Exercise 3: Hate Speech Detection

Bruna Dujmovic, Nesrine Haddar

January, 2021

Our task was to develop, analyse and compare multiple approaches for detecting hate speech in social media posts or comments and identify key words responsible for that detection. We developed three classification models, each of which was trained on three kinds of inputs, respectively.

1 Data set

We used a data set of 24.783 English-language tweets which were labeled by CrowdFlower users as "hate speech", "offensive language" or "neither". It is imbalanced as only about 6% of the tweets were labeled as hate speech, while around 77% are labeled offensive. Apart from just detecting hate speech, this data set enabled us to see how well our models differentiate between language considered as hateful vs. offensive.

2 Setup and tools

The project solution consists of a Jupyter Notebook with code implemented in Python 3.6+. Libraries such as Pandas and NumPy were used to access and preprocess the data. The Natural Langauge Toolkit (NLTK) was useful in the preprocessing stage, since it contains implementations of common NLP tasks. Results and data were visualized with the help of Matplotlib, WordCloud, and Yellowbrick. Classification models for hate speech and offensive language detection were built using the machine learning library scikit-learn.

3 Data Preprocessing

Tweets were preprocessed in the following way: the elements not considered to be plain text (URLs, mentions, special symbols and emojis) along with Twitter-specific words (RT, FAV) were removed as they didn't seem relevant for the classification task. Hashtags, which could contain important text, were segmented into words and replaced by strings consisting of those words separated by whitespaces.

In order to feed the data to our classification models, we first had to separate each tweet into smaller parts called tokens. Tokens were obtained by splitting the tweet on whitespaces and punctuation characters and by splitting standard contractions. The resulting tokens were lowercased. Extra whitespaces, punctuation characters, and stopwords (common English words such as "the", "is", "are", etc.) were removed.

4 Feature extraction

Since classification models don't accept textual input, we had to convert the tokenized tweets into a form that they know how to process (i.e. into feature vectors). Three different approaches were used – Bag-of-Words, TF-IDF, and Word2Vec.

The Bag-of-Words method encoded each tweet as a vector of n-gram (sequence of n adjacent tokens) frequencies (i.e. occurrences of that n-gram in a given tweet). The length of each feature vector is equal to the number of unique n-grams in the data. We used a combination of unigrams, bigrams, and trigrams, and considered only the top 10.000 most frequent n-grams.

TF-IDF also produces vectors of that length, but it takes into account both the occurrences in the given tweet and in the entire data set. Here we again considered only the top 10.000 most frequent unigrams, bigrams, and trigrams.

Word2Vec is a popular text representation method which uses a neural network model to obtain dense vectors for each token in the data set. The feature vector of a tweet was then obtained by averaging the vectors of tokens present in that tweet.

5 Models and performance metrics

Three classifiers were used – Logistic Regression, Random Forest, and Gradient Boosting. These models were chosen due to their good performance on similar hate speech detection problems. Additionally, these models allow for tuning to avoid overfitting in case of high-dimensional data sets, which is our case, and they are pretty simple and easy to interpret.

Since each of the used models have different hyperparameters which impact their performance, we chose one hyperparameter per model to tune as this task requires a lot of computational power. The parameter tuning was done by performing a cross-validation grid-search over a given parameter grid (scikit-learn's `GridSearchCV`).

The data set was first split into training and test sets (80-20 split), after which the training set was split into 5 subsets, keeping the proportions of each label the same as for the original data set. `GridSearchCV` trained a model for each combination of the 5 subsets and model parameters, of which we chose to analyse the best found model.

Model performance was evaluated on the test set using standard performance metrics – precision (correct predictions of the label l / all predictions of l), recall (correct predictions of l / cases where label actually is l), and F1-score (a weighted harmonic mean of precision and recall). These metrics were especially useful since we are dealing with an imbalanced data set. Confusion matrices were plotted to help visualize the performance.

6 Results

As can be seen in the summarizing table below, the models seem to deliver very good results, especially when classifying "offensive language" and "neither" tweets.

The performance on "hate speech" is mediocre at best across models and representations, and particularly bad for Gradient Boosting, as it is often confused with "offensive language". The reason behind this, we assume, is the imbalanced data that we had – only a few tweets had the label "hate speech", and there is a large similarity between the language used in that class and "offensive language", which makes it difficult for the models to differentiate between them.

The Logistic Regression model seems to have the best performance across the representations, and Gradient Boosting the worst.

In all cases, slurs and similar derogatory words turned out to be the most important features.

Using Bag of Words									
	Logistic regression			Random forest			Gradient boosting		
	precision	recall	f1_score	precision	recall	f1_score	precision	recall	f1_score
hate speech	0.33	0.61	0.43	0.33	0.66	0.44	0.26	0.2	0.22
offensive language	0.97	0.87	0.92	0.98	0.84	0.9	0.92	0.93	0.92
neither	0.8	0.93	0.86	0.72	0.95	0.82	0.81	0.82	0.81

Using TF-IDF									
	Logistic regression			Random forest			Gradient boosting		
	precision	recall	f1_score	precision	recall	f1_score	precision	recall	f1_score
hate speech	0.34	0.63	0.44	0.33	0.66	0.44	0.26	0.2	0.22
offensive language	0.97	0.87	0.92	0.98	0.84	0.9	0.92	0.93	0.92
neither	0.79	0.93	0.86	0.72	0.95	0.82	0.81	0.82	0.81

Using W2V									
	Logistic regression			Random forest			Gradient boosting		
	precision	recall	f1_score	precision	recall	f1_score	precision	recall	f1_score
hate speech	0.22	0.67	0.34	0.41	0.42	0.41	0.46	0.2	0.28
offensive language	0.98	0.78	0.86	0.93	0.9	0.92	0.91	0.95	0.93
neither	0.73	0.91	0.81	0.73	0.83	0.78	0.8	0.79	0.79