NLP Test

Ahmad Ammari

2 September 2016

Data Science Task

Objective

In this task, I have been asked to build a predictive model which classifies social signals ‘tweets’ to three categories:

- hate speech
- offensive/insulting language
- neutral

I have been provided with two datasets in two .xlsx files:

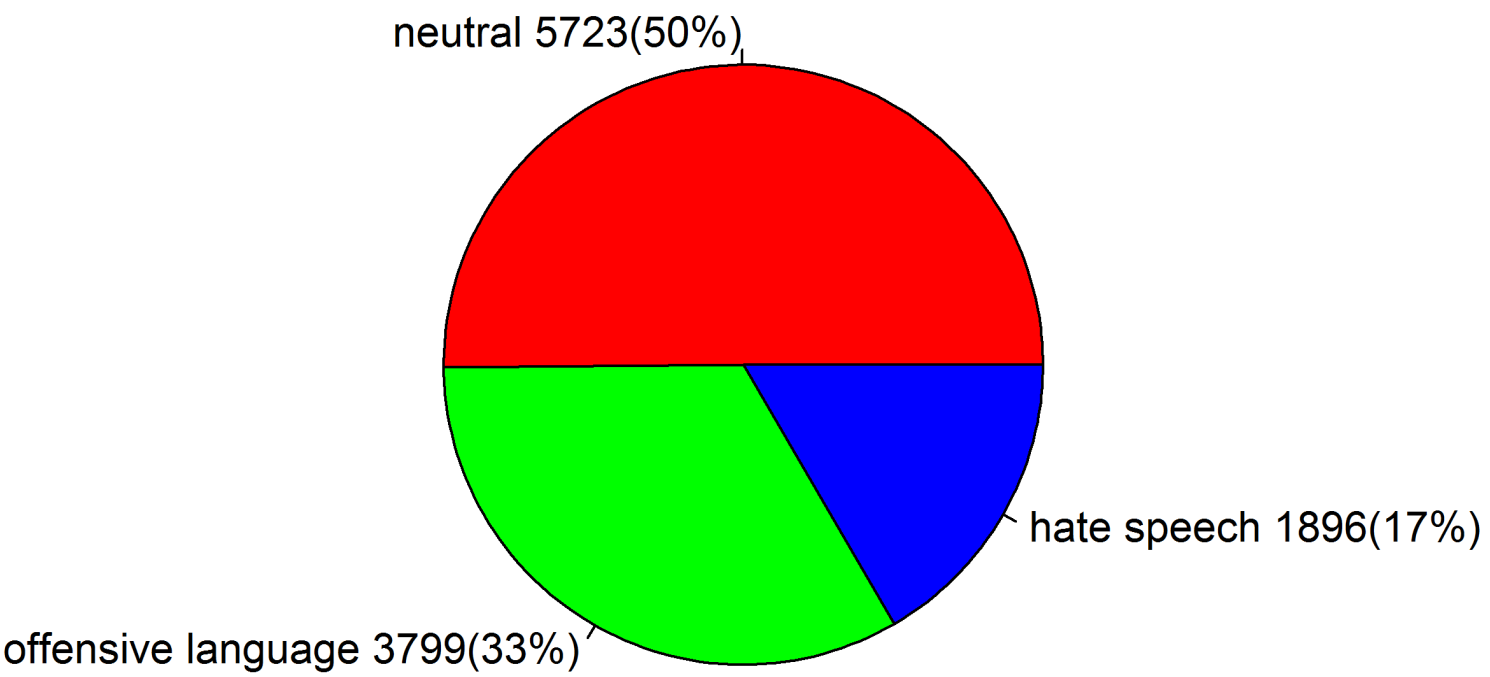
- Labelled set: collection of tweet texts and ‘ground truth’ annotations, where each tweet has been annotated with one of the three categories.
- Unlabeled set: collection of tweet texts with unknown categories.

The objectives are:

- Perform exploratory analysis on the provided data to identify features and techniques that would be relevant to building the predictive model, choose adequate performance metrics to compare the various classifiers, and draw expectations for the performance of the model on unseen data.
- Design and implement a predictive service over HTTP for the chosen model and compute predictions for the unlabelled test dataset.

Exploratory Data Analysis

Frequency and Proportion of the tweet categories in the labeled set



As the proportions of all the tweet categories in the labeled set are statistically significant:

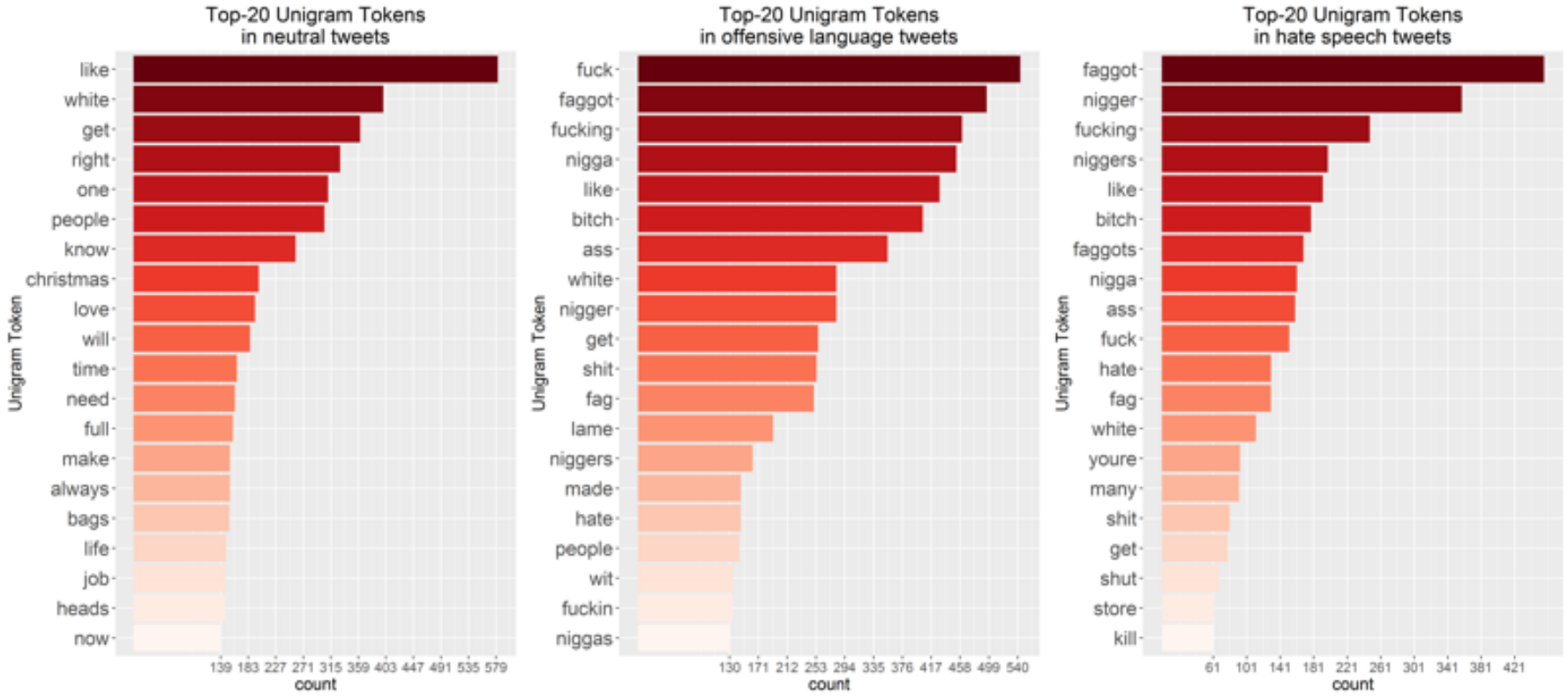
- There is no inclination to address the problem as an anomaly detection problem, but as a classification problem.
- There is a potential of predicting the tweet category based on the tokenized N-gram features of the tweet text.

N-gram Frequency Analysis

In this section, we will develop three frequency N-gram feature models (Unigram, Bigram, and Trigram) and inspect the top-20 features of the tweets annotated by each of the label categories in every model.

Unigram Model Features

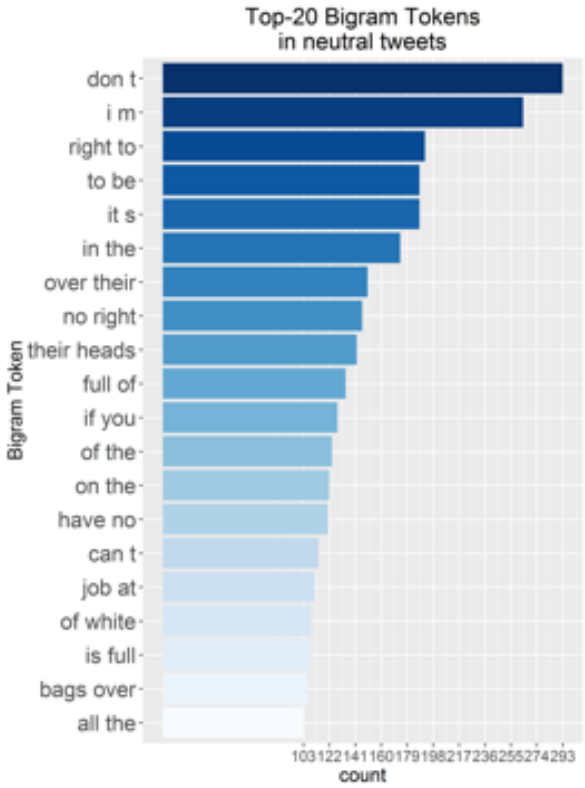
The top-20 unigram features with their counts in the labeled tweets for each category are depicted in the bar charts below:

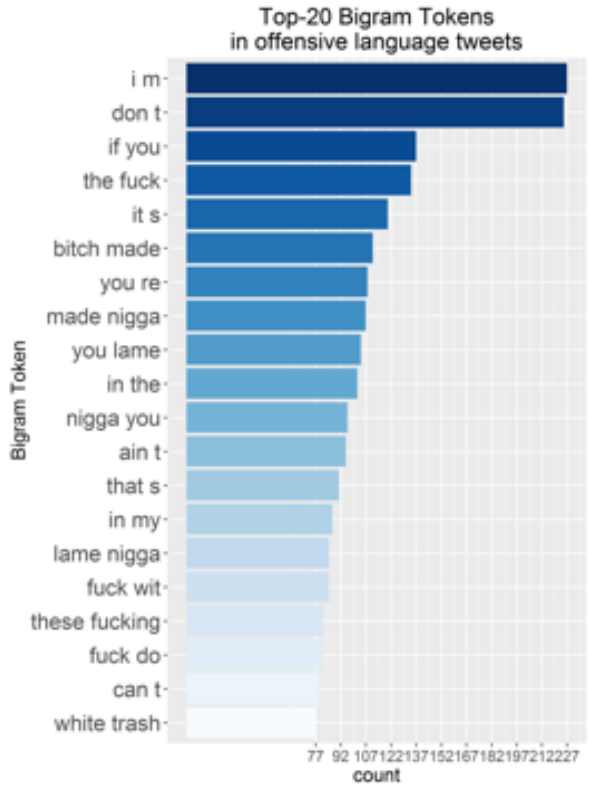


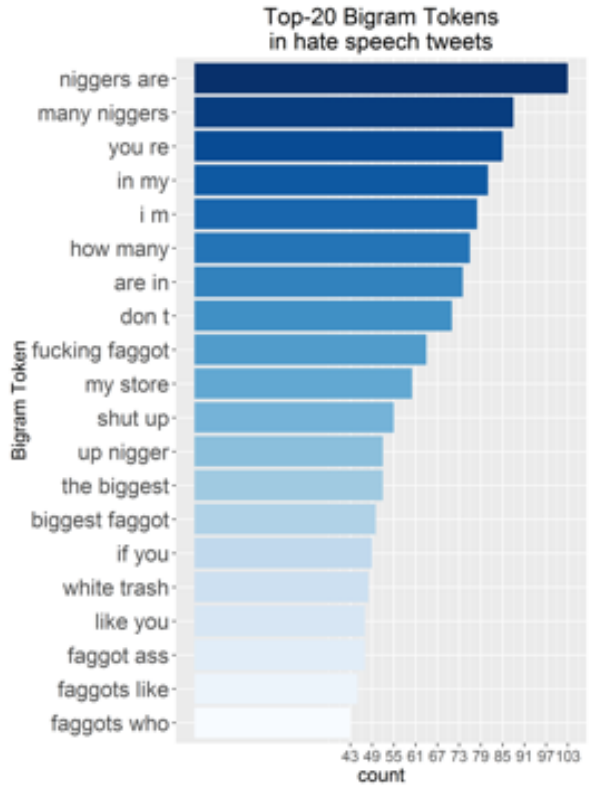
As noted, there is a good distinction between the unigram tokens in the *neutral* category from one side and those in the *offensive language* and *hate speech* categories on the other side. However, the distinction becomes less clearer between the two non neutral labels. To further investigate this, we will compare between the top weighted unigram features in each category through a word cloud inspection.

Bigram Model Features

As the unigram features did not provide a strong predictive discrepancy between the non neutral categories, we will observe the top-20 bigram features with their counts in the same tweets for each category to see if the discrepancy between the features in the *offensive language* and *hate speech* categories becomes clearer. For that, we will not filter out the punctuation, numbers, or stop words in order to preserve the sequence of tokens in the tweet texts.

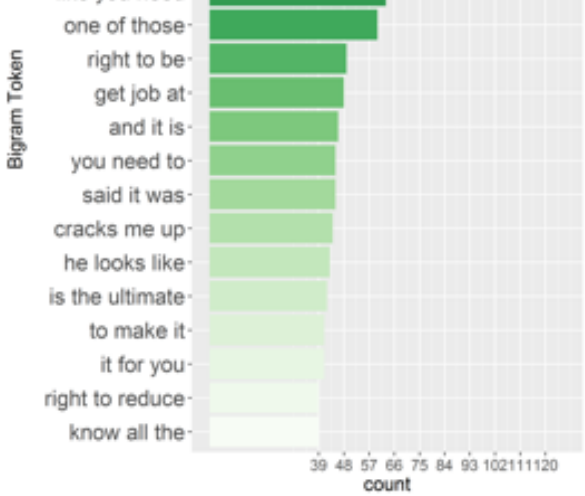






Trigram Model Features

Similarly, we will observe the top-20 trigram features with their counts in the labeled tweets for each category, again with preserving the token sequence by keeping the numbers and stop words.



Recommendations for modelling

- Both unigram, bigram, and trigram tokens could be candidate features in modelling the predictive model as the frequency models shown ealier suggest a promising discrepancy of the token weights across the different label categories.
- As the proportions of the different label categories are all statistically significant, we will address the problem as a text classification problem.
- As the case of the label categories are not equal in proportions, we will normalize weights to instruct the classifier not to give more prior probability to the label category that has the vast majority of the tweets (neutral). An alternative approach is to use Downsampling / Upsampling (http://www.simafore.com/blog/handling-unbalanced-data-machine-learning-models) as this may improve the classification accuracy when the category proportions are unbalanced.

- As the problem is being addressed as a classification, 75% of the labeled data will be used for training and 25% will be used for evaluation. The proposed evaluation metrics are confusion matrix, precision, recall and overall accuracy. The expectation on the model performance on unseen data depends on the computed confidence intervals for the predictions of each label category.

Service Design

In this section, we list the classification algorithms used and the N-gram variations tried with each algorithm. In addition, we present the evaluation for all the tried models and describe the implemented classification service based on the chosen model.

Algorithms Used

The following two algorhtims that are well known to provide good performance in text classification:

- Multinomial Naive Bayes (http://en.wikipedia.org/wiki/Naive_Bayes_classifier): simple and fast algorithm that has very good performance in most cases.
- Support Vector Machines (http://en.wikipedia.org/wiki/Support_vector_machine): more complex algorithm, slightly slower than Naive Bayes but delivers a higher accuracy in general.

N-Gram Variations

For each of the two classification algorithms selected above, we will train three models using the *unigram*, *bigram*, and *trigram* feature vectors, respectively, which will be generated using 75% of the labeled data. The remaining 25% of the data will be used for evaluation.

Evaluation

As it is required to implement the classifier as a service, a MonkeyLearn (http://www.monkeylearn.com/) free account has been used to model all the tried classifiers for evaluation and also for implementing the final selected classifier to compute the predictions on the unlabeled data. The R script in the file *EvaluateTriedModels.R* randomly splits the training data into 75% train and 25% test, then uses the test portion to evaluate each of the six trained models. It also includes information about the service specifications, such as the request headers and the service endpoint.

Notes: The models have been trained in offline mode using the MonkeyLearn Platform (https://app.monkeylearn.com/main/classifiers/cl_znZbrRDB/tab/tree-sandbox). The free MonkeyLearn account has usage limit of 3000 observations for the training data.

The following evlauation metrics have all been computed by classifying the same 25% random split of the labeled data. The confusion matrices computed on the 75% split of the training data, which have been used to build the models, can also be inspected as PNG images in the folder *triedModels* that was submitted with this report.

Multinomial Naive Bayes classifier using Unigrams as Feature Vectors

Actual / Predicted	**Predicted Neutral**	**Predicted Offensive**	**Predicted Hate**	**Class Precision**	**Class Recall**	**Class F1 Score**
Neutral Cases	TNeutral: 936	FOffensive: 347	FHate: 167	83.35 %	64.55 %	72.76 %
Offensive Cases	FNeutral: 135	TOffensive: 471	FHate: 314	48.11 %	51.2 %	49.61 %
Hate Cases	FNeut: 52	FOffensive: 161	THate: 266	35.61 %	55.53 %	43.39 %

overall model accuracy: 58.72 %

Multinomial Naive Bayes classifier using Bigrams as Feature Vectors

Actual / Predicted	Predicted Neutral	Predicted Offensive	Predicted Hate	Class Precision	Class Recall	Class F1 Score
Neutral Cases	TNeutral: 1,011	FOffensive: 283	FHate: 156	84.25 %	69.72 %	76.3 %
Offensive Cases	FNeutral: 134	TOffensive: 477	FHate: 309	53 %	51.85 %	52.42 %
Hate Cases	FNeut: 55	FOffensive: 140	THate: 284	37.92 %	59.29 %	46.26 %

overall model accuracy: 62.2 %

Multinomial Naive Bayes classifier using Trigrams as Feature Vectors

Actual / Predicted	Predicted Neutral	Predicted Offensive	Predicted Hate	Class Precision	Class Recall	Class F1 Score
Neutral Cases	TNeutral: 528	FOffensive: 855	FHate: 67	82.5 %	36.41 %	50.52 %
Offensive Cases	FNeutral: 77	TOffensive: 668	FHate: 175	36.82 %	72.61 %	48.86 %
Hate Cases	FNeut: 35	FOffensive: 291	THate: 153	38.73 %	31.94 %	35.01 %

overall model accuracy: 47.35 %

Support Vector Machines classifier using Unigrams as Feature Vectors

Actual / Predicted	Predicted Neutral	Predicted Offensive	Predicted Hate	Class Precision	Class Recall	Class F1 Score
Neutral Cases	TNeutral: 1,283	FOffensive: 106	FHate: 61	87.1 %	88.48 %	87.78 %
Offensive Cases	FNeutral: 138	TOffensive: 463	FHate: 319	67.2 %	50.33 %	57.55 %
Hate Cases	FNeut: 52	FOffensive: 120	THate: 307	44.69 %	64.09 %	52.66 %

overall model accuracy: 72.06 %

Support Vector Machines classifier using Bigrams as Feature Vectors

Actual / Predicted	Predicted Neutral	Predicted Offensive	Predicted Hate	Class Precision	Class Recall	Class F1 Score
Neutral Cases	TNeutral: 1,226	FOffensive: 143	FHate: 81	82.73 %	84.55 %	83.63 %
Offensive Cases	FNeutral: 193	TOffensive: 451	FHate: 276	62.29 %	49.02 %	54.86 %

Hate Cases	FNeut: 63	FOffensive: 130	THate: 286	44.48 %	59.71 %	50.98 %

overall model accuracy: 68.9 %

Support Vector Machines classifier using Trigrams as Feature Vectors

Actual / Predicted	Predicted Neutral	Predicted Offensive	Predicted Hate	Class Precision	Class Recall	Class F1 Score
Neutral Cases	TNeutral: 1,328	FOffensive: 71	FHate: 51	64.65 %	91.59 %	75.8 %
Offensive Cases	FNeutral: 469	TOffensive: 291	FHate: 160	67.52 %	31.63 %	43.08 %
Hate Cases	FNeut: 257	FOffensive: 69	THate: 153	42.03 %	31.94 %	36.3 %

overall model accuracy: 62.2 %

Model Selection and Service Implementation

The selection of a model for implementation highly depends on the most importance evaluation metric(s) for our needs:

- SVM classifier using Unigrams and Bigrams as feature vectors gave us the *highest overall model accuracy* of 72.06 % and 68.9 % respectively. There is a potential therefore to obtain higher overall accuracy by modelling an SVM classifier using a mixture of top frequently occuring Unigrams and Bigrams. We should select the SVM-Unigram if the three label categories are equal in importance.
- SVM classifier using Trigrams as Feature Vectors gave us the *highest recall* on the *neutral category* with 91.59 %. We sould select this classifier if missing neutral tweets is highly costly. SVM using Unigrams as Feature Vectors give us the *highest precision* on the *neutral category* with 87.1 %. We should select this classifier if misclassifying tweets as neutral is highly costly.
- MNB classifier using Trigrams as Feature Vectors gave us the *highest recall* on the *offensive language category* with 72.61 %. We sould select this classifier if missing offensive language tweets is highly costly. SVM using Trigrams as Feature Vectors give us the *highest precision* on the *offensive language category* with 67.52 %. We should select this classifier if misclassifying tweets as offensive language is highly costly.
- SVM classifier using Unigrams as Feature Vectors gave us the *highest recall* and *highest precision* on the *hate speech category* with 64.09 % and 44.69 %, respectively. We should consider this classifier if detection of hate speech tweets is the most important factor in our classification problem.
- For demonstration, we assume that *hate speech* detection is the most important criteria and therefore we select the *SVM-Unigram* classifier for service implementation on MonkeyLearn to compute the predictions for the unlabeled data. Furthermore, this classifier provides the *highest F1 score* for the three label categories, which is a metric that combines the category precision and recall.
- The R script in the file *ComputePredictionsUnlabeled.R* uses the selected model, which is implemented as a MonkeyLearn service, to compute the predictions on the unlabeled data. It also includes information about the service specifications, such as the request headers and the service

endpoint. The CSV file *UnlabeledDataPredictions.csv* contains the IDs of the unlabeled tweets as well as their predicted classes and the prediction probabilities as provided by the implemented model.

Weight optimization for ‘hate_speech’ category

There are a number of methods that can be used to give more importance to a specific label category performance (precision / recall) for e.g. *hate_speech*. One method is to treat the model as a probabilistic classifier and change the prediction probability threshold for the category of interest. As both the Multinomial Naive Bayes and Support Vector Machines algorithms provide a probability for the tweet to belong to each of the candidate classes, we can:

-Improve the *precision* of the *hate speech* class by increasing the threshold where the classifier determines that the tweet is to be classified to that class, thus reducing the *false positive* rate of that category.

- Improve the *recall* of the *hate speech* class by decreasing the threshold where the classifier determines that the tweet is to be classified to that class, thus reducing the *false negative* rate of that category.

Confidence intervals for the unseen data predictions

As discussed in the previous section, both the Multinomial Naive Bayes and Support Vector Machines algorithms provide a probability for the tweet to belong to each of the candidate classes. Given the assumption that these probabilities form a normal distribution, the lowerbound and upperbound probabilities of the (e.g. 95%) confidence interval for each label category can be defined as:

lbd = x - (z X SE)

and

ubd = x + (z X SE)

where:

- lbd: the lowerbound probability for the label category
- ubd: the upperbound probability for the label category
- x: the mean of the label category probabilities.
- z: the critical value (equals 1.96 for a 95% confidence level)
- SE: The Standard Error for the label category, which equals:

SE = s / sqrt(n)

where:

- s: the standard deviation of the label category probabilities.
- n: the number of tweets assigned to the label category.

Continuous Improvement

Training / Validation Data Collection Approaches

- Crowdsourcing Platforms / APIs: CrowdFlower (https://www.crowdflower.com/) is a software as a service platform that allows users to access an online workforce of millions of people to clean, label

and enrich data. Tweets can be uploaded, annotated by the crowd and retrieved back from crowdsourcing jobs using the CrowdFlower API (https://success.crowdflower.com/hc/en-us/articles/202703445-Integrating-with-the-API) in an automated fashion.

- Using the *Wisdom of the Crowd*: For example, Twitter users can annotate tweets with hashtags to indicate an underlying topic, event, or opinion. Public tweets can be streamed in an automated fashion by tracking keyword / hashtag filters such as #HateSpeech and #OffensiveLanguage using the Twitter Streaming API. These can be used to enrich the crowdsourced tweets, where the hashtags can be used as the labels whereas the text JSON key of the tweets to be used as the tweet content.
- Using a boot-strapping training (http://tinyurl.com/gmyxjmr) approach: First, obtain the first version of the classifier by training the model using the crowdsourced tweets. Second, use the first version of the classifier to score additional examples that can be used enrich the initial training set for training a better classifier. Third, feed the scored tweets into a second crowdsourcing job. Fourth, enrich the initial training set with tweets that have matching annotations from both the initial classifier and the crowdsourcing job, and disregard the tweets that have a mismatching annotations. Fifth, use the enriched training set to train a second version model. Sixth, repeat steps 2-5 iteratively until we obtain a satisfactory matching proportion between the automated classifications and crowdsourced annotations.