IT-UNIVERSITETET I KBH

Natural Language Processing

4th Project Report - Group 13

Josephine Holst
jtih@itu.dk

João Vieira da Silva
joap@itu.dk

Ludek Cizinsky
luci@itu.dk

Disclosure

The group was reduced from 4 to 3 people due to one member dropping out (mnib@itu.dk). Besides this, all group members contributed equally to the project. Due to the real-time collaborative platform we used, the attached git log will not reflect this correctly.

Introduction

With the advancement of technology, vast amounts of data are being generated each day. The majority of this data is produced in an unstructured textual form as people communicate through written text, from social media posts to electronic health records. The complexity and ambiguity of natural language makes information retrieval from textual data a challenging task. Therefore, fields such as Natural Language Processing (NLP) are in high demand and rapidly evolving to enable a high-quality, consistent and unbiased evaluation of the large amount of natural language data. In NLP, linguistic analyses are performed in order to enable computers to "read" and categorise texts by decoding the ambiguities in human language. Yet modern NLP systems often prove insufficient when applied to noisy user-generated texts such as social media data, where language is high-paced and conversational and contextual cues can be limited[1]. To better understand and possibly expand on the evaluation of social media texts, this report is based on two common NLP classification problems, namely a binary and a multi-class classification task performed on Twitter data. These tasks deal with hate speech detection in tweets and sentiment analysis respectively. Thereby, the following pages present and discuss the results of these highly exploratory tasks simultaneously. First, data collection and preprocessing procedures are presented, including tokenization strategies and corpora characterization. Next, the results of annotation quality control are presented in the form of inter-annotator agreement levels. Then the results of multiple classification experiments are reported and discussed together with proper evaluation metrics such F-scores and macro averaged recall. Finally, main conclusions are summarized along with suggestions for future improvements and work.

Data and Preprocessing

Tasks and Provided Data

As mentioned, this study comprise of two classification tasks. The first task tackles a binary classification problem in the form of hate speech detection. It aims at predicting whether a tweet is hateful or not against two target populations: immigrants and women. The second task deals with a multi-class classification problem in the form of sentiment analysis. This task aims to recognize if the opinions, emotions or attitudes expressed in a tweet are positive, negative or neutral. Data for both tasks were obtained from the TWEETEVAL benchmark for tweet classification in English[1]. Focusing on seven tweet classification tasks - including hate speech detection and sentiment analysis - this benchmark provided manually labeled samples of anonymized tweets for the two tasks at hand, with predefined splits into training, validation and test sets (see distributions in Table 1). To reduce the risk of overfitting, training and validation data were merged into one data set ('training') during tokenization to enable cross validation during model training.

[1] https://arxiv.org/pdf/2010.12421.pdf

Task	Labels	Train	Val	Test
Hate speech detection	2	9,000	1,000	2,970
Sentiment analysis	3	45,389	2,000	11,906

Table 1: Number of labels and instances in training, validation, and test sets for each dataset.

Tokenizer Development

During preprocessing all raw data sets were tokenized into separate words. For this task, a tokenizer was manually implemented to adequately deal with the unique characteristics of the data at hand. The pipeline was developed using the first 200 tweets from the training data set for both tasks, which were then excluded from training data in order to ensure unbiased evaluations. The selected tweets were manually inspected to identify contextual challenges such as dealing with emojis, user handles, misspellings and hashtags. This was done by running several iterations to evaluate and make changes. The final tokenizer was implemented using the Python regular expressions library (RegEx). In this process, all matching consecutive characters were added as separate words to a complete list of tokens. This included adding letters mixed with special characters (e.g. contractions, hyphenated words, shortcuts (as '/w' for 'with') and hashtags) as well as removing URLs, anonymized handles ("@user") and numbers, as these sequences were considered of minimal importance to the two tasks. Punctuation, question marks, commas and exclamation marks were saved as individual tokens, while multiple punctuation and dots were saved as one. This was done to preserve the potential emphasis of a given sentiment. Before tokenization the text data was processed using external libraries in order to escape HTML tags, decode unicode, convert emojis and emoticons into predefined words, expand contractions and lowercase all letters. Finally, all stop words were removed from the list of tokens to further reduce vocabulary size.

Tokenizer Evaluation

The final output was compared to the baseline tokenization from the social media tokenizer in the NLTK library (TweetTokenizer) for both tasks. A quantitative comparison yielded a sequence similarity ratio of 0.73 for both data sets. A qualitative inspection revealed that main differences revolved around stop word, number and special character removal as well as lowercasing and emoji/emoticon replacement, all of which were not applied in the TweetTokenizer. While our tokenizer proved more selective, tokenizing fewer words in general, the TweetTokenizer revealed both unicode, and kept anonymized user tags and special characters such as quotation marks; tokens that were otherwise considered unimportant for the specific tasks.

Data Characterization

After tokenization, elementary corpus statistics were evaluated on training data for both tasks to inspect vocabulary features. Corpus and vocabulary size varied according to the number of instances in each data set, yielding significantly larger values for sentiment analysis. The type/token ratio produced distinct values as well, with a ratio of 0.148 for hate speech and 0.081 for sentiment analysis. As both values were low, this signify lean vocabularies where few specific tokens make up the bulk of the texts and a large part consists of rare tokens. This was further exemplified with scatterplots and log-log plots of token frequency vs. rank for both corpora. These showed that the first 2000 tokens make up around 75% of the corpora for both data sets. Similarly, the somewhat straight lines in the log-log plots confirms that Zipf's law holds for both data sets - at least as an approximation (see Figure 1). Thus, the processed data clearly demonstrate the pervasive sparseness of natural language data. For both data sets the most frequent tokens were characters such as punctuation, question marks and exclamation marks, indicating that these might often be used as expressive features. The most frequent words for both data sets deviated highly. Words such as "bitch", "women" and "refugee" were common in hate speech texts while "tomorrow", "may" and "going" were frequent in sentiment analysis. Examining the least common words in both data sets, rare words like "compounded" and "breeders" were largely accompanied by misspellings, hashtags and slang. This further exemplifies how tweets' idiosyncratic characteristics might contribute to a rapid growth of corpora.

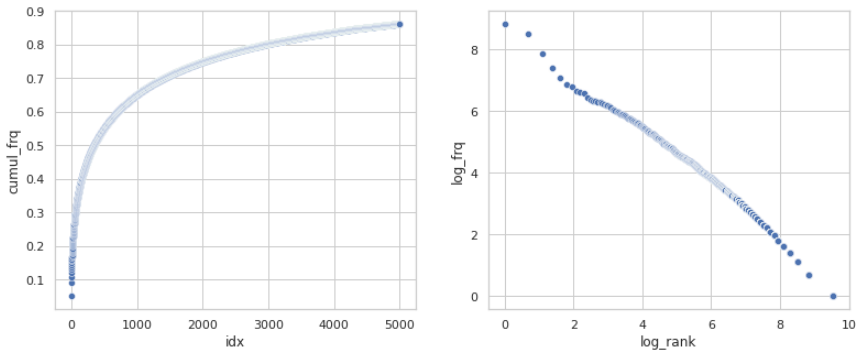

Figure 1: Left: Cumulative freq. vs. rank for the first 5000 tokens (Hate Speech). Right: log(freq.) vs. log(rank) (Hate Speech).

Annotation

As part of the project, a partial annotator training procedure was done to examine the trustworthiness of the annotations as well as highlight the challenges of labeling natural language. Manual labeling was performed on a random sample of 120 tweets from the hate speech dataset. Each group member annotated the sample according to the same guidelines as the provided labels[2]. These guidelines included examples of non-hateful and hateful tweets as well as concept definitions for both target groups (immigrants and women), e.g. hate speech and misogyny. This resulted in a total of four annotators (including the original labels), from which two chance-corrected metrics were computed to measure the level of inter-annotator agreement (IAA): Cohen's Kappa and Fleiss's Multi-π. Cohen's Kappa enabled a pair-wise comparison of IAA levels. A total score was computed by a naively averaging over kappas for each coder pair. Fleiss's Multi-π provided a more robust average score as this metric generalizes Scott's π to include more than two annotators. Results from both methods showed a consistent average around 0.52, which indicates a moderate agreement level for the random sample. Pair-wise levels varied from 0.36 (fair agreement) to 0.70 (substantial agreement). This revealed that all group annotators shared a moderate agreement (0.49 - 0.53) with the original labels while both

[2]https://github.com/msang/hateval/blob/master/annotation_guidelines.md

the lowest and highest agreement appeared between group members. The original labels were derived using similar procedures with a reported average confidence level of 0.83 based on at least three independent judgments for each tweet[3]. The results demonstrate the complexity of constructing good guidelines for manual annotation tasks and producing consistent annotations. Hate speech showed to be challenging to identify as edge cases were difficult to interpret. Disagreement often occurred in cases where guidelines were interpreted differently or concepts were hard to distinguish, when tweets quoted other people's hateful speech, when hate speech occurred only in hashtags or when there was a general lack of context (e.g. reactions and references to other tweets or contexts). These ambiguities allowed for personal opinions to influence the results. Thus, the moderate agreement level indicates that guidelines could appropriately be re-formulated, further specified or more thoroughly consulted for future iterations. Since the original labelling was facilitated by a well-established crowdsourcing platform[4] the results from our own procedure was not thought to jeopardize the confidence level of the provided labels.

Classification

Feature extraction

Four different methods were employed for feature extraction: CountVectorizer, TfidfVectorizer, Word2Vec and LSA. The first two are based on a Bag-of-Words model where the order of tokens is ignored. As the name suggests, CountVectorizer represents tweets by the frequency of tokens occurring within the tweet. TfidfVectorizer builds on this idea, but normalizes frequencies such that words that occur frequently within a tweet, and infrequently within the whole corpus, are given higher weight. The other two methods represent tokens using word embedding, i.e. M-dimensional vectors, which allow to measure distance between words. However, since both methods build on non-contextual word embeddings, these methods do not take surrounding words into account.

Binary Classification: Hate Speech Detection

The exploration of classifiers involved running five classifiers and comparing results from varying compatible parameters and features. This included SGDClassifier, SVC, Naive Bayes (MultinomialNB), BernoulliNB and ComplementNB. RandomForestClassifier and MLPClassifier were disqualified due to insufficient performance and extensive computation time.

Extraction method	CountVectorizer		TfidVectorizer		LSA		Word2Vec	
Classifier/Datasets	Val	Test	Val	Test	Val	Test	Val	Test
ComplementNB	0.71	0.60	0.70	0.61	-	-	-	-

Table 2: Validation and test scores, metric used: F-score

Cross-validation was applied to evaluate the selected models. As primary evaluation metric, it was decided to use F-score which summarizes recall and precision into a single value by taking the harmonic

[3] https://www.aclweb.org/anthology/S19-2007.pdf

[4] https://appen.com/solutions/training-data/

mean of both scores. The highest F-score was achieved by Complement-NB evaluated on features from CountVectorizer with a score of 0.71 (see Table 2). It managed to achieve a recall of 0.8, whilst keeping precision at 0.64. A detailed summary of all metrics can be found in Table 2. Several other models, e.g. BernouliNB, managed to achieve very high precision (up to 0.89), but performed poorly on recall. Studying the test results from table 2, we can see that our selected model achieved very high recall, but considerably low precision. This balance indicates that a lot of tweets are being classified as hateful, although many of these are actually not. This might be acceptable in cases where identifying as many hateful tweets as possible is more important than accidentally categorizing non-hateful tweets as hateful. Moreover, results demonstrated how features played a crucial role in model performance. For instance, SGDClassifier_hinge achieved on training dataset F-score 0.7 when based on CountVectorizer, however, only 0.63 for LSA. This highlights the importance of implementing proper features as a baseline for fine tuning model parameters.

Dataset/Metric	Accuracy	Precision	Recall	F-score
Validation	0.73	0.64	0.80	0.71
Test	0.46	0.44	0.97	0.60

Table 3: Summary of ComplementNB model fitted on features from CountVectorizer

Multi-class Classification: Sentiment Analysis

Again, cross validation was applied for the multi-class task to evaluate the selected models' performance on training data using macro averaged recall (MAR) as the primary metric. This metric was chosen since it summarizes results for multiple classes into one number and is least sensitive to imbalance.

Extraction methods	CountVectorizer		TfidVectorizer		LSA		Word2Vec	
Classifier/Datasets	Val	Test	Val	Test	Val	Test	Val	Test
SGDClassifier_hinge	0.59	0.53	0.51	0.47	0.45	0.43	0.49	0.54

Table 4: Validation and test scores, metric used: macro averaged recall

For this task, three different classifiers were run, again comparing results from varying compatible parameters and features. This included SGDClassifier, Naive Bayes (MultinomialNB) and ComplementNB. The SGDClassifier yielded the highest MAR score of 0.59 (see Table 4). This was run with the 'loss' function using the hinge method and CountVectorizer features. Table 4 display the model's performance for all features. This clearly demonstrates the impact of feature representation on model performance, as scores reached a low of 0.45 for LSA. Examining the results for the SGDClassifier with hinge loss revealed a poor recall for tweets with negative sentiment with a score of 0.34. From the confusion matrix it became apparent, that this was due to negative tweets being mislabeled as neutral (see Figure 2). Another explanation might be that negative labels were underrepresented within the training data. Finally, after evaluating on test data, the model obtained a MAR score of 0.53, suggesting that the model was slightly over-fitted. Again, the dive on performance was due to poor recall for tweets with negative sentiment. The results show that positive sentiment was easier

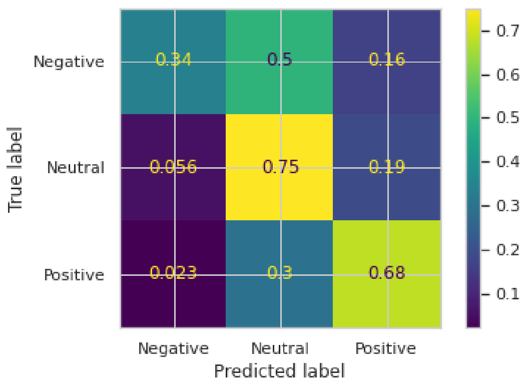

Figure 2: Normalized CM for SGDClassifier_hinge evaluated on features from CountVectorizer

for the model to distinguish from neutral language, while positive and negative sentiment were rarely confused. This also align well with issues that presumably arise for human interpreters.

Dataset/Metric	Accuracy	F-score Macro	Recall Macro
Validation	0.66	0.61	0.59
Test	0.59	0.53	0.53

Table 5: Summary of SGDClassifier_hinge fitted on features from CountVectorizer

Concluding Remarks and Future Work

The main part of this project consisted of two classification tasks. Results were a product of a highly exploratory approach, including the exploration of various models, feature extraction methods and relevant parameters. For hate speech detection, the Complement-NB model evaluated on features from CountVectorizer proved most effective, with the highest F-score of 0.71 for validation and 0.60 for test. For sentiment analysis, the SGDClassifier achieved the highest macro averaged recall of 0.59 for validation and 0.53 for test, while using 'loss' function based on hinge method and evaluated on CountVectorizer features. The result of these experiments demonstrated the importance of preprocessing and feature extraction for the further success of models. Exhaustive tokenization was crucial to manage data sparseness, but proved to be a difficult task - especially for highly contextual and idiosyncratic texts like tweets. This process could be further improved by introducing lemmatization and tools for spelling correction. Feature extraction processes revealed how the performance of models highly depended on the type and quality of features. For future work, test could be run using contextual word-embeddings to possibly improve model performance. Moreover, the choice of models and corresponding parameters also proved to be a challenging part. This was mainly due to time and computation constraints since certain classifiers were too expensive to implement. To improve this process, grid search could appropriately be applied to select the best parameters for a given model in future research. Finally, manual annotation procedures exemplified the core challenges of decoding the ambiguities of natural language, as disagreement prevailed among annotators. This is a crucial experience for any data scientist working within the NLP field, since it illustrates the innate complexity of decoding and interpreting natural language - even for humans.