TOXICITY OF WIKIPEDIA COMMENTS - MILESTONE REPORT

-Disclaimer: the dataset for this project contains text that may be considered profane, vulgar, or offensive.

Introduction:

Wikipedia is a multilingual online encyclopedia based on open collaboration through a model of content edited by web-based applications like web browsers. It is the largest and most popular general reference work on the World Wide Web.

Open collaboration has its own difficulties since some authors insert contents that are toxic, which means they have improper language or are reflecting hatred towards a group of people etc. Moreover, the software that powers Wikipedia provides certain tools allowing anyone to review changes made by others, drawing criticism from opposing groups, which in some cases lead to vulgar language and vandalism.

Editors can enforce these rules by deleting or modifying non-compliant material. However, it is more practical to have computer programs to perform anti-toxic duties, blocking or deleting entries of such properties.

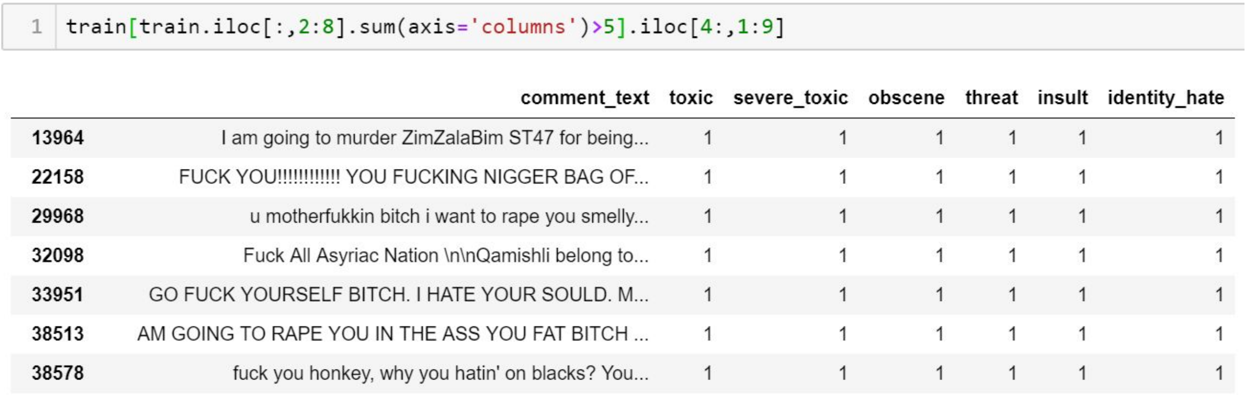
A model, utilizing neural networks (NN) , preferably Recursive NN, that’s capable of detecting different types of of toxicity like threats, obscenity, insults, and identity-based hate will be built and used to block improper comments.

Dataset:

Dataset consists of comments entered by authors word wide about various topics. In this project the aim is to build a multi-headed model that’s capable of detecting different types of toxicity like threats, obscenity, insults, and identity-based hate. A successful model will hopefully help online discussion become more productive and respectful.

The number of observations in the dataset are more than 150 thousand, each including a document under column name ‘comment_text’, length of which ranges from just a couple of words to a few paragraph entry.

The nature of the label set, consisting of 6 columns brings challenge to this dataset because a comment may be both toxic and obscene, it may have elements of threat and identity hate - and even in some cases all six flags may be raised.



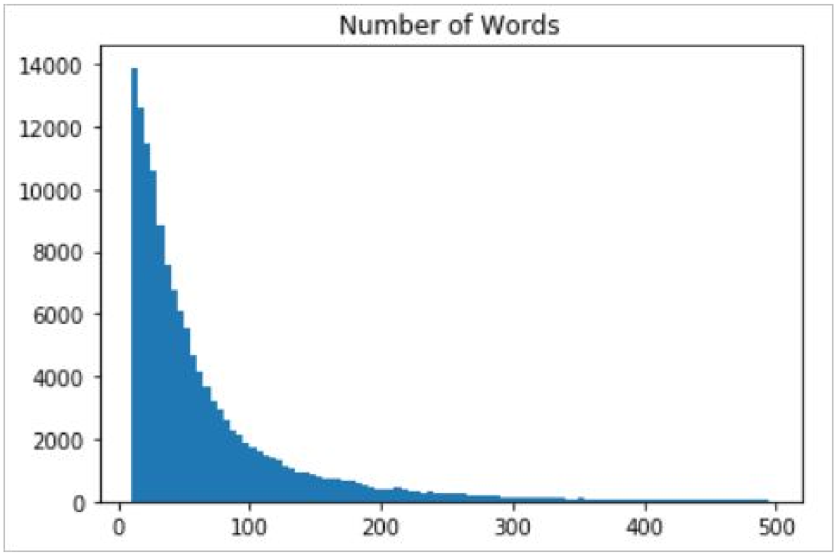
```
train[train.iloc[:,2:8].sum(axis='columns')>5].iloc[4:,1:9]
```

	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
13964	I am going to murder ZimZalaBim ST47 for being...	1	1	1	1	1	1
22158	FUCK YOU!!!!!!!!!!!! YOU FUCKING NIGGER BAG OF...	1	1	1	1	1	1
29968	u motherfukkin bitch i want to rape you smelly...	1	1	1	1	1	1
32098	Fuck All Asyriac Nation \n\nQamishli belong to...	1	1	1	1	1	1
33951	GO FUCK YOURSELF BITCH. I HATE YOUR SOULD. M...	1	1	1	1	1	1
38513	AM GOING TO RAPE YOU IN THE ASS YOU FAT BITCH ...	1	1	1	1	1	1
38578	fuck you honkey, why you hatin' on blacks? You...	1	1	1	1	1	1

This is a classification problem, but it comes with its slight variations since a comment can be labelled with more than one toxicity flag. Any specific comment may be represented with more than one label.

The aforementioned problem can be solved with different approaches, depending on the model used to solve the problem. For a baseline Multinomial Naive Bayes, model has to be run for each class label. However, on the other hand, with utilization of a neural network, output layer nodes can be adjusted to represent all class labels at once. Both methods were used for this study.

Data Exploration:

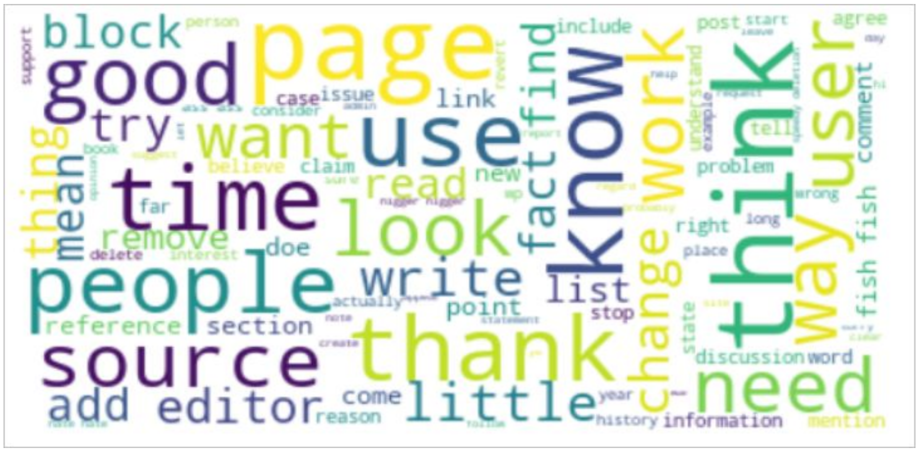


The 159’571 documents were joined into a corpus for analysis. The number of words in the corpus were 7’153’449 and number of unique words were 210’337.

Distribution of the number of words per document is represented by histogram on the left.

Most common 50 words, after text preprocessing were:
[('article', 8850), ('page', 7312), ('wikipedia', 5399), ('edit', 4893), ('talk', 4702), ('use', 4317), ('like', 3591), ('think', 3022), ('know', 2995), ('source', 2930), ('good', 2810), ('time', 2404), ('add', 2316), ('people', 2236), ('user', 2234), ('want', 1937), ('block', 1911), ('need', 1891), ('image', 1823), ('find', 1746), ('delete', 1746), ('link', 1728), ('look', 1715), ('work', 1714), ('remove', 1711), ('thank', 1627), ('information', 1602), ('write', 1582), ('fuck', 1541), ('change', 1512), ('way', 1504), ('little', 1489), ('comment', 1481), ('editor', 1457), ('thing', 1446), ('section', 1441), ('list', 1415), ('point', 1409), ('deletion', 1406), ('fact', 1384), ('try', 1378), ('help', 1377), ('thanks', 1375), ('read', 1360), ('doe', 1342), ('question', 1336), ('new', 1317), ('mean', 1316), ('wp', 1312), ('right', 1253)]

The unprocessed corpus, once input to a frequency based word cloud, generated the words represented by the graph on the right. Note that only the stop words were removed for the word cloud.



Data Cleaning / Text Pre-processing:

Text pre-processing is a very important stage of data cleaning and the methods chosen, their order and detail level has significant impact on the quality of the classifier. Since the corpus consists of more than 7 millions of words, pre-processing is time consuming and sometimes computationally expensive.

First of all, URL addresses were removed by Python built-in library Regex. Then accented characters were normalized into ASCII characters, which really comes into play when dealing with non-English alphabets. This also prevents some of the flagged words to go unnoticed, since sometimes collaborators try to cloak swear words by making minor alterations, such as changing an “e” to “é”.

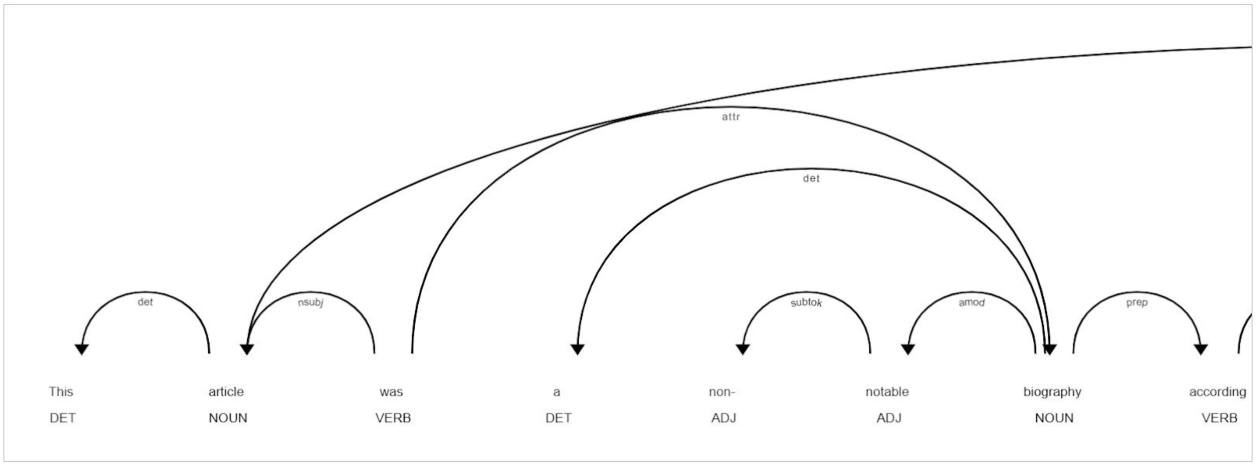
Next, contractions are expanded via a dictionary. Contractions are shortened versions of words or syllables. These shortened versions of existing words or phrases are created by removing specific letters and sounds. Examples would be, do not to don’t and I would to I’d. Converting each contraction to its expanded, original form often helps with text standardization.

Special characters including non-English alphabet and symbols were regarded as noise and removed with regular expressions code.

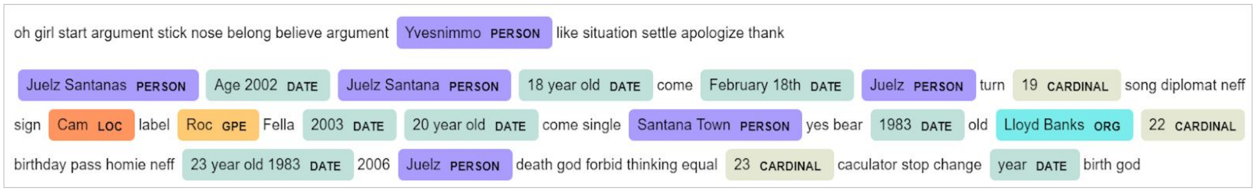
Next step in text preprocessing was lemmatization, where we remove word affixes to get to the base form of a word, which is always lexicographically correct. Lemmatization is computationally cumbersome compared to stemming, but it was preferred over stemming since it is more precise.

Later stop words were removed together with extra white spaces and text was converted to lowercase.

Spacy, a free open-source library for Natural Language Processing, was used in this project since it is one of the most sophisticated libraries and has great documentation. Spacy features NER, POS tagging, dependency parsing, word vectors.



Two pre trained statistical English language models were used. The first model was ‘en_core_web_lg’, which supports NER and POS features, the other one was 'en_vectors_web_lg'. The latter model does not support NER or POS, but has 1070971 unique vectors (300 dimensions) trained with GloVe.



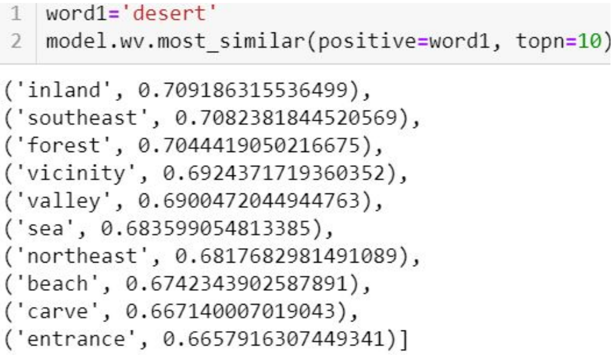
oh girl start argument stick nose belong believe argument Yvesnimmo PERSON like situation settle apologize thank

Juelz Santanas PERSON Age 2002 DATE Juelz Santana PERSON 18 year old DATE come February 18th DATE Juelz PERSON turn 19 CARDINAL song diplomat neff sign Cam LOC label Roc GPE Fella 2003 DATE 20 year old DATE come single Santana Town PERSON yes bear 1983 DATE old Lloyd Banks ORG 22 CARDINAL birthday pass homie neff 23 year old 1983 DATE 2006 Juelz PERSON death god forbid thinking equal 23 CARDINAL caculator stop change year DATE birth god

Non English comments:

This is an area of improvement since there are many comments that are not English. There are some libraries that can detect non-English words, however, this problem was not addressed at this time and left as provisional room for improvement. This also warrants a revisit to the filtering stage if the language uses a non-Latin alphabet, since such letters are filtered out.

Similarities:

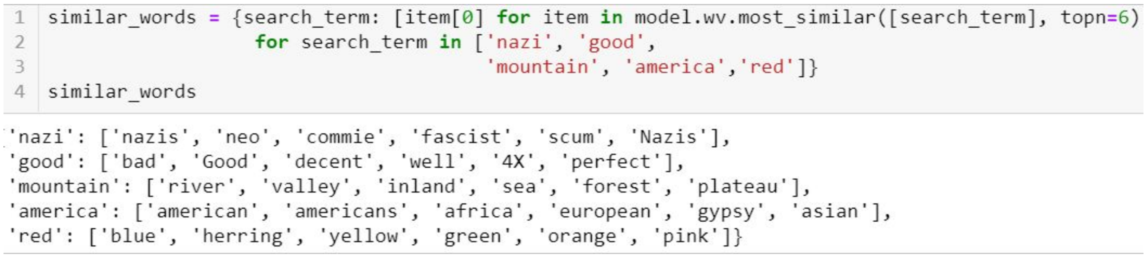


```
word1='desert'
model.wv.most_similar(positive=word1, topn=10)
```

```
('inland', 0.709186315536499),
('southeast', 0.7082381844520569),
('forest', 0.7044419050216675),
('vicinity', 0.6924371719360352),
('valley', 0.6900472044944763),
('sea', 0.683599054813385),
('northeast', 0.6817682981491089),
('beach', 0.6742343902587891),
('carve', 0.667140007019043),
('entrance', 0.6657916307449341)]
```

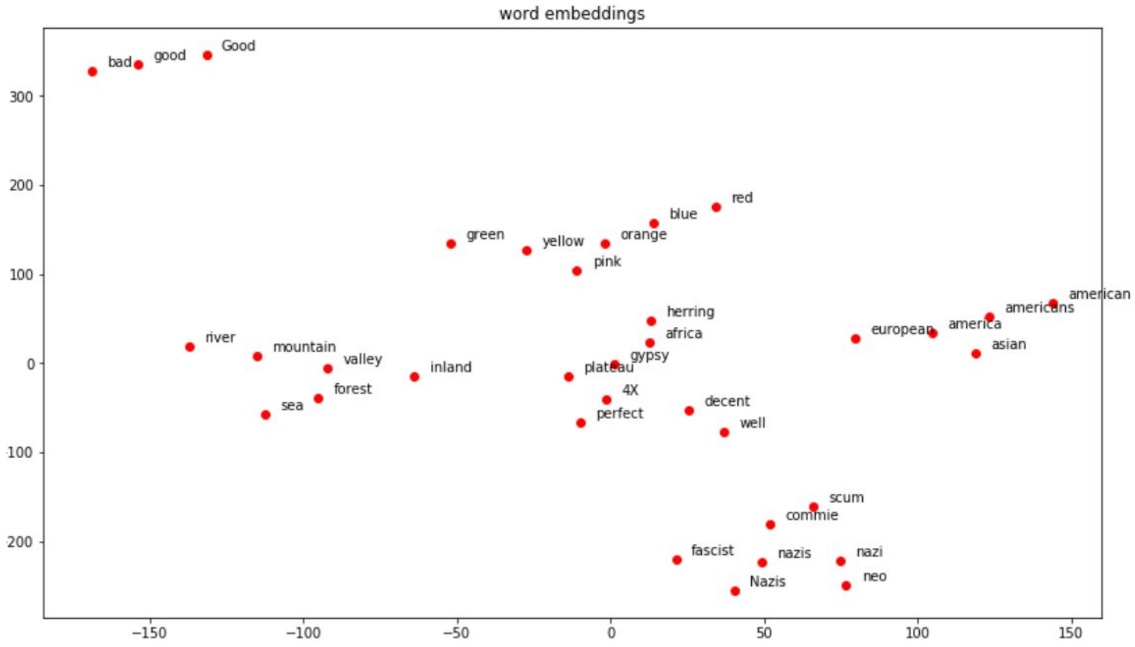
The pre-trained vectors of spacy did not work well for similarities since language of comment texts is not daily used language. After training own word2Vec vectors with Gensim, the similarity results were quite good. On the most similar words to selected word can be seen. For visualization of similarities, dimension reduction is required.

T-distributed Stochastic Neighbor Embedding (t-SNE) algorithm for visualization was used for embedding high-dimensional data for visualization in a low-dimensional space of two dimensions. t-SNE models each high-dimensional object by a two- or three-dimensional point in such a way that similar objects are modeled by nearby points and dissimilar objects are modeled by distant points with high probability.



```
similar_words = {search_term: [item[0] for item in model.wv.most_similar([search_term], topn=6)
                  for search_term in ['nazi', 'good',
                                      'mountain', 'america','red']}
similar_words
```

```
'nazi': ['nazis', 'neo', 'commie', 'fascist', 'scum', 'Nazis'],
'good': ['bad', 'Good', 'decent', 'well', '4X', 'perfect'],
'mountain': ['river', 'valley', 'inland', 'sea', 'forest', 'plateau'],
'america': ['american', 'americans', 'africa', 'european', 'gypsy', 'asian'],
'red': ['blue', 'herring', 'yellow', 'green', 'orange', 'pink']}
```


word embeddings
bad
good
Good
red
blue
green
yellow
orange
pink
herring
africa
american
americans
america
european
asian
river
mountain
valley
inland
plateau
gypsy
forest
sea
4X
perfect
decent
well
scum
commie
fascist
nazis
nazi
Nazis
neo