Toxic Comment Classification

Project report

1 Definition

1.1 Project Overview

The use of different online platforms (social media, messaging platforms, gaming platforms and mobile phones) has become a part of our everyday life. Negative consequences for mental health could include exposures to online forms of aggression. Cyberbullying is an emerging issue in times of digital technologies. Since adolescent public makes an extensive use of online platforms, they are affected by cyberharassment the most.

Cyberbullying can take on many forms, including personal attacks, harassment or discriminatory behavior, spreading defamatory information, misrepresenting oneself online, spreading private information, social exclusion and cyberstalking [1]. Pew research center [2] and Cyberbulling research center [3] conducted research which involved US teens and now provide more data and visualisations on subject matter.

The comment sections of social media and news outlets have become the new playground for online bullying. A research shows that one out of five online harassment takes place in comment space [4]. Keyboard courage, driven by anonymity and the absence of real world response, may ruin the experience of a lot of users. As a result, many news organizations are even choosing to eliminate comments altogether to avoid this problem.

Users' feedback becomes quickly unmanageable if an online community grows fast, manual moderation of comments seems no longer an option, automatic tools should be used to detect and block hate speech in web discourses. Recent developments in natural language processing, as well as in machine and deep learning, are able to provide such solutions.

Both project idea and data come from the Conversation AI team, a research initiative founded by Jigsaw and Google (both a part of Alphabet). They are working on tools to help improve online conversations. One area of focus is the study of negative online behaviors, like toxic comments (i.e. comments that are rude, disrespectful or otherwise likely to make someone leave a discussion). [5].

The data[1] is available on Kaggle competition page:
https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/data. Three files are in our disposal: `train.csv`, `test.csv`, `test_labels.csv`. These files provide a large number of Wikipedia comments which have been labeled by human raters for toxic behavior. There are six types of toxicity: `toxic`, `severe_toxic`, `obscene`, `threat`, `insult`, `identity_hate`.

1.2 Problem Statement

The goal of this project is to explore the space of possible models that allow to detect different types of toxicity (obscenity, threats, insults, and identity-based hate).

[1]The data set is under CC0, with the underlying comment text being governed by Wikipedia's CC-SA-3.0.

To address the problem of automatic detection of toxic comments we will first try to build a simple and efficient Naive Bayes classifier, an algorithm which is commonly used in text classification. However, since we deal with text data, neural networks (e.g. LSTMs) seem to be a more appropriate choice for this type of input; pretrained embeddings are also often used in this case. We will explore them both in our main model.

Before diving into modeling, we will perform exploratory data analysis and visualize data. These steps should give us some hints on data distributions and potential problems we may face while building classifiers. Since we work with textual data, it needs to be preprocessed and transformed into numerical one. These steps will be discussed in the following sections.

1.3 Evaluation Metrics

We will use several metrics which are generally useful for classification problems [6] to assess the performances of all the models:

- Accuracy score (fraction of correct predictions),
- Precision (the ability of the classifier not to label as positive a sample that is negative) and recall (the ability of the classifier to find all the positive samples),
- ROC AUC score (fraction of true positives out of the positives, true positive rate, vs. the fraction of false positives out of the negatives, false positive rate).

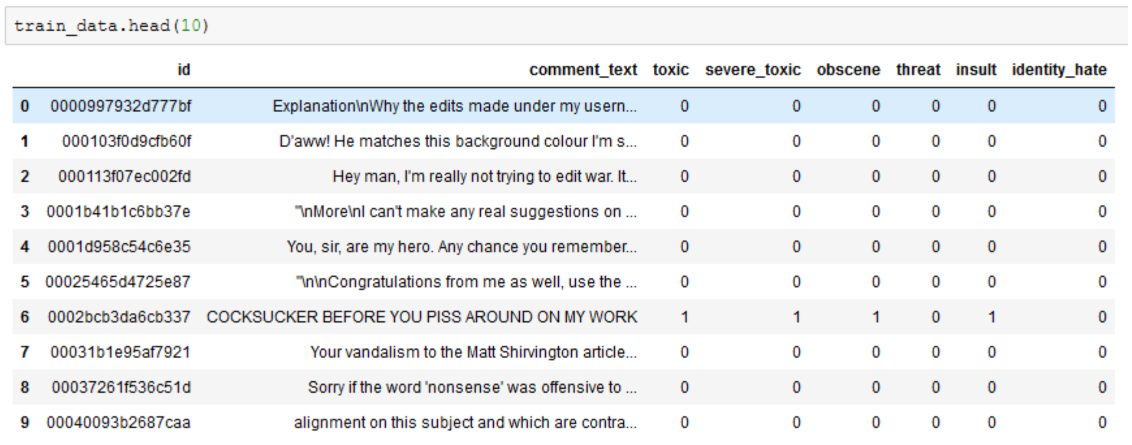
We will study the results by analyzing all these metrics: we hypothesize that, since our data is unbalanced, the three metrics in question might be very different; however, ROC AUC might be the most reliable metric given data distribution peculiarities.

2 Analysis

2.1 Data Exploration

Having loaded and unpacked all the necessary data (`train.csv`, `test.csv`, `test_labels.csv`), we proceed with data diagnostics, such as checking for duplicates and/or missing values. No duplicated or missing values were found in training and test sets.

Next, we load the first ten lines of each file to get an overview of the data.

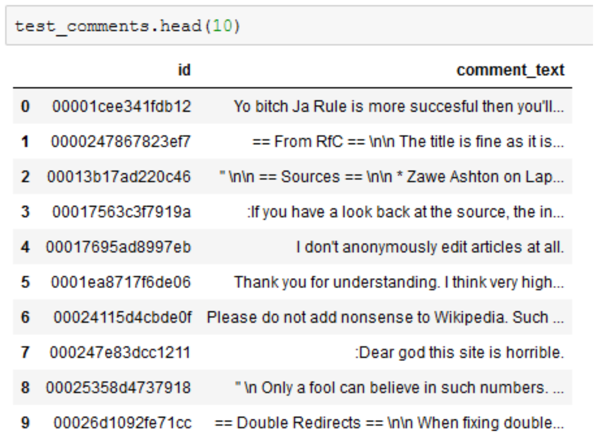


```
train_data.head(10)
```

	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0
3	0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0
5	00025465d4725e87	"\n\nCongratulations from me as well, use the ...	0	0	0	0	0	0
6	0002bcb3da6cb337	COCKSUCKER BEFORE YOU PISS AROUND ON MY WORK	1	1	1	0	1	0
7	00031b1e95af7921	Your vandalism to the Matt Shirvington article...	0	0	0	0	0	0
8	00037261f536c51d	Sorry if the word 'nonsense' was offensive to ...	0	0	0	0	0	0
9	00040093b2687caa	alignment on this subject and which are contra...	0	0	0	0	0	0

Figure 1: Overview of the training set

Figure 1 displays some trends. There are nine of ten first lines where comments are not tagged with any toxicity labels, meaning they are neutral, however, there is no specific label to tag neutrality in comments. Another observation can be made about toxicity labels: they are not mutually exclusive, every toxic comment can be tagged with one or more toxicity labels. Neutrality in this case is an exclusive feature. As for comment text column, we can observe that not only punctuation is present, but also end of line characters. The same observations are true for test set, which is displayed in Figure 2.

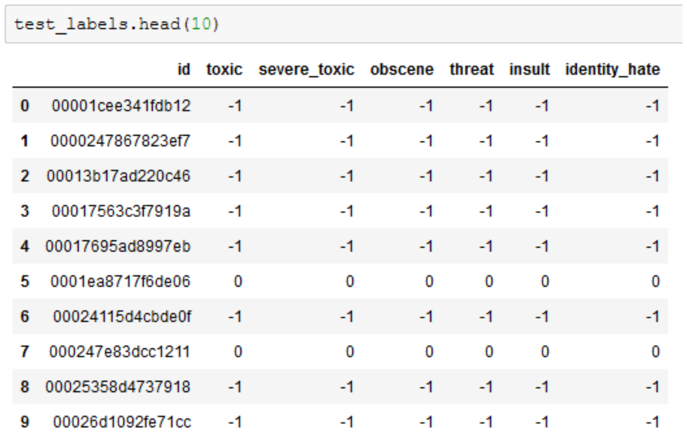


```
test_comments.head(10)
```

	id	comment_text
0	00001cee341fdb12	Yo bitch Ja Rule is more succesful then you'll...
1	0000247867823ef7	== From RfC == \n\n The title is fine as it is...
2	00013b17ad220c46	" \n\n == Sources == \n\n * Zawe Ashton on Lap...
3	00017563c3f7919a	:If you have a look back at the source, the in...
4	00017695ad8997eb	I don't anonymously edit articles at all.
5	0001ea8717f6de06	Thank you for understanding. I think very high...
6	00024115d4cbde0f	Please do not add nonsense to Wikipedia. Such ...
7	000247e83dcc1211	:Dear god this site is horrible.
8	00025358d4737918	" \n Only a fool can believe in such numbers. ...
9	00026d1092fe71cc	== Double Redirects == \n\n When fixing double...

Figure 2: Overview of the test set

Figure 3 shows an overview of test set labels. Comparing to the training set label, negative values can be found here. This was done on purpose by Kaggle competition organizers: to deter hand labeling, the test set contains some comments which are not included in scoring, the value of -1 corresponds to these comments in test labels set.



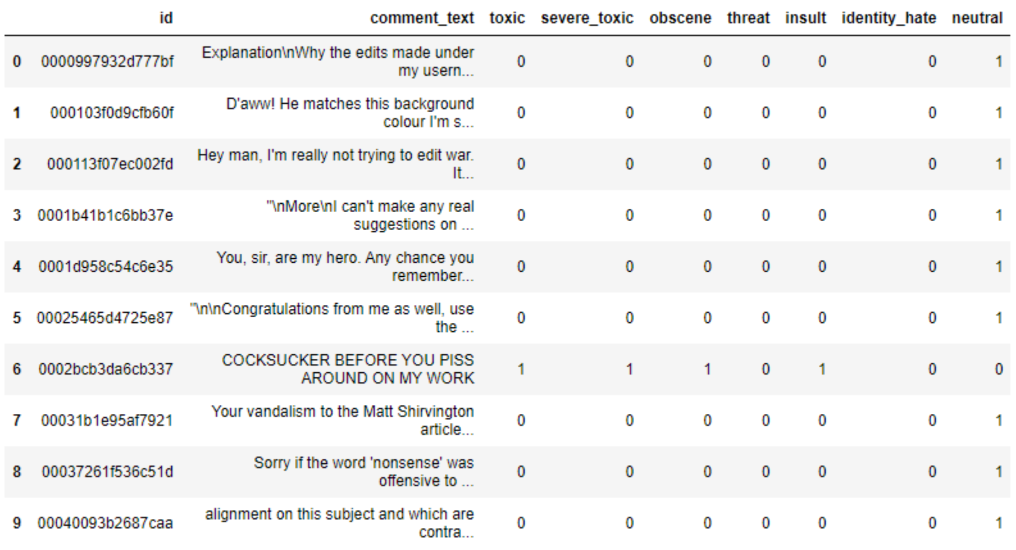
```
test_labels.head(10)
```

	id	toxic	severe_toxic	obscene	threat	insult	identity_hate
0	00001cee341fdb12	-1	-1	-1	-1	-1	-1
1	0000247867823ef7	-1	-1	-1	-1	-1	-1
2	00013b17ad220c46	-1	-1	-1	-1	-1	-1
3	00017563c3f7919a	-1	-1	-1	-1	-1	-1
4	00017695ad8997eb	-1	-1	-1	-1	-1	-1
5	0001ea8717f6de06	0	0	0	0	0	0
6	00024115d4cbde0f	-1	-1	-1	-1	-1	-1
7	000247e83dcc1211	0	0	0	0	0	0
8	00025358d4737918	-1	-1	-1	-1	-1	-1
9	00026d1092fe71cc	-1	-1	-1	-1	-1	-1

Figure 3: Overview of the test labels set

Additionally, we can find an id column in all the 3 sets, this will be particularly useful while working with test data: though in ten first rows the ids of test set and test labels set seem to have linear relation, we will use the id column as a key to concatenate both sets into one.

At this stage, we will deliberately introduce a `neutral` label in both data sets for further analysis, as well as for classification task. An overview of training set with additional `neutral` label is shown in Figure 4.



	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate	neutral
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0	1
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0	1
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0	1
3	0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0	1
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0	1
5	00025465d4725e87	"\n\nCongratulations from me as well, use the ...	0	0	0	0	0	0	1
6	0002bcb3da6cb337	COCKSUCKER BEFORE YOU PISS AROUND ON MY WORK	1	1	1	0	1	0	0
7	00031b1e95af7921	Your vandalism to the Matt Shirvington article...	0	0	0	0	0	0	1
8	00037261f536c51d	Sorry if the word 'nonsense' was offensive to ...	0	0	0	0	0	0	1
9	00040093b2687caa	alignment on this subject and which are contra...	0	0	0	0	0	0	1

Figure 4: Overview of the final training set

To correctly use the test set for models evaluation, all the comments with -1 labels will be removed from the data. The total number of comments in both sets are: 159571 comments in training data set and 63978 - in test set (there were 153164 test comments before cleaning).

2.2 Exploratory Visualization

At this stage we have a data set for classification, where there is only one input (comment) and several outputs (labels of toxicity and neutrality). That allows us to perform summary calculations: total number of comments per each label and a total number of comments per total number of associated labels.

Figure 5 shows that the data classes are highly disproportionate: neutral comments constitute 80.27% of all the data, while the comments tagged with `identity_hate` label represent just 0.003%.

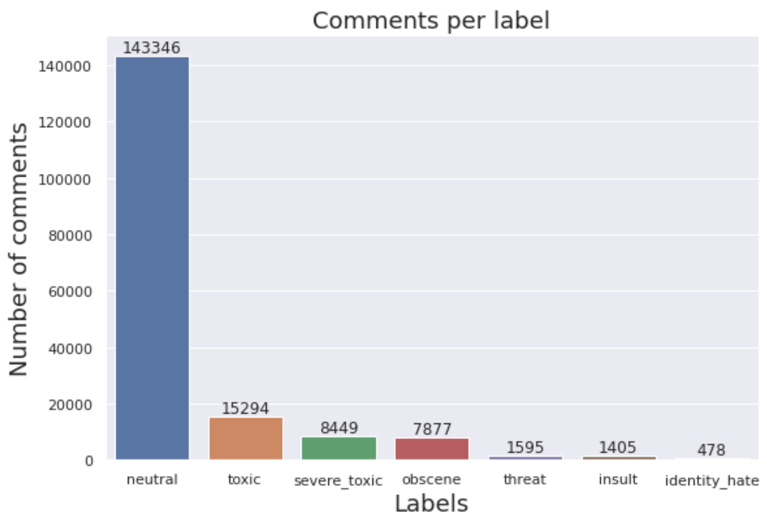


Figure 5: Distribution of comments per each label

Figure 6 reveals another disproportion in data set: 93.82% of comments are tagged only with one label, 0.07% have two labels. Somme comments are tagged with all the six labels of toxicity, but their fraction in data set is extremely low.

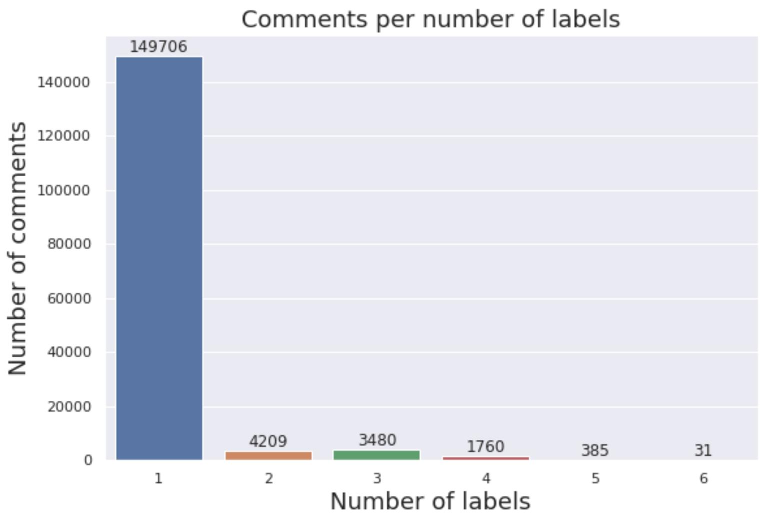


Figure 6: Distribution of comments per number of associated labels

We can also perform word cloud visualizations, thanks to a supplementary data processing step, discussed further down in section 3.1.

Figure 7 shows such a word cloud for neutral comments, the vocabulary used here seem to be overall correct.

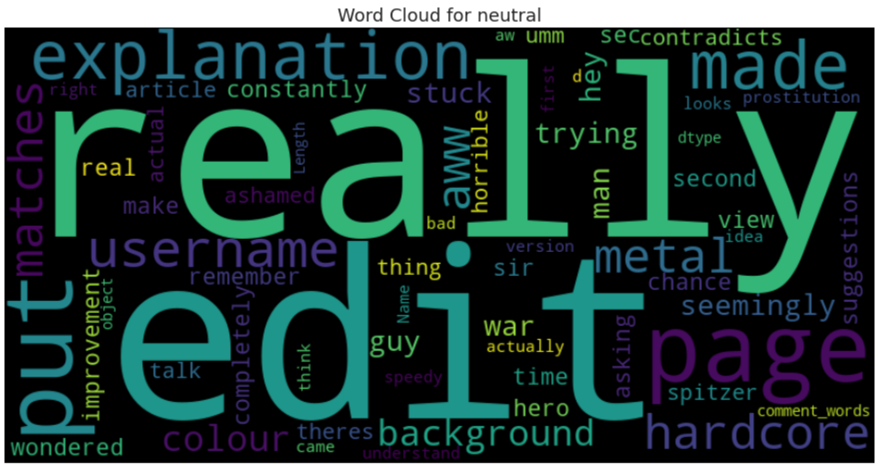


Figure 7: Word cloud for neutral comments

Figures 8 to 13 show word clouds separately for all the six labels of toxicity, all these clouds reveal hate speech lexicon (both noun patterns and hate-related verbs) to target certain groups of people based on their religion, ethnicity, nationality, color and gender[2].

[2]Disclaimer: the word clouds contain text that may be considered profane, vulgar, or offensive.

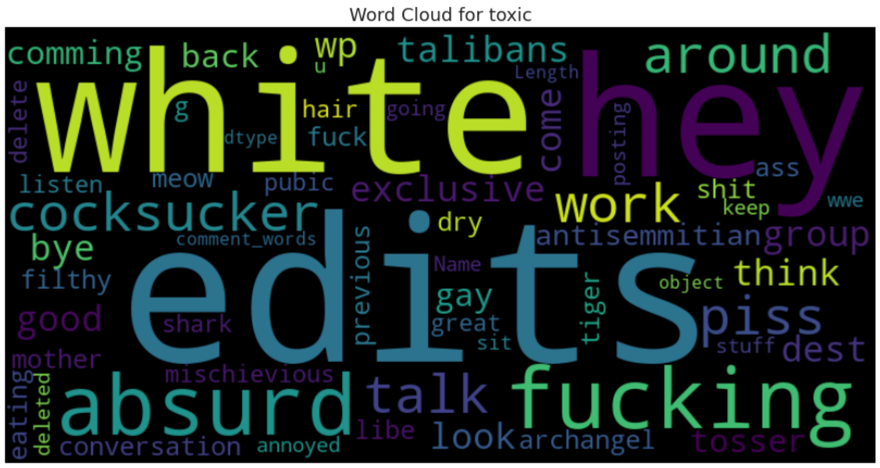


Figure 8: Word cloud for toxic comments

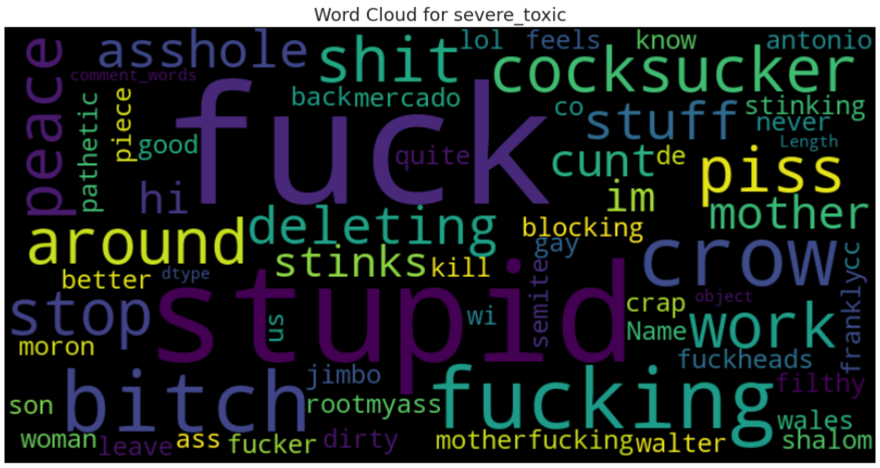


Figure 9: Word cloud for severe toxic comments

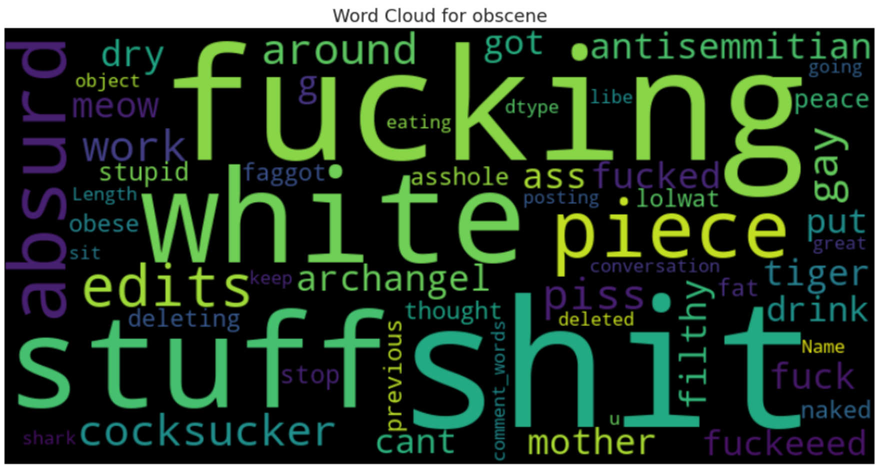


Figure 10: Word cloud for obscene comments

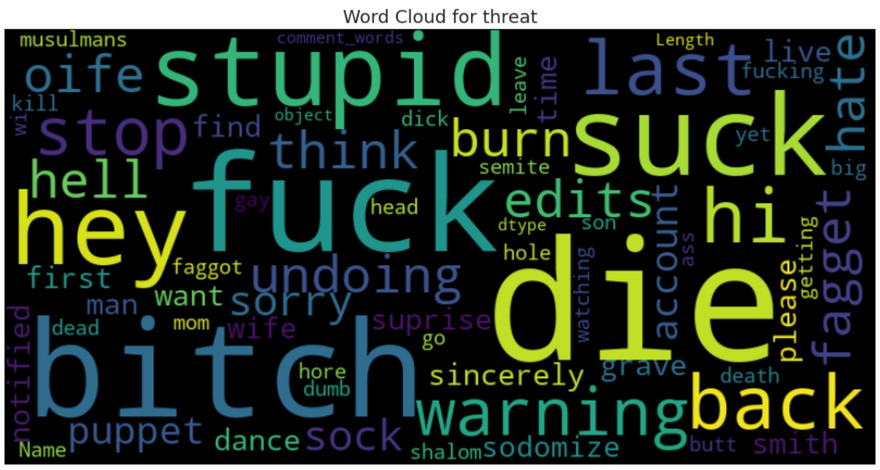


Figure 11: Word cloud for threat comments

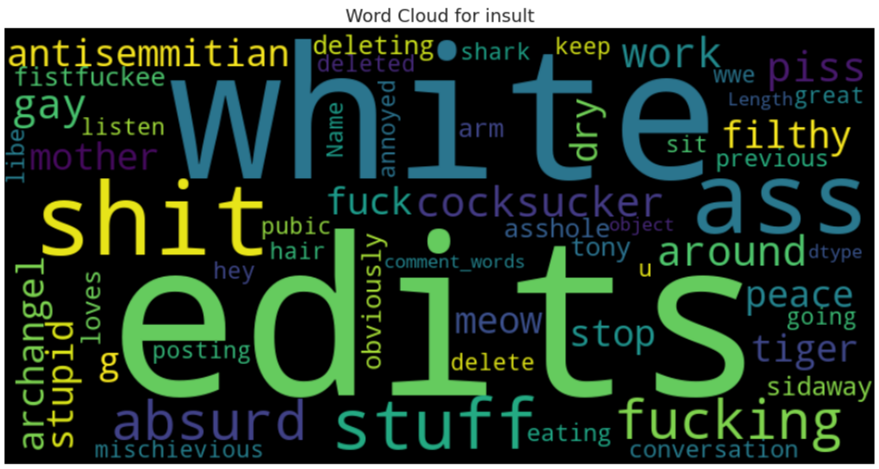


Figure 12: Word cloud for insult comments

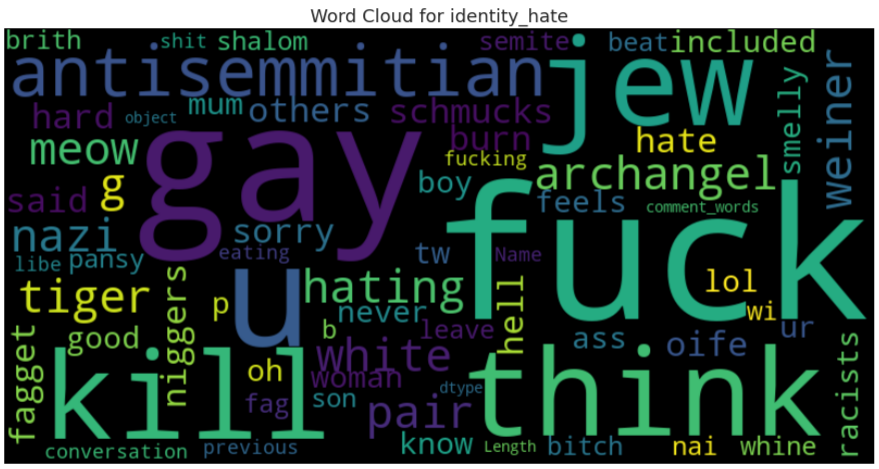


Figure 13: Word cloud for identity hate comments

2.3 Algorithms and Techniques

In what follows we will use two types of approach: classifiers built with machine learning tools, such as Naive Bayes, and deep learning classifiers - with artificial neural networks, such as LSTMs (Long-Short-Term-Memory classifier)[3].

1. Naive Bayes is an algorithm that performs class labels assignment based on a set of features which can be represented numerically with statistical methods. It is based on Bayes theorem with an 'naive' assumption of independence among features, which is, however, typically not the case in real life.
    - In probability theory and statistics, Bayes' theorem (alternatively Bayes' law or Bayes' rule, also written as Bayes's theorem) describes the probability of an event, based on prior knowledge of conditions that might be related to the event [7].
    - There is not one algorithm for training such a classifier, but a series of algorithms based on the same principle: all naive Bayes classifiers assume that each feature of the sample is not related to other features.
2. LSTM is a recurrent neural network (RNN) which is able to learn order dependence in sequence problems. In contrast to the Naive Bayes algorithm discussed above, this type of models does not make an assumption on independence of inputs and outputs, which is a great advantage when dealing with sequential information, such as text.
    - RNN is a type of neural network where the output from previous step are fed as input to the current step. In other words, RNN is a generalization of feed-forward neural network that has an internal memory. RNNs are designed to recognize a data's sequential characteristics and use patterns to predict the next likely scenario [8].
    - RNN's do not perform well on long sequences. If we need to process toxic comments, for instance, RNN's may leave out important information from the beginning. Therefore, there is a need of Long Short Term Memory networks, able to 'remember' the information across the whole comment.

Additionally we will use pretrained word embeddings for text data and SMOTE method to address the problem of unbalanced data.

1. Word embeddings are vector representations of words where words with similar meaning have similar vectors. We will use two types of vector representations:
    - TF-IDF vectorization (term-frequency times inverse document-frequency), this method takes into account not just the occurrence of a word in a single document but in the entire corpus [9]. TfIdf vectorizer will be used to prepare data for Naive Bayes classifier.
    - Pretrained word embeddings such as Glove: global vectors for word representation. It is an unsupervised learning algorithm developed by Stanford for generating word embeddings by aggregating global word-word co-occurrence matrix from a corpus [10]. We will use it to prepare the first embedding layer for the neural network and discuss the implementation in section 3.2.
2. The principle which lies behind SMOTE method involves creating synthetic minority class examples, i.e. for all the toxic comments. Note that SMOTE is only applied to training and validation sets, to avoid overfitting and poor generalizations.

[3]Note that for Naive Bayes classification only training and test sets were used; for LSTMs the training data was split into a proper training set (127656 inputs) and a validation set (31915 inputs).

2.4 Benchmark

Benchmarking is the process of comparing the results to existing method or running a very simple machine learning model [11]. We will implement a simple Naive Bayes classifier adapted for a multilabel classification for this task (cf. section 3.2 for further details). This algorithm is widely and successfully used as a baseline model and it often outperforms much more advanced classifiers. It can be appropriate in the presence of high dimensional features, which is the case of our data set, since text data will be converted into numerical data (cf. section 3.2) [12].

	Precision		Recall		Accuracy	ROC AUC
Multilabel NB [+`neutral`]	neutral	0.91	neutral	0.99	0.8910	0.5047
	others	0.09	others	0.01		

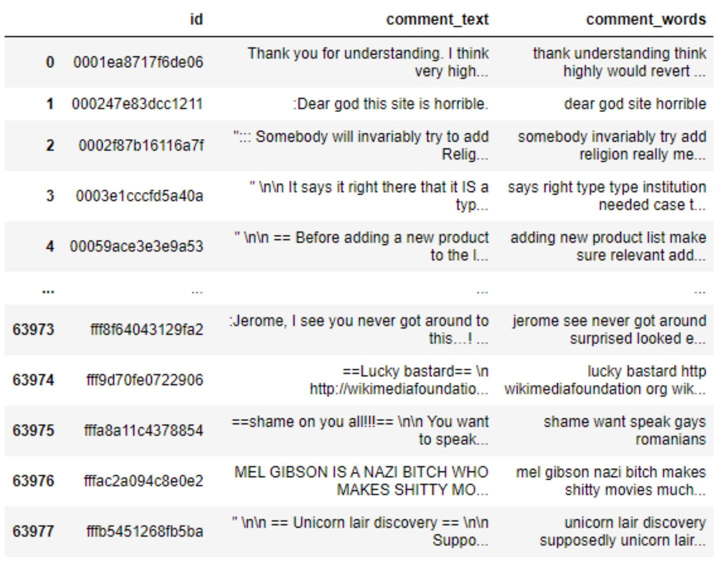
Table 1: Results of multilabel Naive Bayes, with additional `neutral` label

The results of this baseline model, shown in Table 1[4], provide the point from which the skill of all other models trained further can be evaluated. The baseline model performs very well in terms of accuracy and precision/recall metric for neutral labels. However, the ROC AUC score is as good as the pure chance, and precision/recall in classification of toxic comments is very low. To assess the performance of other models we will pay close attention especially to the ROC AUC score.

3 Methodology

3.1 Data Preprocessing

Having analyzed the data in section 2.1 the next step will be to preprocess the raw text data using NLTK tools: we will remove any non alpha-numeric characters (punctuation, end of line characters), lowercase all the strings and remove stopwords. Figure 14 shows the results of this operation.



	id	comment_text	comment_words
0	0001ea8717f6de06	Thank you for understanding. I think very high...	thank understanding think highly would revert ...
1	000247e83dcc1211	:Dear god this site is horrible.	dear god site horrible
2	0002f87b16116a7f	":::: Somebody will invariably try to add Relig...	somebody invariably try add religion really me...
3	0003e1cccfd5a40a	" \n\n It says it right there that it IS a typ...	says right type type institution needed case t...
4	00059ace3e3e9a53	" \n\n == Before adding a new product to the l...	adding new product list make sure relevant add...
...	...	...	...
63973	fff8f64043129fa2	:Jerome, I see you never got around to this...! ...	jerome see never got around surprised looked e...
63974	fff9d70fe0722906	==Lucky bastard== \n http://wikimediafoundatio...	lucky bastard http wikimediafoundation org wik...
63975	fffa8a11c4378854	==shame on you all!!!== \n\n You want to speak...	shame want speak gays romanians
63976	fffac2a094c8e0e2	MEL GIBSON IS A NAZI BITCH WHO MAKES SHITTY MO...	mel gibson nazi bitch makes shitty movies much...
63977	fffb5451268fb5ba	" \n\n == Unicorn lair discovery == \n\n Suppo...	unicorn lair discovery supposedly unicorn lair...

Figure 14: Clean comments after preprocessing

[4]For the sake of space and simplicity of interpretation, in case of multilabel classification, precision and recall for all toxicity labels are grouped in 'other' - with a harmonic mean.

The output of this preprocessing step allowed us to visualize the most frequent words appearing under each label with word clouds.

Nevertheless, this step is not enough, since all the machine and deep learning algorithms require numerical data. The final step of data processing is thus to transform textual information into numbers. As discussed in the previous section, we used two algorithms for this task.

1. `TfidfVectorizer` was implemented with SKLearn package, we chose the `ngram_range` to be (1,1), meaning that we are working with words and not with groups of words; the `max_features` parameter was set to 1000. As the result, the training set is composed out of 159571 entries with 1000 dimensions; test set has the shape (63978, 1000).

2. The implementation of pretrained GloVe embeddings for neural network required some additional preprocessing steps[5]:

   - TensorFlow `Tokenizer` was used to create a vocabulary (word index) for all the words in comments (with `vocab_size` of 1000) and numerically encode comments[6],
   - `pad_sequences` function was performed on encoded comments to achieve the desired length (`maxlen` is 120), i.e. harmonize comments length. This resulted in the following shapes: (127656, 120), (31915, 120) and (63978, 120) for training, validation and test sets respectively.

Since the final numerical feature set was obtained from text data, we will evaluate it as is, without proceeding with feature elimination techniques.

3.2 Implementation

After data preprocessing step (with all relevant implementations discussed in previous section) we proceeded with modeling.

1. The first model, Naive Bayes classifier, can be implemented directly after converting text to numbers. We used SKLearn `MultinomialNB`. To adapt this binary classification model the multilabel classification problem, the following loop was created to concatenate multiple binary classification models into one:

   ```
   clf_nb = []
   for ix in range(7):
       clf_nb.append(MultinomialNB())
       clf_nb[ix].fit(X_train_nb, y_train_nb.iloc[:, ix])
   ```

   The same logic was used for predictions.

2. As for implementation of nural networks, an additional step was required in our workflow: using pretrained embeddings prepare the embedding matrix of weights for Kearas embedding layer [13]:

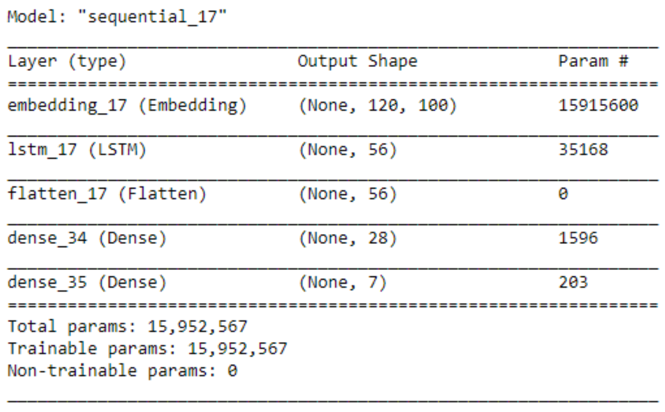
   - compute an embedding index mapping words to known embeddings,
   - leverage this embedded index dictionary on word index to compute embedding matrix.

[5]The implementation of pretrained embeddings does not concern data preprocessing and will be discussed in the next section.

[6]Since we have already turned each comment into vector with `TfidfVectorizer`, there was no particular need to perform comment to integer transformation with TensorFlow. The application of TensorFlow `Tokenizer` is rather motivated by the coherence of using the same framework in building neural networks from the beginning to the end, and to avoid inference between models.

We then implemented a neural network with the following simple architecture[7], using one embedding layer, one LSTM layer and two fully connected lays. Binary crossentropy loss was used to compute the loss function, with Adam optimizer. The model summary is shown in Figure 15:

```
clf_lstm = tf.keras.Sequential([
    tf.keras.layers.Embedding(len(word_index) + 1, embedding_dim,
            weights=[embedding_matrix], input_length=max_length),
    tf.keras.layers.LSTM(56),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(28, activation='relu'),
    tf.keras.layers.Dense(7, activation='softmax')
])
clf_lstm.compile(loss='binary_crossentropy',optimizer='adam',
            metrics=['accuracy'])
clf_lstm.summary()
```



```
Model: "sequential_17"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_17 (Embedding)     (None, 120, 100)          15915600
_________________________________________________________________
lstm_17 (LSTM)               (None, 56)                35168
_________________________________________________________________
flatten_17 (Flatten)         (None, 56)                0
_________________________________________________________________
dense_34 (Dense)             (None, 28)                1596
_________________________________________________________________
dense_35 (Dense)             (None, 7)                 203
=================================================================
Total params: 15,952,567
Trainable params: 15,952,567
Non-trainable params: 0
_________________________________________________________________
```

Figure 15: Neural network architecture

To evaluate the performances of both models the following SKLearnnig tools were used: `classification_report`, `accuracy_score`, `roc_auc_score`.

3.3 Refinement

The refinement step will focus on dealing with unbalanced data. As discussed in subsection 3.2:

- data is highly disproportionate between classes (`neutral` class is dominant - 80% of cases),
- comments which are tagged only with one label constitute 94% of cases.

We suggest exploring three possibilities to improve the results of classification and deal with unbalanced data:

1. Build multilabel classifiers based only on toxicity labels: this implies removing `neutral` label from output list. The data is still not evenly distributed among classes, but the differences are smoother.

[7]We will use the same architecture for different problem statements. Note that two parameters needs to be updated: the shape of the last fully connected layer (7, 6 or 1) and the activation function: `softmax` or `sigmoid`.

2. Since the majority of comments have only one label: group all the toxicity labels under one `toxic` label and build binary models which may tease apart toxic and non-toxic comments. The data is highly imbalanced again: 80% neutral comments against 20% of toxic. Besides, with this problem statement we are not able anymore to perform a fine grained classification of toxic comments.

3. Build binary classifiers using resampling techniques. In this project we will focus on SMOTE, or Synthetic Minority Oversampling Technique.

4 Results

4.1 Model Evaluation and Validation

The results of the for models are displayed in Table 2. Our benchmark model performances are available in the first line, the metrics of all the other models are sorted by decreasing ROC AUC score.

	Precision		Recall		Accuracy	ROC AUC
Multilabel NB [+`neutral`]	neutral	0.91	neutral	0.99	0.8910	0.5047
	others	0.09	others	0.01		
Multilabel NNs [-`neutral`]	neutral	0.10	neutral	1.00	0.0389	0.5001
	others	0.00	others	0.00		
Multilabel NB [-`neutral`]	neutral	0.25	neutral	0.03	0.8938	0.5030
	others	0.06	others	0.01		
Binary NB	toxic	0.25	toxic	0.04	0.8942	0.5146
	neutral	0.91	neutral	0.99		
Binary NB [+SMOTE]	toxic	0.13	toxic	0.24	0.7743	0.5345
	neutral	0.91	neutral	0.83		
Multilabel NNs [+`neutral`]	neutral	0.96	neutral	0.96	0.8745	0.5817
	others	0.11	others	0.10		
Binary NNs [+SMOTE]	toxic	0.19	toxic	0.85	0.6395	0.7338
	neutral	0.97	neutral	0.62		
Binary NNs	toxic	0.56	toxic	0.70	0.9180	0.8210
	neutral	0.97	neutral	0.94		

Table 2: Results of classification models

The first consideration we can make is that all Naive Bayes models do not change a lot for different problem statements, comparing to the benchmark (the only exception is binary Naives Bayes classifier with SMOTE, which lost more than 10% in accuracy, however, when the oversampling technique is applied, ROC AUC score in slightly above the baseline).

The next general consideration concerns the overall precision and recall tendencies: both of these metrics are significantly higher for classification of neutral comments comparing to the toxic ones (for both multilabel and binary classifiers). This may be explained by disproportionate fraction of neutral vs. toxic comments in data.

Now we are going to analyse closely the results of neural networks which results vary a lot for different problem statements.

First, the NNs without `neutral` label performances are slightly below baseline. Comparing to the performances of Naive Bayes classifier for the same problem statement, which are also below baseline, we can conclude that at this stage our models are not very strong in prediction of only toxic labels for the test data.

Second, adding `neutral` label to outputs allows to the NNs to gain 8% to the ROC AUC score. However, the best results at this stage are observed for binary NNs classifiers.

Binary NNs with SMOTE is at 73% of ROC AUC, wich is a significant improvement comparing to the baseline model (however, the accuracy drops by more than 20%).

Binary NNs, without SMOTE has the highest performances, in terms of both ROC AUC (82%) and accuracy (92%).

The results of our models performances suggest that at this stage of investigation the best choice would be to choice binary classification models (with neural networks), at the cost of losing the fine-grained classification of toxic comments.

4.2 Justification

In this project, initially designed for building a multilabel classifier, we explored different possibilities to approach the classification problem in question, built two types of models and presented multiple solutions.

We chose a simple Naive Bayes model as a benchmark model, and it has shown good performances in terms of accuracy (89%); however, all the other models had a room for improvement in terms of ROC AUC score (50%).

At this stage of our investigation, if we want to build a multilabel classifier, the best model to achieve such a classification is multilabel neural network which can distinguish not only between toxic and non-toxic comments, but also between different degrees of toxicity. However, the performances of this model are poor (while the accuracy is quite high, ROC AUC score is just 58%).

We demonstrated, that approaching this classification as a binary problem may give a better results. Binary neural networks have great performance distinguishing toxic from non toxic comments (both for accuracy and ROC AUC score).

If we want to stay with initial problem of multilabel classification, some improvements to the models may be investigated [14]:

- Change the algorithm (decision trees and random forest can generally perform better when data is unbalanced) and/or use more complex neural network architecture (i.e. biderectional LSTMs, GRUs, CapsNets) and/or explore different pretrained embeddings (VordToVec, FastText);
- Evaluate the dimensions of the inputs after converting textual data into numerical and/or perform feature correlation analysis and/or use dimensionality reduction techniques (e.g. PCA);
- Perform distinct techniques to address unbalanced data problem, such as oversampling minority class or undersampling majority class.

Further research may give better results, official Kaggle competition page also provides a lot of inspiring solutions.

5 References

[1] Social media addiction linked to cyberbullying:
https://news.uga.edu/social-media-addiction-linked-to-cyberbullying/

[2] A Majority of Teens Have Experienced Some Form of Cyberbullying:
https://www.pewresearch.org/internet/2018/09/27/a-majority-of-teens-have-experienced-some-form-of-cyberbullying/#fnref-21353-1

[3] 2016 Cyberbulling data:
https://cyberbullying.org/2016-cyberbullying-data

[4] About 1 in 5 victims of online harassment say it happened in the comments section:
https://www.pewresearch.org/fact-tank/2014/11/20/about-1-in-5-victims-of-online-harassment-say-it-happened-in-the-comments-section/

[5] Toxic Comment Classification Challenge:
https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge

[6] Choosing the Right Metric for Evaluating Machine Learning Models — Part 1:
https://medium.com/usf-msds/choosing-the-right-metric-for-machine-learning-models-part-1-a99d7d7414e4

[7] Introduction to Naive Bayes for Classification:
https://medium.com/@srishtisawla/introduction-to-naive-bayes-for-classification-baefefb43a2d

[8] Long Short Term Memory (LSTM) Networks in a nutshell:
https://ahmetozlu93.medium.com/long-short-term-memory-lstm-networks-in-a-nutshell-363cd470ccac

[9] Word embedding:
https://medium.com/data-science-group-iitr/word-embedding-2d05d270b285

[10] What is GloVe?:
https://medium.com/analytics-vidhya/word-vectorization-using-glove-76919685ee0b

[11] How To Know if Your Machine Learning Model Has Good Performance:
https://machinelearningmastery.com/how-to-know-if-your-machine-learning-model-has-good-performance/

[12] Keep it Simple, Stupid — The Naive Bayes Classifier:
https://medium.com/codex/keep-it-simple-stupid-the-naive-bayes-classifier-fa7d8832eb1a

[13] Using pre-trained word embeddings in a Keras model:
https://blog.keras.io/using-pre-trained-word-embeddings-in-a-keras-model.html

[14] Dealing with Imbalanced Data:
https://towardsdatascience.com/methods-for-dealing-with-imbalanced-data-5b761be45a18