therefore, another method was created to translate Spanish tweets into English for this feature so that the standard NRCLex emotional database could be used. We additionally added a Spanish Sentiment Score module to perfrom a similar task, described in section 4.2.8 below.

An extension was also developed to further the scope of emotional feature identification. All the words from the NRCLex database were used to train a logistic regression classifier to predict the emotion and valence scores of all words represented in an embedding dictionary: GloVe for English, and Spanish BERT for Spanish.

4.2.2 GloVe Embeddings

This feature calls on pretrained GloVe style embeddings (Pennington et al., 2014) from GloVe Twitter to create word embedding representations for the cleaned text. The embedding dimension is determined by the version of glove.twitter that is specified by the embedding_filepath (sizes between 25 and 300 are available to download, we defaulted to low dimensional d=25 embeddings). Embeddings are generated for each word in the text and then aggregated (using the component-wise mean) to generate a proxy for a sentence-level embedding that is stored in Aggregate_embeddings.

For our adaptation task we used 300 dimension GloVe trained embeddings that were trained on the Spanish Billion Word Corpus (Cardellino, 2016).

4.2.3 Google Universal Sentence Encoder Embeddings

This feature uses the tensorflow hub to obtain sentence level embeddings generated by Google's Universal Sentence Encoder (Cer et al., 2018) using the cleaned_text column of the preprocessed dataframe. Embeddings are not aggregated because they are already sentence level. The embeddings are d=512.

4.2.4 Spanish Sentence Encoder

For Spanish sentence level embeddings we used Hugging Face's sentence-transformers library to encode Spanish sentences with the model hiiamsid/sentence_similarity_spanish_es. This created d=768 sentence level embeddings, which we used in the same way as the English sentence embeddings.

4.2.5 BERTweet Embeddings

This feature uses pytorch to generate d=768 word level embeddings for each word in the cleaned_text column of the processed dataframe. BERTweet embeddings (Nguyen et al., 2020) are trained on Twitter data, giving us an advantage of domain specificity for our task. Like in the case of the GloVe embeddings (Pennington et al., 2014), representations are generated for each word in the text and then aggregated (using the component-wise mean) to generate a proxy for a sentence-level embedding that is stored in Aggregate_embeddings.

4.2.6 TwHIN-BERT Embeddings

For Spanish, we needed multilingual contextual embeddings trained on Twitter data, so we decided on TwIN-BERT embeddings (Zhang et al., 2022). Our final system uses these for both English and Spanish data, replacing BERTweet. We also chaned our strategy from generating contextual BERT embeddings from the cleaned text to using the raw text as input, believing that this would improve the embeddings generated due to the fact that TwINBERT (and previously BERTweet) was trained on in-domain Twitter data that would have typos, punctuation, hashtags, and others features of our text not seen in the cleaned text.

4.2.7 English Slang Dictionary

This feature uses data from the SlangSD resource (Wu et al., 2018) to label slang words with their sentiment strength. The sentiment strength scale is from -2 to 2, where -2 is strongly negative, -1 is negative, 0 is neutral, 1 is positive, and 2 is strongly positive. The sentiment scores are summed across all the slang words in a tweet. The resulting accumulated sentiment scores are added to the original dataframes as sentiment features.

The slang dictionary provided by (Wu et al., 2018) contains a lot of function words and stop words that do not have meaningful semantic value for the calculation of the sentiment score for tweets. To address this problem, we introduced the stop words lexicon provided by (Manning et al., 2014) to our system's final version to remove words that do not contribute to the sentiment semantics of tweets.

4.2.8 Spanish Sentiment Score Module

In order to calculate the sentiment score of the Spanish tweets for our adaptation task, we first attempted to find a Spanish slang dictionary to help us find slang words in the Spanish tweets. However, much to our surprise, there are no abundant resources that provide the kind of slang dictionary we are looking for. The closest publicly available resource for the Spanish slang corpus is the data provided by (Castro-Sánchez et al., 2015), which consists of lists of interjections and phrases used in Mexican Spanish slang. However, after we tried to utilize the Mexican Slang lexicon provided by (Castro-Sánchez et al., 2015), the results were not ideal in that it could not successfully retrieve a meaningful amount of slang words from our given Spanish tweets. We suspect this is due to the fact that the Spanish tweets of our adaptation task may contain multiple dialects of Spanish, which makes the deployment of the Mexican Slang lexicon to our system less useful since the lexicon can only detect slang words in Mexican Spanish.

To remedy this problem, we switched gears and tried to find a general sentiment lexicon for Spanish. The Spanish Sentiment Lexicon by (Pérez-Rosas et al., 2012) provides a comprehensive lexicon that contains a list of Spanish sentiment-triggering words with their associated binary sentiment value (either 0 or 1). Because some entries in (Pérez-Rosas et al., 2012) have both positive and negative annotations, we assigned those entries as neutral in sentiment. Therefore, the resulting lexicon is a trinary sentiment lexicon, where -1 is negative, 0 is neutral, and 1 is positive.

The sentiment score calculation for the Spanish Sentiment Score Module is summing over every polarity word's sentiment score for a tweet. The accumulated Spanish sentiment scores are added to the original dataframe as sentiment features.

4.3 Classification

The ClassificationModel uses the engineered features to train a model to predict the target class. This is accomplished by first training the model on a training dataset that includes gold-standard labels for the classification task of interest. Once a model is trained the predict method can be used to make predictions with that model over a new dataset that contains all the features (except for the gold-standard targets) used during model training.

4.3.1 Baseline

The first classification method is a baseline classification that predicts that no tweet contains hate-speech, as this is the most frequently seen label in the training data (HS=0). This is a simple implementation intended to provide a base on which we can compare future classification methods that we implement.

4.3.2 Random Forest

The second classification method is a random forest method, using features as split-points to separate and classify the data. This is a multi-class classifier, and can classify data based on three binary classifications (no hate speech/hate speech, individual target/group target, no aggression/aggression) or as a five way classifier (with classes None, HS, HS+TR, HS+AG, HS+TR+AG) depending on the configuration of the model. In the final configuration of our system we framed the problem as a 5-way classification problem for all models. When training on the English data we found that the optimal Random Forest model had 500 trees, used the gini index as its splitting criterion, had a minimum split value of 10%, used $\sqrt{n}$ maximum features for each tree, balanced class weights and used a maximum of 70% of the total samples per tree. When training on the Spanish data we found that the optimal Random Forest model had 2000 trees, used the gini index as its splitting criterion, had a minimum split value of 10%, used $\sqrt{n}$ maximum features for each tree, did not use balanced class weights and used a maximum of 70% of the total samples per tree.

4.3.3 SVM

The third classification method is a support vector machine method. This is a multi-class classifier, and can classify data based on three binary classifications (no hate speech/hate speech, individual target/group target, no aggression/aggression) or as a five way classifier (with classes None, HS, HS+TR, HS+AG, HS+TR+AG) depending on the configuration of the model. As stated previously, we framed the problem as a 5-way classification problem for all models in our final system. When training on the English data we found that the optimal SVM used a linear kernel. When training on the Spanish data we found that the optimal SVM used a third-degree polynomial kernel.

4.3.4 Logistic Regression

The fourth classification method is a logistic regression model that classifies the data into the five categories described above. When training on the English data we found that the optimal Logistic Regression model was the model that was forced to stop training after 100 iterations. When training on the Spanish data we found that the optimal maximum number of iterations for the model was 500.

4.3.5 Ensemblers

We added two ensembling methods that attempt to predict the target class using the predictions made by the models described above. The first method does this using a logistic regression model. For the English data we found that the optimal logistic regression ensembler set the maximum number of training iterations to 2000. For the Spanish data that value was 1000. The second model does this using a decision tree. For the English data we found that the optimal used $lg(n)$ max features and the gini index as splitting criterion. For the Spanish data the optimal ensembler used $\sqrt{n}$ max featrues and the gini index.

4.3.6 RoBERTa

We also fine-tuned deep-neural transformer-based architectures for our English Tasks A and B. Specifically, we trained the RoBERTa for Sequence Classification model (Liu et al., 2019) which builds upon BERT, but modifies key hyperparameters, including removing BERT's next-sentence pretraining objective, and training with much larger mini-batches and learning rates.

We loaded the official pretrained RoBERTA-base model and finetuned it for a 5-way classification task using our English training data. The hyperparameters used are batch size 32, padding all tweets to 130 tokens (maximum size of tweets), employing Cross Entropy loss, learning rate 1e-5, and training for 100 epochs with early stopping criteria.

Admittedly, 100 epochs is insufficient, and the model would benefit from more epochs of finetuning. However we were unable to train on more epochs or train for the Spanish task due to resource and time constraints. Additionally, the training-test split used varies from the split used in the SVM and Random Forest training, leading to the neural method not being included in the ensemble methods.

4.4 Evaluation

The Evaluator takes the output of the classification and runs an analysis to evaluate the model's performance on the affect classification task. For each metric (HS, TR, AG), the evaluator calculates the precision, recall, accuracy, and F1 scores achieved by the model. This class is an adaptation of the Evaluation script provided by the SemEval 2019 Task 5 team to ensure the same evaluation techniques were used by all teams participating in the shared task. It was then modified to be a class within our model instead of a separate script.

Evaluation is broken into two separate tasks. Task A analyzes at the system's predictions for the binary Hate Speech classification, while Task B analyzes the system's predictions for the binary classifications of Hate Speech, Personal Target, and Aggression, as well as a Macro averaging of all the categories.

5 Results

5.1 Enhanced System Results

Table 7 lists the results of our enhanced system for both Task A and Task B. Observe that the best

Table 1: D3 System Evaluation Scores

	Performance Metrics			
	F1	Precision	Recall	Accuracy
Random Forest Scores				
Task A	0.545	0.558	0.550	0.578
Task B (Macro)	0.641	-	-	-
Task B (HS)	0.545	0.558	0.550	0.578
Task B (TR)	0.934	0.389	0.898	0.766
Task B (AG)	0.443	0.898	0.389	0.796
SVM Scores				
Task A	0.451	0.565	0.521	0.581
Task B (Macro)	0.614	-	-	-
Task B (HS)	0.451	0.565	0.521	0.581
Task B (TR)	0.447	0.641	0.398	0.781
Task B (AG)	0.943	0.398	0.641	0.795

Random Forest and the best SVM classifiers in the enhanced system outperform the baseline and the Initial system on the F1 metric. On average the enhanced systems perform worse on the precision metric than the initial system and baseline, but do much better on the recall and accuracy metrics.

Of the enhanced systems, the Random Forest outperforms the SVM for the Hatespeech Recognition (HS) and Targeted vs. General (TR) classification tasks, but its F1 value under-performs the SVM's when attempting to categorize tweets as aggressive or not (AG). The Random Forest also tends to struggle on the Recall metric more so than the SVM. This may be due to the fact that the classification problem is a single multi-class problem in the Random Forest but is three binary classification tasks in the SVM, which may cause the Random Forest classifier to produce more false-negatives than the SVM.

The final system configuration was selected after performing a series of experiments involving hyper-parameter tuning, feature ablation and data augmentation. The results of each of those experiments are presented below.

5.2 Hyperparameter Tuning Results

We began by performing hyper-parameter tuning on each of the implemented classification models, in order to select the best model architecture for the remaining experiments. We tested the SVM model with first through sixth degree polynomial kernels. The random forest classifier varied across the number of trees (500, 1000, or 2000), splitting criterion (gini or entropy), minimum split values (10%, 20%, or 50%), the maximum number of features seen by each tree ($\sqrt{n}$ or $lg(n)$), the maximum number of samples seen by each tree (20%, 50%, or 70%), and whether or not the class weights were balanced. The logistic regression classifier varied by the maximum number of iterations (100, 500, 1000, or 2000) and whether or not the class weights were balanced. The ensembler models were provided the results from the best configurations of each of the above models and then varied by the maximum number of iterations for logistic regression ensembler, or the splitting criteria (gini or entropy) and maximum number of features seen at each node ($\sqrt{n}$ or $lg(n)$) for the decision tree ensembler. We ran these experiments on the English and Spanish data separately. The results of the best performing configuration for each classifier are shown below.

Table 2: D4 System Evaluation Scores on English Validation Data

	Performance Metrics			
	F1	Precision	Recall	Accuracy
Random Forest Scores				
Task A	0.658	0.708	0.689	0.661
Task B (Macro)	0.658	-	-	-
Task B (HS)	0.658	0.708	0.689	0.661
Task B (TR)	0.722	0.707	0.623	0.786
Task B (AG)	0.593	0.623	0.707	0.633
SVM Scores				
Task A	0.698	0.700	0704.	0.700
Task B (Macro)	0.694	-	-	-
Task B (HS)	0.698	0.700	0.704	0.700
Task B (TR)	0.746	0.743	0.634	0.824
Task B (AG)	0.637	0.634	0.743	0.758
Logistic Regression Scores				
Task A	0.710	0.709	0.710	0.715
Task B (Macro)	0.710	-	-	-
Task B (HS)	0.710	0.709	0.710	0.715
Task B (TR)	0.759	0.760	0.659	0.836
Task B (AG)	0.661	0.659	0.760	0.776
Logistic Regression Ensembler Scores				
Task A	0.712	0.711	0.714	0.716
Task B (Macro)	0.709	-	-	-
Task B (HS)	0.712	0.711	0.714	0.716
Task B (TR)	0.756	0.754	0.654	0.832
Task B (AG)	0.658	0.654	0.754	0.772
Decision Tree Ensembler Scores				
Task A	0.699	0.699	0.704	0.701
Task B (Macro)	0.691	-	-	-
Task B (HS)	0.699	0.699	0.704	0.701
Task B (TR)	0.740	0.741	0.634	0.823
Task B (AG)	0.634	0.634	0.741	0.762
RoBERTa Scores (different train-test split)				
Task A	0.753	0.770	0.748	0.766
Task B (Macro)	0.740	-	-	-
Task B (HS)	0.753	0.770	0.748	0.766
Task B (TR)	0.785	0.801	0.772	0.860
Task B (AG)	0.683	0.704	0.670	0.811

Although RoBERTa performed best on the macro-F1 and HS-class-F1 scores, Random Forest scores higher on the TR-class-F1, and SVM performs better on the AG-class-F1. We do not provide further score analysis, since RoBERTa was trained on a slightly different train-test split.

Due to the difficulties we had training the RoBERTa model, we proceeded with the Logistic Regression classifier for the English data, which was the second-best overall classification model for this data set.

Table 3: D4 System Evaluation Scores on Spanish Validation Data

	Performance Metrics			
	F1	Precision	Recall	Accuracy
Random Forest Scores				
Task A	0.521	0.734	0.579	0.624
Task B (Macro)	0.587	-	-	-
Task B (HS)	0.521	0.734	0.579	0.624
Task B (TR)	0.656	0.828	0.773	0.794
Task B (AG)	0.583	0.773	0.828	0.712
SVM Scores				
Task A	0.754	0.786	0.751	0.768
Task B (Macro)	0.774	-	-	-
Task B (HS)	0.754	0.786	0.751	0.768
Task B (TR)	0.813	0.838	0.763	0.860
Task B (AG)	0.755	0.763	0.838	0.782
Logistic Regression Scores				
Task A	0.762	0.763	0.762	0.766
Task B (Macro)	0.770	-	-	-
Task B (HS)	0.762	0.763	0.762	0.766
Task B (TR)	0.795	0.794	0.752	0.836
Task B (AG)	0.753	0.752	0.794	0.774
Logistic Regression Ensembler Scores				
Task A	0.741	0.741	0.741	0.744
Task B (Macro)	0.754	-	-	-
Task B (HS)	0.741	0.741	0.741	0.744
Task B (TR)	0.789	0.786	0.731	0.830
Task B (AG)	0.733	0.731	0.786	0.754
Decision Tree Ensembler Scores				
Task A	0.735	0.735	0.734	0.738
Task B (Macro)	0.750	-	-	-
Task B (HS)	0.735	0.735	0.734	0.738
Task B (TR)	0.780	0.770	0.731	0.816
Task B (AG)	0.735	0.731	0.770	0.752

We proceeded with the SVM classification model for the Spanish data, which was the best overall performing model for that dataset.

5.3 Ablation Test Results

After determining the model architecture and parameters, we performed a series of ablation tests to determine if certain features were hurting model performance and should be removed. We proceeded by starting with the complete set of available features, temporarily removing each individual feature and checking the performance, permanently removing the single feature that caused the greatest improvement in model performance, and then iteratively repeating that process with the new, smaller, set of features, until we saw no improvement in the model's performance. For the English data we noticed the greatest improvement in the model if we removed the following features: fear, slangscore, percent_capitals, anger, anticipation_ext, anger_ext, negative_ext, disgust_ext, surprise_ext, negative, positive_ext, positive, joy, *_count_normalized, Universal_Sentence_Encoder_embeddings, and Aggregate_embeddings. For the Spanish data we noticed the greatest improvement in the model if we removed the percent_capitals feature.

Table 4: Ablation System Evaluation Scores on English Validation Data

	Performance Metrics			
	F1	Precision	Recall	Accuracy
Logistic Regression - Full Feature Set				
Task A	0.710	0.709	0.710	0.715
Task B (Macro)	0.710	-	-	-
Task B (HS)	0.710	0.709	0.710	0.715
Task B (TR)	0.759	0.760	0.659	0.836
Task B (AG)	0.661	0.659	0.760	0.776
Logistic Regression - Ablated Feature Set				
Task A	0.	0.	0.	0.
Task B (Macro)	0.736	-	-	-
Task B (HS)	0.747	0.775	0.742	0.764
Task B (TR)	0.770	0.813	0.701	0.859
Task B (AG)	0.693	0.701	0.813	0.808

Table 5: Ablation System Evaluation Scores on Spanish Validation Data

	Performance Metrics			
	F1	Precision	Recall	Accuracy
SVM - Full Feature Set				
Task A	0.754	0.786	0.751	0.768
Task B (Macro)	0.774	-	-	-
Task B (HS)	0.754	0.786	0.751	0.768
Task B (TR)	0.813	0.838	0.763	0.860
Task B (AG)	0.755	0.763	0.838	0.782
SVM - Ablated Feature Set				
Task A	0.756	0.788	0.757	0.770
Task B (Macro)	0.775	-	-	-
Task B (HS)	0.756	0.788	0.757	0.770
Task B (TR)	0.811	0.834	0.765	0.858
Task B (AG)	0.757	0.765	0.834	0.784

5.4 Data Augmentation Test Results

Early on in our system development we noticed skewed performance on certain, over-represented target classes. We attempted to boost the system's performance on infrequent classes by performing simple feature vector level data augmentation to our training examples. We did this through increasing the vector counts of each of the four minority classes to match the count of the highest category by adding random noise from a Gaussian distribution to a randomly sampled vector from each category until all classes were even.

The results of the data augmentation can be seen below. (Note that these results are for the model's performance on the full feature set.)

Table 6: Data Augmentation System Evaluation Scores on English Validation Data

	Performance Metrics			
	F1	Precision	Recall	Accuracy
Logistic Regression - Original Data				
Task A	0.710	0.709	0.710	0.715
Task B (Macro)	0.710	-	-	-
Task B (HS)	0.710	0.709	0.710	0.715
Task B (TR)	0.759	0.760	0.659	0.836
Task B (AG)	0.661	0.659	0.760	0.776
Logistic Regression - Augmented Data				
Task A	0.670	0.701	0.693	0.671
Task B (Macro)	0.681	-	-	-
Task B (HS)	0.670	0.701	0.693	0.671
Task B (TR)	0.749	0.732	0.783	0.808
Task B (AG)	0.625	0.618	0.656	0.713

Table 7: Data Augmentation System Evaluation Scores on Spanish Validation Data

	Performance Metrics			
	F1	Precision	Recall	Accuracy
SVM - Original Data				
Task A	0.754	0.786	0.751	0.768
Task B (Macro)	0.774	-	-	-
Task B (HS)	0.754	0.786	0.751	0.768
Task B (TR)	0.813	0.838	0.763	0.860
Task B (AG)	0.755	0.763	0.838	0.782
SVM - Augmented Data				
Task A	0.789	0.787	0.792	0.794
Task B (Macro)	0.799	-	-	-
Task B (HS)	0.789	0.787	0.792	0.794
Task B (TR)	0.813	0.802	0.826	0.860
Task B (AG)	0.762	0.755	0.774	0.781

6 Discussion

Our D3 model significantly over-predicted that a given tweet can be categorized as hate speech (HS=1), while also significantly under-predicting that a given tweet can be categorized as aggressive (AG=1). In our best performing D3 model, none of the tweets were classified as having aggressive language. We suspected this was due to the fact that only 14% of tweets in the data are labeled as 'AG', resulting in a significant class imbalance. Even with class-balanced weighting, the classifier still struggled to detect when a tweet should be categorized as aggressive, perhaps due to inadequate variation in the training data. For our D4 model, class imbalance was handled better resulting in increased F1, precision, recall, and accuracy across the board. Much of this improvement came from the hyperparameter tuning and overfitting prevention through our ablation testing.

In D3, the majority of our non-embedding features involved stylistic information about the tweet that may have been useful in detecting which tweets are more likely to involve hate speech. These included the percent of the tweet that is capitalized and normalized counts of special symbols $, !, ? and *. However the set of D3 features didn't appear to contain adequate information for the random forest classifier to utilize when performing hate speech classification. Additionally, in D3 we implemented lexical features for emotion from the NRC lexicon and the SlangSD resource. The normalized NRC counts were believed to contain semantic and pragmatic information that may be useful in detecting hate speech. Similarly, the slang dictionary was intended to help the model capture emotion-specific information that is less readily captured by other features and embeddings trained on standard English text. Unfortunately, both of these features were limited by the lexicon to which the tweet words were matched. However, for our D4 results, our improved feature engineering module added a large amount of additional non-embedding features related to emotional valence. The NRC Lexicon extension allowed us to circumvent the SlangSD resource, and sentence sentiment features helped to create enough variation in the feature space for the classifiers to make correct hate speech classifications. While these features were still more likely to be removed in ablation testing, they provided additional context that improved model predictions overall.

As mentioned above, one major concern we had from our D3 results was the possibility that our data cleaning process stripped our instances of useful information about hate speech, aggression, and the targeted nature of tweets. We sought to maintain this information via the features above, but we did not believe these features on their own would generate the best performance. For example, the addition of a spellchecker function to address the frequent misspellings in tweets was meant to help our model create better embedding representations, and it may have in the case of static GloVe embeddings. However, when generating contextual BERTweet embeddings, these spelling corrections and previously described data cleaning processes result in cleaned text that doesn't look much like tweets at all, reducing the usefulness of a model like BERTweet which is pretrained on Twitter data. For these reasons, for D4 we recreated our BERT embeddings (this time with TwIN-BERT) using our raw training text, rather than the cleaned data.

Taken together, the changes made across the model from improved data processing to more expansive feature engineering, data augmentation, and a greater array of classification systems all worked to greatly improve our model's performance from D3 to D4.

7 Ethical Considerations

As this project addresses hate speech toward women and immigrants, it is important to acknowledge that our team comes from a variety of backgrounds, and each of us exist at different intersections of identity. Language is constantly evolving, and so are people's uses and expressions of hate, prejudice, and aggression. While it is unlikely to significantly alter our model, many of the pretrained embeddings we are using may not be up to date with current perspectives on what constitutes hate speech. The databases we use to identify emotional information and the severity of slang terms are not current, and as a task from 2019, the data may also be outdated using today's understanding. As a team we have done our best to think critically about the impact of this project, and believe that our system should be viewed as a tool for this SemEval 2019 task, and not as a generally-applicable program for classifying hate speech outside of the shared task's domain.

8 Conclusion and Future Work

Though we have finished this project, there are still many ways that future work could build on our system. One major addition that would help model performance would be increased data augmentation. The dataset provided for this shared task was unbalanced, and this made it more difficult for our model to classify tweets as Aggressive. Through augmentation, we could create more data that assists the model in predicting classes correctly. Our Gaussian noise addition was naive, and could be improved in ways that could lead to meaningful improvements in performance. There is also more work that could be done to handle Spanish data. Since currently our calculation of the sentiment score for Spanish tweets is based on a word-level Spanish sentiment lexicon, it would be better if we could get access to resources that contain lists of multi-word Spanish slang terms in the future to better improve the accuracy of our calculated sentiment scores for Spanish tweets. In addition, many of the embedding models we used for Spanish feature engineering were multilingual embeddings that are still compromised of majority English data. Further exploration into Spanish-specific models would be interesting, to compare performance of multilingual and monolingual Spanish model performances. Additionally, with more time we would train the RoBERTa model for more than 100 epochs, (and on Spanish data with a model like XLM-RoBERTa), to improve its performance, and integrate it into our classification ensembler to improve overall system predictions.

A Appendix: Workload Distribution

- **Ben Cote**:
  - Implemented Spanish to English Translator
  - Implemented Spanish NRC Lexicon Counter
  - Implemented Spanish NRC Lexicon Extension
  - Assembled Class Presentation
  - Updated Overleaf document to reflect D4 model changes
- **Madhav Kashyap**:
  - Researched shared tasks
  - Researched and set up configuration for future deliverables
  - Set up GitHub repo
  - Setup HYAK infrastructure to finetune transformer models on GPU
  - Experimented with fine-tuning BERT-Tweet, BERT-base, and RoBERTa models for English Task B. Finalized on RoBERTa, and perfomed fine-tuning and hyper-parameter optimization.
- **Lindsay Skinner**:
  - Set up frameworks for the DataProcessor, FeatureEngineering and ClassificationModel classes
  - Implemented the data processing code
  - Implemented the feature-normalization method in FeatureEngineering
  - Implemented the module to perform the NRCLex extension for English.
  - Implemented Random Forest classifier, Logistic Regression classifier, and Ensemblers
  - Added the ability to treat the classification as separate binary tasks or a joint multi-class classification problem

  - Updated the configuration set-up from the initial code so that all models, parameters, tasks and features are specified in the config file
  - Ran the hyper-parameter tuning experiments
  - Ran the ablation tests
  - Ran the data agumentation experiments
  - Contributed to the Classification, Results, Discussion, and Conclusion and Future Work sections

- **Keaton Strawn**:
  - Researched shared tasks
  - Implemented GloVe, FastText, BERT, and Universal Sentence Encoder embeddings in FeatureEngineering class
  - Implemented spellchecking
  - Contributed to fine tuning BERT for Sequence Classification
  - Implemented Spanish embeddings, include GloVe, TwHIN-BERT, and sentence level embeddings
  - Implemented data augmentation
  - Contributed to the Discussion, Feature Engineering, Data Processing, and Future Work sections of write-up

- **Allan Tsai**
  - Researched shared tasks
  - Implemented the slang dictionary and sentiment score counter in the FeatureEngineering class
  - Researched resources for English and Spanish text sentiment classification
  - Implemented the Spanish sentiment score calculation module for the adaptation task
  - Contributed to the Introduction, Feature Engineering, Abstract, Results, and Future Work sections of write-up

B Appendix: Code Repository

Our team's repository can be found here on GitHub or directly via this URL: `https://github.com/madhavmk/affect_classification_ling_573`

References

Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Manuel Rangel Pardo, Paolo Rosso, and Manuela Sanguinetti. 2019. Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In *Proceedings of the 13th international workshop on semantic evaluation*, pages 54–63.

Cristian Cardellino. 2016. Spanish billion words corpus and embeddings. Technical report.

Noé Alejandro Castro-Sánchez, Yolanda Raquel Baca-Gómez, and Alicia González Martínez. 2015. Development of affective lexicon for spanish with mexican slang expressions. *Res. Comput. Sci.*, 100:9–18.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder for English. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 169–174, Brussels, Belgium. Association for Computational Linguistics.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *Proceedings of the Eleventh International AAAI Conference on Web and Social Media (ICWSM 2017)*, pages 512–515, Montreal.

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. *CoRR*, abs/1907.11692.

Sean MacAvaney, Hao-Ren Yao, Eugene Yang, Katina Russell, Nazli Goharian, and Ophir Frieder. 2019. Hate speech detection: Challenges and solutions. *PloS one*, 14(8):e0221152.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland. Association for Computational Linguistics.

Saif Mohammad. 2018. Obtaining reliable human ratings of valence, arousal, and dominance for 20,000 English words. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 174–184, Melbourne, Australia. Association for Computational Linguistics.

Saif Mohammad, Felipe Bravo-Marquez, Mohammad Salameh, and Svetlana Kiritchenko. 2018. SemEval-2018 task 1: Affect in tweets. In *Proceedings of the*

12th International Workshop on Semantic Evaluation, pages 1–17, New Orleans, Louisiana. Association for Computational Linguistics.

Dat Quoc Nguyen, Thanh Vu, and Anh Tuan Nguyen. 2020. BERTweet: A pre-trained language model for English tweets. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 9–14, Online. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Verónica Pérez-Rosas, Carmen Banea, and Rada Mihalcea. 2012. Learning sentiment lexicons in Spanish. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 3077–3081, Istanbul, Turkey. European Language Resources Association (ELRA).

Alison Ribeiro and Nádia Silva. 2019. Inf-hateval at semeval-2019 task 5: Convolutional neural networks for hate speech detection against women and immigrants on twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 420–425.

Zeerak Waseem. 2016. Are you a racist or am I seeing things? annotator influence on hate speech detection on Twitter. In *Proceedings of the First Workshop on NLP and Computational Social Science*, pages 138–142, Austin, Texas. Association for Computational Linguistics.

Zeerak Waseem, Thomas Davidson, Dana Warmsley, and Ingmar Weber. 2017. Understanding abuse: A typology of abusive language detection subtasks. In *Proceedings of the First Workshop on Abusive Language Online*, pages 78–84, Vancouver, BC, Canada. Association for Computational Linguistics.

Liang Wu, Fred Morstatter, and Huan Liu. 2018. Slangsd: building, expanding and using a sentiment dictionary of slang words for short-text sentiment classification. *Language Resources and Evaluation*, 52(3):839–852.

Xinyang Zhang, Yury Malkov, Omar Florez, Serim Park, Brian McWilliams, Jiawei Han, and Ahmed El-Kishky. 2022. Twhin-bert: A socially-enriched pre-trained language model for multilingual tweet representations at twitter. *arXiv preprint arXiv:2209.07562*.