Hate Speech and Offensive Language

Emanuela Elli, Federica Madon and Tommaso Strada

Master's degree Data Science, University of Milano-Bicocca Course of Text Mining and Search, Academic year 2022-2023

Abstract. The *Hate Speech and Offensive Language* dataset [1] is the subject of this study's **text classification** and **text clustering** sections. The dataset consists of **tweets** that may use *hate speech*, *offensive language* or *neither*.

1 Dataset

The dataset is built on information gathered about CrowdFlower users, a platform for data enrichment with access to an online, on-demand workforce of millions of people who complete tasks that algorithms can't alone.
There are **24783 rows** and **6 columns**. The variables are as follows:

- `count`: Number of CrowdFlower users who coded each tweet (min is 3, sometimes more users coded a tweet when judgments were determined to be unreliable by CF);
- `hate_speech`: Number of CF users who judged the tweet to be hate speech;
- `offensive_language`: Number of CF users who judged the tweet to be offensive;
- `neither`: Number of CF users who judged the tweet to be neither offensive nor non-offensive;
- `class`: Class label for majority of CF users;
- `tweet`: Text of the tweet.

2 Labels

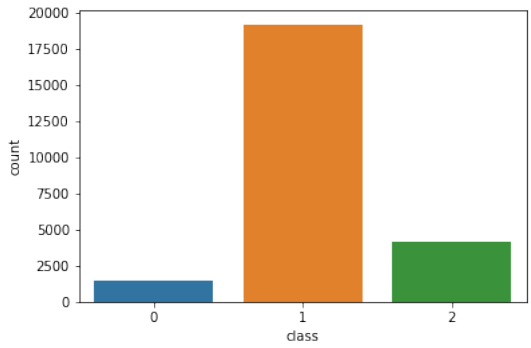

Figure 1. Distributions of the labels within the dataset.

The variable `class` contains the tweets' labels, and it can take one of three different types of values:

1. "0" indicates that the row's tweet contains hate speech. The following tweet is an example: "*We hate niggers, we hate faggots and we hate spics*";
2. "1" indicates that the row's tweet contains offensive language, for example: "*Hairy pussy bitch you the type that got herps*";
3. "2" denotes that the row's tweet does not contain any of the earlier. "*Got my vans on.. My pockets chunky*" is a tweet of this type.

3 Pre-Processing

These pre-processing stages are carried out for text classification and text clustering, respectively:

1. Starting from the labels, a **binary variable** is created and called `type`:
   - "1" for tweets with hate speech or offensive language;
   - "0" for tweets with neither of them;
2. **Lower case** has been applied to every character;
3. The tweets are cleaned up of **extraneous characters** like links, references, punctuation, emoticons, numbers, and excess white space;
4. **Repeating characters** within tweets are removed, if they appear more than twice;
5. A **typographical error** has been found and fixed: "*amp*" has been replaced with "*and*";
6. Remove 2 rows with **NaN values**;
7. **Tokenization** is applied on tweets.

4 Text Classification

"*Can we identify tweets that contain hate speech or offensive language?*" is the research question that guides the task of text classification.

4.1 Dataset

The dataset has been divided into two sections for this task:

- 70% for the **training test** (17346 rows);
- 30% for the **test set** (7435 rows).

Then the **Smote** method (Synthetic Minority Oversampling Technique [3]) is used to balance out the training set's imbalanced classes.

4.2 *Pre-Processing and Text Representation*

Only the training set has been subjected to the pre-processing steps, while the tweets of the test set are only removed of the punctuation. It was decided to use two different *text representations*, such as **TF-IDF** and **Word2vec**.

TF-IDF requires to:

1. Remove the **stopwords** from tweets of the training test;
2. Apply **lemmatization** on tweets, already divided in tokens;
3. Divide tweets in **n-grams** (unigram, bigram and trigram).

Word2vec requires to:

1. Divide tweets in **unigram**;
2. Use a **Google pretrained model**.

4.3 *Models*

Three alternative models were chosen to be applied to each **text representation**:

- XGBoost;
- Naive Bayes;
- SVM.

4.3.1 *Text Classification with TF-IDF*

The model with the best TF-IDF performance is represented by the yellow table cell.

	XGBOOST	Naive Bayes	SVM
Unigram	Accuracy: **0.74** AUC: **0.89**	Accuracy: **0.84** AUC: **0.93**	Accuracy: **0.78** AUC: **0.90**
Bigram	Accuracy: **0.73** AUC: **0.89**	Accuracy: **0.75** AUC: **0.93**	Accuracy: **0.74** AUC: **0.88**
Trigram	Accuracy: **0.73** AUC: **0.89**	Accuracy: **0.75** AUC: **0.92**	Accuracy: **0.73** AUC: **0.88**

Figure 2. Models implemented with TF-IDF.

4.3.2 *Text Classification with Word2vec*

The model with the best Word2vec performance is represented by the yellow table cell.

	XGBOOST	Naive Bayes	SVM
Unigram	Accuracy: **0.95** AUC: **0.98**	Accuracy: **0.90** AUC: **0.92**	Accuracy: **0.95** AUC: **0.97**

Figure 3. Models implemented with Word2vec.

Due to implementation-related computational issues, the classification algorithms in this scenario have not been validated for bigram and trigram. As a result, it may be open to future developments and updates.

4.3.3 *The best model for Text Classification*

The model with the best performance is **XGBOOST Classifier** with Word2vec as text representation and unigram:

- `accuracy = 0.95`;
- `AUC = 0.98`;
- `precision = 0.76` for `0`;
- `precision = 1.00` for `1`;
- `recall = 0.98` for `0` and `recall = 0.94` for `1`.

The following graphs display the **test set results**.

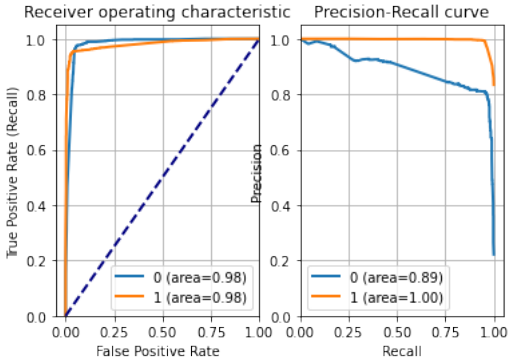

Figure 4. ROC Curve and Recall-Precision Curve.

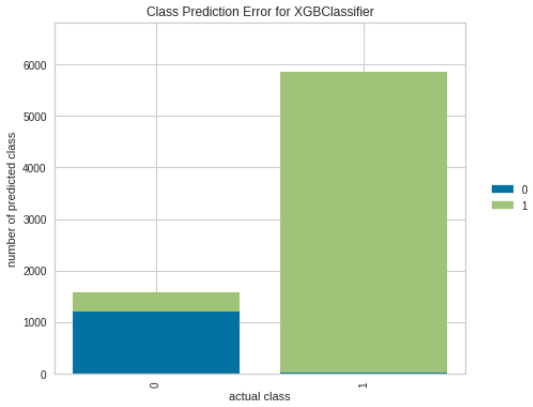

Figure 5. Class prediction error.

5 Text Clustering

"*Is there any division of tweets into clusters?*" is the research question that guides the task of text clustering.

5.1 *Dataset*

A **sample** is extracted from the dataset. This sample respects the **imbalance** between the new two classes of the labels.

5.2 *Pre-Processing and Text Representation*

In this case it was chosen to:

- Apply **pre-processing** steps on all the dataset;
- Remove **stopwords** within the tweets;
- Apply also **lemmatization** step.

For this assignment, just one kind of textual representation is utilized: Word2vec. **Word2vec** requires to:

1. Set up **CBOW** architecture with:
   - `size = 300`;
   - `alpha = 0.03`;
2. Divide tweets in **n-grams** (unigram, bigram and trigram).

5.3 *How many clusters?*

The **Elbow Method** and **Silhouette Method** are computed for each type of n-gram to determine the *optimal number of clusters*. The following plots concern *only* unigrams since the same results are obtained also for bigrams and trigrams.

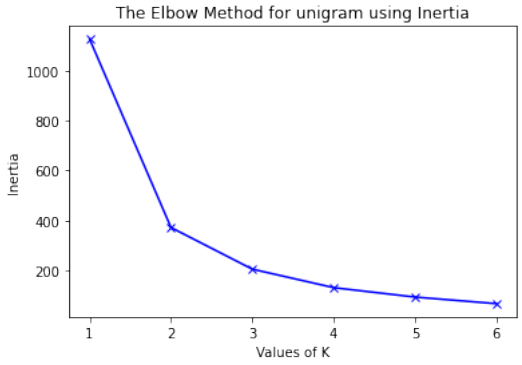

Figure 6. The Elbow Method.

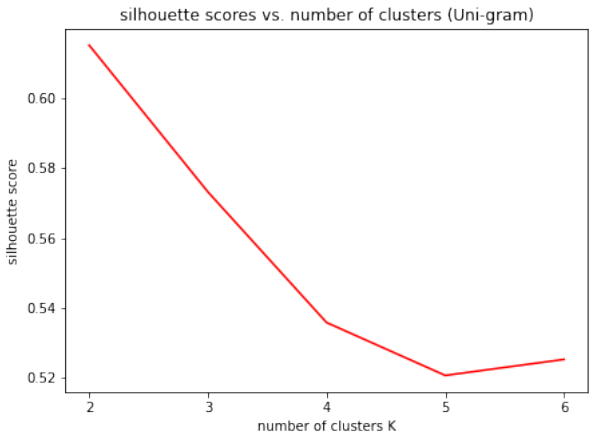

Figure 7. The Silhouette Method.

It turns out that **three clusters** are the ideal quantity for this task.

5.4 *Models*

Two distinct models are put into practice:

- K-Means;
- Agglomerative.

5.4.1 *Text Clustering with Word2vec*

The model with the best Word2vec performance is represented by the yellow table cell.

	K-Means	Agglomerative
Unigram	Silhouette index: **0.15** Davies bouldin index: **1.94**	Silhouette index: **-0.24** Davies bouldin index: **0.58**
Bigram	Silhouette index: **0.74** Davies bouldin index: **0.95**	Silhouette index: **-0.59** Davies bouldin index: **0.56**
Trigram	Silhouette index: **0.96** Davies bouldin index: **6.41**	Silhouette index: **-0.73** Davies bouldin index: **0.55**

Figure 8. Models implemented with Word2vec.

5.4.2 *The best model for Text Clustering*

The model with the best performance is **K-Means** with Word2vec as text representation and bigram. The following plot shows how the algorithm divides tweets into clusters.

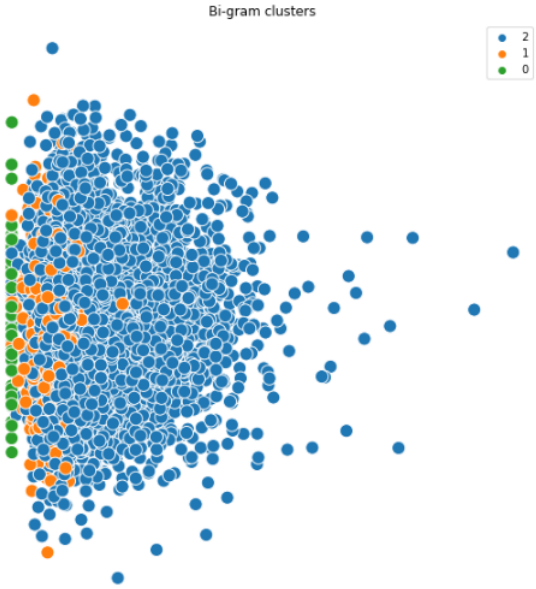

Figure 9. K-Means with bigram.

6 Conclusion and Future Developments

This work was done to acquire knowledge of some of the text mining jobs. The results obtained by incorporating several text representations are not as good as one may hope for, but this is also because of the computing capabilities of the tools used.

As has been shown, for **classification** (even if only for unigrams) with the Word2vec representation we get a considerable improvement on the results for our dataset. As far as the **clustering** task is concerned, however, the available dataset is probably *not suitable* for this type of task since the results obtained in terms of metrics seem to be quite satisfactory but visually they do not seem to show particular patterns or features that allow us to clearly distinguish groups of tweets.

This work can therefore be used as a foundation for making changes and attempting to produce more noteworthy outcomes that can be compared. For instance, the following points could be raised in the first instance:

- Dividing the test set into **validation** and test set with **cross validation**;
- Using a function to correct **spelling mistakes**;
- Using a function to **normalize** all the vocabulary for all the models;
- Implementing Word2Vec classification also with **bigram** and **trigram**;
- Test the chosen model of classification on **another dataset** with tweets.

7 Tools and Library

Google Colab is used to develop the project. The most well-known libraries in use include:

- `Numpy`;
- `Pandas`;
- `Seaborn` and `Matplotlib`;
- `RE` [6];
- `Beautiful Soup` [12];
- `Natural Language Toolkit` [17];
- `scikit-learn` [11];
- `GENSIM` [10];
- `dask` [13].

References

[1] Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber, 'Automated hate speech detection and the problem of offensive language', in *Proceedings of the 11th International AAAI Conference on Web and Social Media*, ICWSM '17, pp. 512–515, (2017).

[2] Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber, 'Automated hate speech detection and the problem of offensive language', in *Proceedings of the international AAAI conference on web and social media*, pp. 512–515, (2017).

[3] Imbalanced Learn, 'Documentation for smote method', in *https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html*.

[4] Amir H Razavi, Diana Inkpen, Sasha Uritsky, and Stan Matwin, 'Offensive language detection using multi-level classification', in *Advances in Artificial Intelligence: 23rd Canadian Conference on Artificial Intelligence, Canadian AI 2010, Ottawa, Canada, May 31–June 2, 2010. Proceedings 23*, pp. 16–27. Springer, (2010).

[5] https://ai.intelligentonlinetools.com/ml/k-means-clustering-example-word2vec/, 'K means clustering example with word2vec in data mining or machine learning'.

[6] https://docs.python.org/3/library/re.html, 'Re'.

[7] https://github.com/Hate-Speech-Detection, 'Hate speech detection for tweets with k8s cluster'.

[8] https://medium.com/@dilip.voleti/classification-using-word2vec-b1d7\9d375381, 'Classification using word2vec'.

[9] https://medium.com/unsupervised-text-clustering-using-natural-langua\ge-processing-nlp, 'Unsupervised-text-clustering using natural language processing(nlp)'.

[10] https://radimrehurek.com/gensim/, 'Gensim'.

[11] https://scikit-learn.org/stable/, 'scikit-learn'.

[12] https://www.crummy.com/software/BeautifulSoup/bs4/doc/, 'Beautiful soup'.

[13] https://www.dask.org/, 'dask'.

[14] https://www.guru99.com/word-embedding-word2vec.html, 'Word embedding and word2vec model with example'.

[15] https://www.kaggle.com/hate-offensive-language, 'Hate and offensive language'.

[16] https://www.kaggle.com/nlp-model-to-predict-hate-speech#Importing-the-dataset, 'Nlp model to predict hate speech'.

[17] https://www.nltk.org/, 'Natural language toolkit'.