First Year Project 3

Natural Language Processing

Alexander Nielsen
alwn@itu.dk

Christian Hetling
chrhe@itu.dk

Erling Amundsen
erla@itu.dk

Krzysztof Parocki
krpa@itu.dk

Malthe Musaeus
mhmu@itu.dk

IT UNIVERSITY OF CPH

June 2, 2022

Contents

1 Introduction **1**

2 Data and Preprocessing **1**

3 Annotation **3**

4 Classification **3**

5 Limitations **5**

6 Conclusion and Future Work **5**

List of Figures

1 zipf's law log Rank - log Frequency plot 2
2 Perplexity score histogram for the hate language model run on the full offensive dataset. . . . 4

List of Tables

1 Distribution of labels in the hate and sentiment datasets. 1
2 Precision, Recall and F1-scores for the different models tested on trained on the hate dataset. 4
3 Precision, Recall and F1-scores for the best fitting model found through grid search for the hate dataset. 4

1 Introduction

Social media is a central part of today's society. It's used by billions of people worldwide to both send and receive messages, pictures and videos. Twitter is but one of the major social medias. The platform is built around tweets, which are 280-character posts that can be liked, commented and shared. This functionality enables us to share our every thought on a whim. Twitter is especially known as a platform for political discussion, where both politicians and civilians can share their stance on political debates. Even though this is great in terms of free-speech, this also creates a breeding ground for hate-speech and misinformation.

Manually detecting hateful comments online is infeasible and monotonous. It's infeasible because of the large amount of comments continuously being produced online, and monotonous because each classification of a comment has only two possible outcomes: hate or not hate. Natural language processing (NLP) models are therefore essential in the censorship of social media platforms. In this paper we present an automated solution for a computer classifier that classifies a comment as being either hate or not hate, trained on pre-labeled tweets. Applications of such a classifier could be to govern online communities and help flag authors who write many hateful comments.

Furthermore, we introduce a multi class sentiment classification of tweets as having either positive, neutral or negative sentiment. The sentiment classification is also trained on pre-labeled tweets, but as opposed to the hate dataset, the labels of the sentiment classifier is multi class.

The ethical considerations surrounding the infringement on free-speech will not be considered while constructing the models. The goal of this paper is to construct a model that can automatically classify if a tweet is hateful and the sentiment of tweets.

A tweet is limited to only 280 characters, forcing the writer to be concise and precise in their writing. This, combined with the popularity of the platform has made it one of the most develop parts in natural language modeling, and one of the biggest sources for development of natural language models.

2 Data and Preprocessing

We used the TweetEval corpus[1], a collection of 7 datasets for different classification tasks based on social media posts. The datasets came with predefined splits into training, test and validation sets.

We used two datasets to classify hate speech and sentiment. In order to binary classify a comment as being either hateful or not, we used the dataset "Hate Speech Detection.". And for the multi class classification we used the dataset "Sentiment Analysis", to classify a text as being either positive, neutral or negative.

	Hate Classification				Sentiment Classification					
	hate		non-hate		negative		neutral		positive	
dataset	count	%	count	%	count	%	count	%	count	%
train	1252	42.2%	1718	57.8%	3972	32.3%	5937	48.3%	2375	19.33 %
test	3783	42.0%	5217	58.0%	7093	15.5%	20673	45.32%	17849	39.1%
validation	427	42.7%	573	58.3%	312	15.6%	869	43.5%	819	40.9 %

Table 1: Distribution of labels in the hate and sentiment datasets.

As can be seen from Table 1, the hate dataset is very close to being evenly balanced between the two labels, with the hate count being 42% ± 1% and the non hate then being 58% ± 1%. Conversely, the multi class sentiment classification has a distribution that is skewed. The train dataset cointained a higher rate of negative tweets and neutral tweets, and a lower rate of positive tweets, compared to the test and validation sets.

Before training any models on the corpora we characterized the data by collecting different numerical statistics. The characterization was done after tokenization but before the preprocessing was applied - we didn't want to remove some of the most common tokens (stopwords) before finding the size of the vocabulary and the overall corpus. The vocabulary size of Hate was 18561, with a full corpus size of 215580, yielding a type/token ratio of 0.0862, indicating that a lot of words repeated. The sentiment dataset has a larger vocabulary of size 47551 with a corresponding corpus size of 1069745 resulting in a type/token ration of 0.0444.

In the context of tweets and the data size, it makes sense that both sets have a lot of repeated words - it is in line with Zipf's law. Additionally, it can be seen that the sentiment data set has a lower score, indicating that words are more likely to be reused.

The most common words in the corpora were "@user" and "the" for hate and sentiment respectively. "@user" being the most common token in the hate data makes sense, since a lot of the tweets containing hate often are directed at someone specific. Words like "the" are very common but without actually communicating any additional information about.

When looking at the least common words in both, that only appear once, twice or thrice, it was evident that they were dominated by nouns and proper nouns. It makes a lot of sense that these are less frequent, since these are usually more unique and relate to a specific topic. It makes sense as the corpus increases that type/token relationship decreases, and it highlights why a large corpus is good. With words occurring again more often, language models will presumably become more accurate.

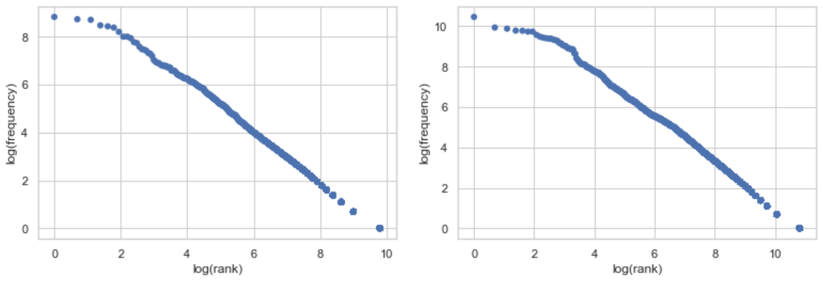

Figure 1: zipf's law log Rank - log Frequency plot

In both corpora we find strong evidence for Zipf's law. Zipf's law dictates that the rank of tokens is inversely proportional to the frequency of occurrence. To visualize this we plotted the log frequency against the log rank as seen in Figure 1. From the figure it can be seen that the log frequency and the log rank seem to be linearly correlated, since it almost resembles a straight line.

For tokenizing we used several different methods. We made our own tokenizer that struggled with certain tokens, like '...' and dashes that extended words. Initially, we tried using our custom tokenizer, arguing that the specific nature of tweets i.e. short and containing a lot of slang, would be lost if we used a general one. We debated a long time if emoji's made enough of a difference to be included as tokens. In the end we observed that a large majority of emoji's only showed up a couple of times, and that because of this not a lot information could be inferred from them. Later, we used NLTK's word tokenizer and tweet-tokenizer, both accounting for more edge cases, and the latter more accurately tokenizing hashtags and '@user'. However, they all had some cases that were tokenized differently. For the final model, we used NLTK's tweet-tokenizer, as it seemed more suitable for the edge cases one might find in a tweet.

After tokenizing the corpora we implemented some preprocessing to further refine the data before passing it on and training our models. The preprocessing consisted of lower-casing tokens, so tokens like "family" and "Family" would be considered equal. Next, we removed punctuation and stopwords, to remove clutter and tokens that didn't necessarily convey that much information about the sentiment or hate in a tweet. Stopwords (from NLTK) are a list of some of the most common English words, which often mean they do not convey additional information about the given text, which is especially true when using a bag-of-words approach. Lastly, all the tokens were stemmed. The process of stemming is to reduce words to their root/stem, to avoid different conjugations of words being evaluated as unique tokens.

We trained a language model (LM) based on the hate dataset. We had to smooth the data to account for words that had not been seen before during the training of the model. The smoothing we implemented was the Witten-Bell method from the NLTK library, which treats the probability of getting a new word based on the amount of possible words that finish the N-gram. The reason we chose the Witten-Bell over the Kneser-Ney method was mainly due to the NLTK implementation of Kneser-Ney only working on a closed vocabulary, and as such did not meet our requirements for a Language Model - we couldn't implement it.

For the training of the classifiers, we set up a pipeline through Scikit-learn. The pipeline is a series of functions that all tweets are passed through during the training process. It starts by vectorizing the tweet into a count matrix, during which the preprocessing and tokenization is also applied. The final step in the pipeline uses our own preprecessor and NLTK's *TweetTokenizer* function. The tokenized tweet is then passed to the tf-idf transformer, which can weigh the importance of words based on the their frequency times the inverse of their document frequency. In this case, that would be their frequency in the given tweet times the amount of times the token is seen in total. Finally, the tweet is passed to the classification model for training or prediction of the label.

3 Annotation

The datasets we worked with already had pre-labeled annotations for each sample text. For the binary hate dataset we did a quality check to see if we all agreed on the same annotation scheme. After computing the inter-annotator agreement of all five group members, along with the original labels from the dataset, we got a Fleiss' κ score of 0.29. Fleiss' κ is a generalization of Scott's π for working with multiple annotators. When annotating data it is preferable to have multiple annotators the same sample of training data to verify the labels. We used the multi-version of Scott's π over Cohen's κ because we were interested in overall agreement, and not in the reliability of annotations everyone agreed on. Ideally, the score should be nearing 1, which would imply that all annotators agree on the annotations of each tweet. But since we only used the original annotations from the dataset, this was not a big concern for the accuracy of the models. However, if we wanted to use our annotations, we should refine the guidelines and repeat the process.

4 Classification

We explored several different classification models during our experimentation. For the different models, the exact same pipeline was used to minimize the influence that the differing prepossessing steps could have. We chose to compared naive bayes, linear SVM and logistic regression on the hate data set.

These initial results were made using a pipeline with standard parameters. From this result we chose to further look into logistic regression and the linear SVM model. Hyperparameter tuning was done using SK-Learn's GridSearchCV function. This function exhaustively searches over specified parameter values for a given model and pipeline. The parameters of the model were found by cross-validated grid-search over our specified parameter grid.

Model Evaluation			
Model	Precision	Recall	F1-Score
Logistic regression	0.64	0.52	0.51
Linear SVM	0.64	0.51	0.45
Naive Bayes	0.55	0.54	0.47

Table 2: Precision, Recall and F1-scores for the different models tested on trained on the hate dataset.

To obtain more training data, we used data augmentation techniques. Our approach was to compute the perplexities of each text sample in the "offensive" dataset. The perplexity scores were then plotted in a histogram (Figure 2) to see the spread of perplexities. For tweets with low perplexity score, we assumed that those tweets have strong similarity to the hate dataset our language model was trained on. Therefore, we computed the 1st quantile of the perplexities, and the data obtained from the 1st quantile was run through the hate classifier to get hate-labels for the newly added tweets. We then added those tweets to the combined training set with tweets from both the hate and newly-annotated offensive datasets. That meant that we now had extra text data, corresponding to 25% of the offensive dataset. The combined training set with extra data along with their computed label was then used to train a new hate classifier. This resulted in an increase in accuracy from 70% before data augmentation to 71% accuracy afterwards, with the precision and recall metrics both up. In our case, recall is measured as properly labelled non-hate tweets, and precision as how many of non-hate labels were correct. Therefore, it would make sense for recall to increase and precision to fall - by using the augmented dataset, we mostly provided our model with more examples of non-hate tweets. This was indeed the case for other hyperparameters, however, for this particular one, the accuracy increased so much that both precision and recall metrics went up.

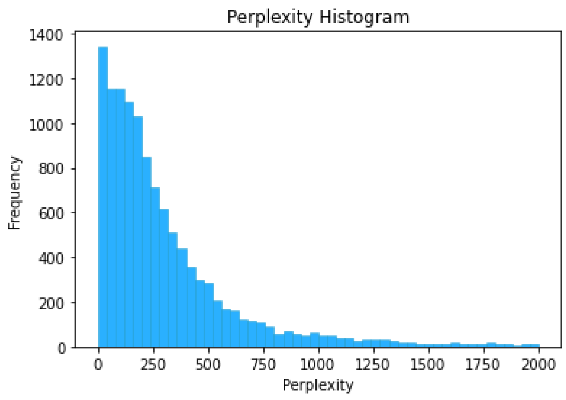

Figure 2: Perplexity score histogram for the hate language model run on the full offensive dataset.

Evaluation of best fitting model with data augmentation (hate)				
Category	Precision	Recall	F1-Score	Tested on
Not hate	0.74	0.78	0.76	573
Hate	0.68	0.62	0.65	427
Accuracy			0.71	1000
Macro avg	0.71	0.70	0.71	1000
Weighted avg	0.71	0.71	0.71	1000

Table 3: Precision, Recall and F1-scores for the best fitting model found through grid search for the hate dataset.

Although a 71% combined F1-score is arguably not high enough on its own to create a meaningful filter on platforms such as Twitter, this algorithm could be used to alleviate the workload of moderators by automatically flagging accounts with a higher percentage of hateful tweets. After fitting the model, we manually fed sentences to it and discovered several words or phrases that seemed to get miss-classified. One such phrase was "You dumb fucking immigrant", which ended up being classified as not hate.

For the sentiment classification task, we used the same pipeline, using grid search again to optimise the parameters, but didn't augment the data. This resulted in a classifier with an accuracy of 0.57 and an F1-Score of 0.57.

5 Limitations

As our training dataset was derived from Twitter, all the models were trained on only tweets. This could imply that the models will struggle adapting to other types of texts such as product reviews and other internet forum comments. With regards to the character limit on Twitter, our model is specialised on short expressions of opinions, and would likely struggle with classifying longer format texts. Compared to the data that modern NLP models are trained on, our dataset is relatively small, being orders of magnitude smaller that recent models trained on similar data[3]. Also, our model evaluates all tweets as a "bag of words" meaning all positional data is lost. The amount of information possible to infer from position decreases with the size of the data, therefore, it is hard to teach the model to understand the position of words without a more substantial data set.

One limitation that did not affect our work model, but deserves a mention as a limitation for real world applications, is the poor inter-annotator agreement. It serves neither consumers nor the platform to spread hate messages or derogatory opinions. It is clear that a boundary is needed for censorship. However, as can be seen from our Inter Annotated Agreement, defining that border is difficult. Over-censoring can also be dangerous, and companies are constantly adjusting that border to find the balance.

The manual annotations done in the group got a Fleiss' κ score of 0.29 for the inter-annotator agreement, which is a fair score [2]. The main point of disagreement in the Inter Annotated Agreement seemed to be centered around a discussion of what should be censurable. While we all agreed that a line should be drawn, and we often agreed when discussing tweets in hindsight if they where offensive or hateful, we would disagree if it it was enough to be censored. Often these tweets were based on political opinions, and there is a larger debate going on if social media platforms should censor such tweets.

6 Conclusion and Future Work

Natural language classification is an inherently difficult task. We observe through Zipf's law that a large amount of tokens only appear once or twice. This limits the information we can gain from simply looking at word occurrences. However, we've learned new tools that helped us cope with this difficulty. Quite often, it turned out that what was already implemented in the field was better than our own implementation - but it's better to learn that first-hand. After doing hyper-parameter optimization on the logistic regression we found that many of the optimal parameters were the default parameters already specified by Scikit-learn.

Given that we embedded the trained machine learning models into complete pipelines, including pre-processing and vectorization steps, the model can be used directly to assist classifying text on social media. That means that a sentence can be passed directly to a fitted pipeline and a classification will come out, forming an end-to-end pipeline from text to prediction.

For the hate classification task, one might argue that an accuracy score of 71% on a binary classification task is below human standards. But given the large amount of text produced on social media, we think that our proposed model could provide a good baseline for detecting and flagging hate speech when human availability is limited.

A natural next step to further improve this model could be to gather data from other sources than just Twitter. That would arguably improve the overall robustness of the model towards new text. Perhaps a greater and more broadened Tweet dataset could be used to make the hate classifier perform better for general sentences, and thus minimize the amount of false negatives.

References

[1] Francesco Barbieri, Jose Camacho-Collados, Luis Espinosa-Anke, and Leonardo Neves. TweetEval:Unified Benchmark and Comparative Evaluation for Tweet Classification. In *Proceedings of Findings of EMNLP*, 2020.

[2] J Richard Landis and Gary G Koch. The measurement of observer agreement for categorical data. *biometrics*, pages 159–174, 1977.

[3] Ram Sagar. When do language models need billion words in their datasets, Oct 2021.