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 a random forest classification method and an SVM classifier. We present the results obtained for both Subtask A and Subtask B for English tweets. Our initial system achieved a Macro F1-score of 0.415 for English tweets. Our enhanced system achieved a Macro F1-score of 0.614 for English tweets.

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 initial system. Section 6 contains the discussion and summary. We conclude and describe future work 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

[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

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 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 will adjust our system in order to classify tweets in both English and Spanish on the three binary metrics listed above. In both cases, evaluation will be done by comparing our system's classification of each tweet with the pre-labeled classifications. Precision and recall of the classification will be amalgamated into a macro-averaged F1-score. Additionally, we will 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

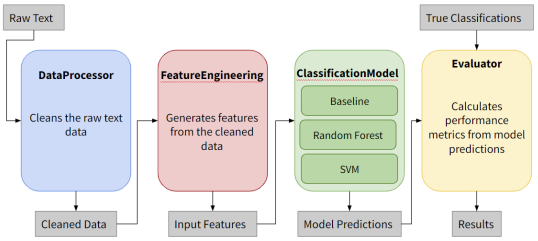

Figure 1: A graphical depiction of our Hate Speech Classification system

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.

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

This feature uses the resource from (Mohammad, 2018). This feature involves the binary classification of words across eight emotional dimensions (anger, anticipation, disgust, fear, joy, sadness, surprise, trust), then a positive dimension and a negative dimension. Raw counts are normalized across tweets.

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 (in the case of D2 they are d=25). 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.

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 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.

4.2.5 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.

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. We have three classification methods implemented in the current system.

4.3.1 Baseline

The first classification method is a baseline classification that predicts that no tweet contains hatespeech, 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. We found that the Random Forest performed best when treating the classification problem as five-class classification. After hyper-parameter tuning we determined the optimal Random Forest model had 500 trees, used entropy as the splitting criterion, required a minimum of 20% of observations to create a new branch, allowed each tree to use a maximum of $\sqrt{n}$ features (where n is the total number of features), used class-balanced weights, and allowed each tree to observe a maximum of 50% of the training observations when training.

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. We found that the SVM performed best when treating the classification problem as three separate binary classifications. After hyper-parameter tuning we determined the optimal SVM model used a five-degree polynomial kernel.

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 Initial System Results

Table 1: D2 System Evaluation Scores

	Performance Metrics			
	F1	Precision	Recall	Accuracy
Baseline Scores				
Task A	0.364	0.787	0.500	0.573
Task B (Macro)	0.415	-	-	-
Task B (HS)	0.364	0.787	0.500	0.573
Task B (TR)	0.439	0.891	0.500	0.781
Task B (AG)	0.443	0.898	0.500	0.796
Random Forest Scores				
Task A	0.299	0.714	0.500	0.427
Task B (Macro)	0.370	-	-	-
Task B (HS)	0.299	0.714	0.500	0.427
Task B (TR)	0.640	0.668	0.745	0.669
Task B (AG)	0.169	0.602	0.500	0.204

Table 1 lists the results of our initial system for both Task A and Task B. As illustrated, our initial system achieved an F1-score of 0.299 in identifying hate speech in English. For the personal target classification, we achieved an F1-score of 0.64. As for the aggressive behavior detection, our initial system has an F1-score of 0.169. The initial system under performed the baseline in all metrics, except for Recall on the Target-vs-Group classification subtask of Task B.

5.2 Enhanced System Results

Table 2: 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

Table 2 lists the results of our enhanced system for both Task A and Task B. Observe that the best 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.

6 Discussion

The model significantly over-predicts 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 model, none of the tweets were classified as having aggressive language. We suspect this is 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 struggles to detect when a tweet should be categorized as aggressive, perhaps due to inadequate variation in the training data. For this reason, we suspect the final iteration of this project will benefit from an exploration of data augmentation approaches in order to prevent the model from biasing towards a particular class label. In addition to this issue with the classifier, we have identified some weaknesses in our current feature set that we hope to improve upon in future iterations.

The majority of our non-embedding features involve stylistic information about the tweet that may be useful in detecting which tweets are more likely to involve hate speech. These include the percent of the tweet that is capitalized and normalized counts of special symbols $, !, ? and *. When combined with other features that contain more semantic information, we expect these stylistic features may provide an edge in detecting hate speech.

However, there are certain aspects of the data cleaning process that are not yet being used for feature generation. Specifically, we separate hashtags from the tweet but have yet to incorporate any information from the hashtag into the set of features passed to the classification model. For certain tweets this practice can remove all of the relevant signal needed to perform the classification task.[3] In the future we intend to encode this information using either hashtag-specific embeddings, or a more generalized embedding that encodes the raw text (which is discussed at the end of this section).

In addition to the aforementioned features, we included lexical features for emotion from the NRC lexicon and the SlangSD resource. The normalized NRC counts are 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 are limited by the lexicon to which the tweet words have been matched. In the future we hope to use global embeddings to extend these features so that we can generate emotion and valence scores for all words in a tweet, not just those that match the provided lexicon.

As mentioned above, one major concern we have is the possibility that our data cleaning process is stripping our instances of useful information about hate speech, aggression, and the targeted nature of tweets. While we have sought to maintain this information via the features above, we do not believe these features on their own will 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 we intend to recreate our BERT embeddings using our raw training text, rather than the cleaned data.

To solve this problem more generally, we are working on fine-tuning BERT for Sequence Classification with our raw data in order to implement a neural method that can be combined with our existing system. Our intention is to use an ensembling method that will aggregate feedback from

[3] For example, the tweet "Hurray, saving us $$$ in so many ways @potus @realDonaldTrump #LockThemUp #BuildTheWall #EndDACA #BoycottNFL #BoycottNike" is categorized as hate speech. However, after the data cleaning process the cleaned text for this tweet is "hurray saving us in so many ways user user", which does not contain any information that obviously marks it as hate speech.

each of our trained models in order to capture different dimensions of the data when making the final classification decision.

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

In future iterations of this project we plan to improve our feature set in several ways. We plan to add embedding features that represent the hashtag and emoji information, which is currently removed from the text during the data cleaning process. We also plan to improve our slang dictionary feature through stop word removal. We will also improve the NRC word counts feature by training a linear classifier to predict the emotion and valence category values using word embeddings, so that we can extend our counts to include all words in the text, and not just those that are found in the lexicon. Finally, we plan to explore a variety of different classifiers and different configurations through ablation testing, and explore ensembling with the addition of BERT for Sequence Classification as a neural method for the task.

A Appendix: Workload Distribution

- **Ben Cote**:
  - Implemented SVM classifier
  - Implemented BERTweet embeddings from Keaton's initial code
  - Implemented Universal Sentence Encoder Embeddings from Keaton's initial code
  - Compiled System Overview, Approach, and Results sections
- **Madhav Kashyap**:
  - Researched shared tasks
  - Researched and set up configuration for future deliverables
  - Set up GitHub repo
- **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 Random Forest classifier
  - Added the ability to treat the classification as separate binary tasks or a joint multi-class classification problem
  - Updated the configuration set-up from Ben's initial code so that all models, parameters, tasks and features are specified in the config file
  - Ran the hyper-parameter tuning experiments
  - Contributed to the Results, Discussion, and Conclusion and Future Work sections
  - Implemented the NRC-Lex extension module (not yet integrated into Feature Engineering)
- **Keaton Strawn**:
  - Researched shared tasks
  - Implemented GloVe, FastText, BERT, and Universal Sentence Encoder embeddings in FeatureEngineering class
  - Implemented spellchecking
  - Began work on fine tuning BERT for Sequence Classification
  - 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 text sentiment calculation
- Contributed to the Introduction, Feature Engineering, Abstract, and Results sections of write-up

B Appendix: Code Repository & Additional Resources

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

Additional Resources:

- nltk

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.

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.

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.

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.

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.