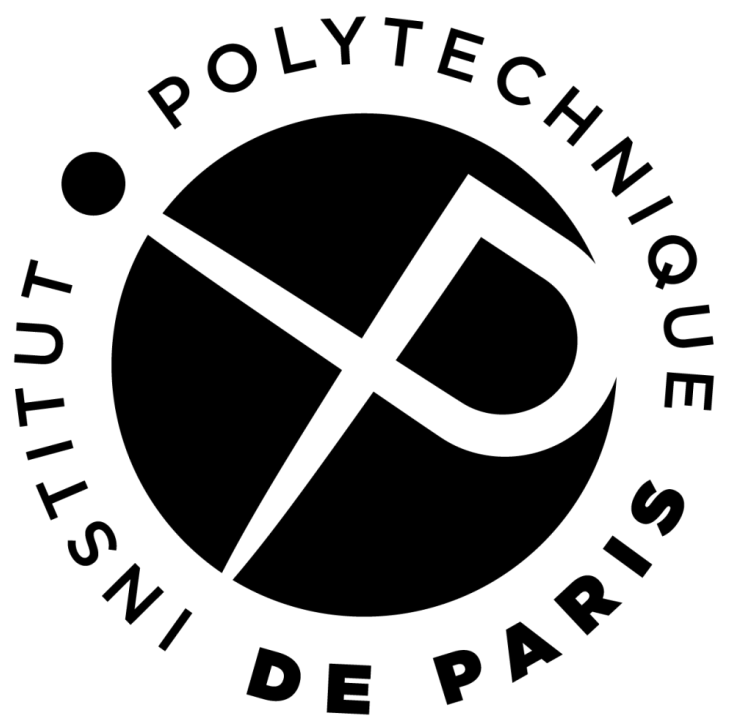


Machine Learning Approach to Rating Relative Severity of Toxic Comments

Contributors: Chenxi Deng, Guanlin Li[1]

[1] The two authors have equal contributions on this project.

Table of content

1. Abstract

The rapid growth of online platforms and forums has brought an influx of online toxic comments. This phenomenon not only exposes billions of internet users to the risk of harassment, but triggers more toxic behaviours. Early detection and removal of hate speech is therefore essential to mitigate adverse consequences of this antisocial behaviour. Our project is motivated by this notion. In this report, we strategise three approaches (regression, ranking and classification methods) using linear models. We aim to develop a model that accurately ranks the comments based on their relative severity. We find the best performing model is Universal sentence Encoder + Support Vector Regression.

2. Introduction

2.1. Overview

Online social media platforms connect billions, allowing their users to share their thoughts and opinions with one another, often anonymously and thus with little fear of repercussions[2][3]. As a consequence, there has been an explosion of abusive language and hate speech over the internet over the past decade[2]. This phenomenon exposes billions of people to various forms of online harassment and provocation[4], compelling technology companies to recruit armies of comment moderators. However, human moderation can be heavily influenced by their personal biases. Machine learning classification methods offer attractive alternatives that provide large degrees of automation and reduce biases in individual judgements.

This project uses datasets from past and active Jigsaw Kaggle competitions. The training dataset consists of a large set of comments annotated with one toxicity label indicating whether the comment is toxic or not, and five additional toxicity attributes. The test dataset consists of pairs of comments, one labeled “more toxic” and the other labeled “less toxic”, which were annotated by human moderators. Detailed description can be found in the Data Preparation section.

2.2 Task and Evaluation

The objective of this project is to build a model that effectively ranks comments on the basis of their relative toxicity. The performance of our model is tested on how well our prediction matches the average evaluation of human moderators. That means, for every pair of comments i, we get a score $s_i = 1$ if our prediction on relative toxicity matches that of the annotators, and $s_i = 0$ if it does not. Our key criterion for evaluating our model is the mean score computed from all comment predictions:

---

[2] J, Ashok Kumar, S, Abirami, Trueman TE & Cambria E (2021) Comment toxicity detection via a multichannel convolutional bidirectional gated recurrent unit. *Neurocomputing* 441: 272–278

[3] Duggan M.(2017). Online harassment.

[4] Cheng J, Bernstein M, Danescu-Niculescu-Mizil C & Leskovec J (2017) Anyone Can Become a Troll: Causes of Trolling Behavior in Online Discussions. CSCW Conf Comput Support Coop Work 2017: 1217–1230

$$score = \frac{\sum_{i=1}^{n} s_i}{n}$$

We intend to explore several methods:

1) A regression model that uses data labeled with toxicity levels to predict a toxicity level for a new comment.
2) A pairwise ranking method that views the toxicity of the text as “query” and performs pairwise learning.
3) A classification method that predicts whether a comment falls into the “more toxic” or “less toxic” class.

1.3 Related work

Many techniques for comment classification have been proposed over the past few years. These techniques can be split into two broad categories, classical methods (refer to the ones that require feature engineering) and deep learning methods[5]. Classical methods are the ones that require feature engineering, which often demands enormous efforts and has its shortcomings. Dictionary-based methods, for example, often fail to capture the different forms of hate[6]. As an instance, “I hate scientists” would be classified as toxic but not “scientists are pigs”. To address this problem, Saleem *et al*[7]. developed a labeled Latent Dirichlet Allocation (LLDA) approach that aims to learn and infer topics of texts for the purpose of classifying them. This approach showed better performance than traditional keyword-based classifiers.

On the other hand, the bag-of-words (BOW) methods have been shown to obtain accurate results in various circumstances. Using bag-of-words feature representation techniques along with a Naïve Bayes classifier, Kwok, I. and Y. Wang[8] achieved an F-score of .76. Djuric et al.[9] used Paragraph2vec with a BOW neural network model that was capable of discovering masked insults and swearing.

Recent trends in hate speech detection deploy deep learning methods. Badjatiya P, et al.[10] revealed that the application of deep neural networks, combined with gradient boosted decision trees gave better performances than most traditional machine

---

[5] J, Ashok Kumar, S, Abirami, Trueman TE & Cambria E (2021) Comment toxicity detection via a multichannel convolutional bidirectional gated recurrent unit. *Neurocomputing* 441: 272–278

[6] Mondal M, Silva LA and Benevenuto F (2017) A Measurement Study of Hate Speech in Social Media. In: Proceedings of the 28th ACM Conference on Hypertext and Social Media, HT ’17, New York, NY, USA: ACM, pp. 85–94.

[7] Saleem HM, Dillon KP, Benesch S & Ruths D (2017) A Web of Hate: Tackling Hateful Speech in Online Social Spaces. *arXiv [csCL]*

[8] Kwok I & Wang Y (2013) Locate the hate: detecting tweets against blacks. In Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence pp 1621–1622. AAAI Press

[9] Djuric N, Zhou J, Morris R, et al. (2015) Hate Speech Detection with Comment Embeddings. In: Proceedings of the 24th Interna- tional Conference on World Wide Web, WWW ’15 Companion, New York, NY, USA: ACM, pp. 29–30.

[10] Badjatiya P, Gupta S, Gupta M, et al. (2017) Deep Learning for Hate Speech Detection in Tweets. In: Proceedings of the 26th International Conference on World Wide Web Companion, WWW ’17 Companion, Republic and Canton of Geneva, Swit- zerland: International World Wide Web Conferences Steering Committee, pp. 759–760.

learning classifiers (LR and SVM, particularly). Other notable studies by Park JH, Fung P[11] and Zhang et al.[12] also experimented with deep neural networks and reached similar conclusions.

3. Data Preparation

3.1 Data sets

To train our models, we used datasets from two past Kaggle competitions including "Jigsaw Unintended Bias in Toxicity Classification[13]", and "Toxic Comment Classification[14]". The concatenated dataset contains a total of 2,223,065 user comments, each with an unique identifier and a toxicity label "toxic" that distinguishes the toxic comments from the innocent ones. Additionally, each sample is annotated with one or more subtypes of toxicity: "severe toxic", "obscene", "threat", "insult", "identity hate". Furthermore, the original dataset contains texts of comments that had been pro-processed. However, the cleaned comments are not used in our experiment. We applied alternative methods for cleaning this dataset.

The test dataset was obtained from an active Kaggle competition labeled "Jigsaw Rate Severity of Toxic Comments[15]". The dataset consists of pairs of comments, one of which is supposedly more toxic than the other. These ratings were determined by the average rankings given by one or more moderators. This dataset will be used to examine how well our model performs.

3.2 Data Exploration

The characteristics of the data used for this study is analyzed in this section. This analysis gave us insights about the distribution of the data, the most frequent tokens.

The training dataset contains over 2 million comments, each was annotated with six different toxicity types, including toxic, severe toxic, obscene, threat, insult, identity hate; a possibility score was assigned to the comments for each toxicity category. To visualise the distribution, the total number of comments falling into each of the six toxicity types was computed, respectively. The comments labeled as "toxic" are the most prevalent in our datasets, which is expected, as the comments falling into any of the toxicity subtype attributes ("severe toxic", "obscene", "threat", "insult", "identity hate") must also be classified as "toxic"(Fig 1a).

3.2.1 Data distribution

We also checked if there are any samples that do not belong to any toxic categories. The sum of all values per row was computed. The percentage distribution of non-toxic vs. toxic comments for all comments in the training dataset was plotted. The resulting graph clearly suggests that the distribution of our training dataset is

[11] Park JH and Fung P (2017) One-step and Two-step Classification for Abusive Language Detection on Twitter. arXiv preprint arXiv:1706.01206.
[12] Zhang Z et al (2018) Detecting hate speech on twitter using a convolution-gru based deep neural network. Eur Semant Web Conf 2018:745–760
[13] https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/data
[14] https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/data
[15] https://www.kaggle.com/c/jigsaw-toxic-severity-rating/data

skewed, with the fraction of non-toxic comments more than twice of the toxic comment percentage. This distribution imbalance can impose challenges for predicting the minority class (toxic class, in this case) in later experiments (Fig 1b, Appendix 1).

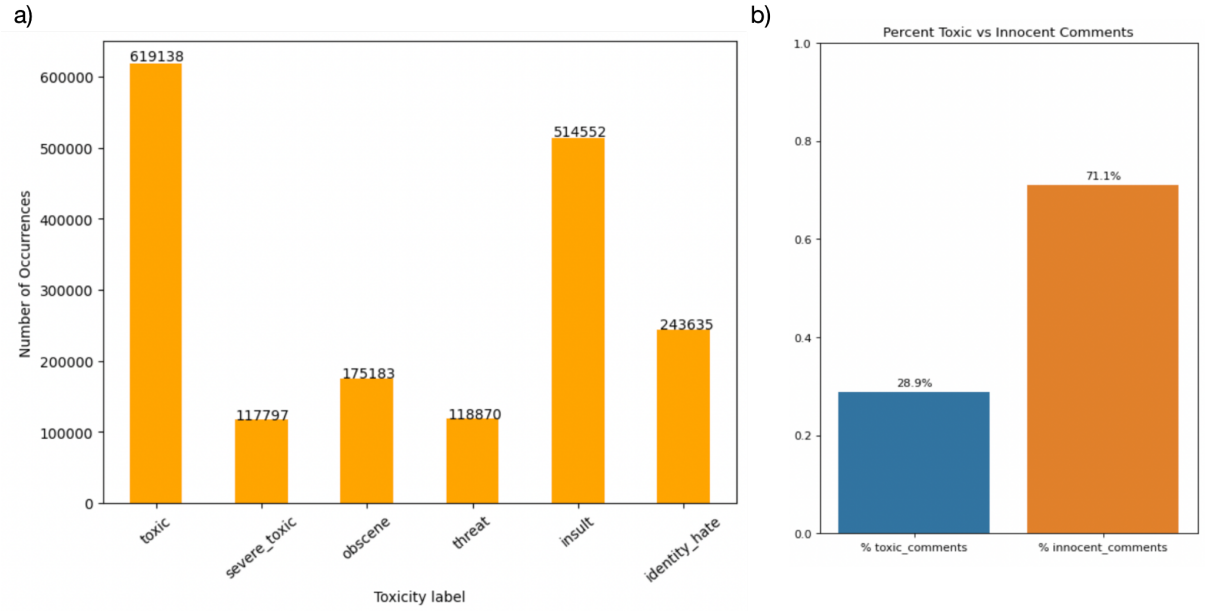


Figure 1. a) The distribution of comments in each toxic class. The most prevalent class is "toxic", followed by "insult'.
b) The percentages of toxic vs innocent comments in training dataset. The distribution appears skewed.

On the other hand, the test dataset contains 30,108 comment pairs and two columns ("less toxic" and "more toxic") stating which comment is more toxic than the other. The distribution of "less toxic" and "more toxic" comments is well-balanced (according to Kaggle data description).

3.2.2 N-gram Feature Visualisation

N-grams are simply defined as contiguous sequences of n words. We can generate n-grams for a given text and compute the frequency of the occurrence of the n-grams. In this experiment, n values of 1 and 2 are used to produce unigrams, which allows us to look into the occurrence of each word independent of its surrounding texts, and bigrams that provide us with insights into the context of the words used.

Prior to producing n-grams, the comments were split into a list of iterable tokens, which can be characters, digits, punctuations, etc. To facilitate n-gram feature visualisation, we removed punctuations and stopwords (common words that occur at a high frequency but are not particularly meaningful, such words as "the", "a') from the tokens. The resulting unigrams and bigrams generally support the overall sentiment of the sample comments. For example, the most frequent unigram and bigram appearing in "more toxic" comments in the test dataset were "fuck" and "fuck fuck" and the common terms in "less toxic" comments were "wikipedia", "shit shit" (Appendix 2).

3.2.3 Latent Dirichlet Allocation (LDA)

To further explore the data, a topic model was run using Latent Dirichlet Allocation (LDA) method. LDA is a generative statistical model that allows a fixed number of topics to be represented as a distribution of words. The text data from both training and test datasets was transformed into vectors according to the frequency of each word that occurs in the text. The LDA was run with a k value of 10 (topics) for unique comments from train data and test data, respectively. LDA successfully extracted unique topics in both datasets (Appendix 3)

3.3 Data Pre-processing

The idea of data pre-processing is to clean the data so that texts can be easily tokenised later. This includes getting rid of unwanted samples, converting variations of words to their original forms and removing words that contain little useful information. We achieved this by using regular expressions to replace strings with empty strings or alternative forms of the strings. After cleaning, we adopt additional transformation methods to prepare data for later model building, such as dividing the original continuous level into discrete level range, tokenizing texts etc.

4. Approach

We propose three strategies to score the texts according to their toxic severity: regression methods (Linear model - Ridge Regression, Support Vector Regression (SVR) and Random Forest Regression), a ranking method (SVM rank) and classification methods (Linear model - Logistic Regression, Support Vector Classifier (SVC) and Random Forest Classifier)

4.1 Regression Method

The goal of the task is to determine, give a pair of comments, which text between the two is the more toxic one. Given the dataset we collected, we have 6 additional toxicity subtype attributes related to the toxicity level, apart from the field toxicity. The field toxicity describes the overall toxicity level of the data, meaning that if a data sample has 0 toxicity, the other 6 subfields should be 0; but if two samples have the same toxicity, the other 6 subfields attributed would still be different. They, combined together, reflect the overall toxicity level of the sample. So, We acquire a new field by summing all the values from the fields that are related to the toxicity level. When summing up, it's possible to adjust the weights to reflect the importance of a certain field, eg. The field, severe toxicity, may reflect more severity, so we can give it extra weights when summing.

After we get the new field toxicity level, which has a real value range $r \in [0, 6]$, we use it as a regression target. Given a sentence, our goal is to learn a real value number that actually reflects the toxicity level of the sentence. Then, this value can be served to determine which sentence is more toxic between two given sentences.

3.1.1 Embedding Methods

We transform texts into vectors with three different word-embedding methods: the TF-IDF method, word2vec method and pre-trained language model encoding methods.

TF-IDF Method TF-IDF views the sentence as a bag of words, and uses TF-IDF value of the words in the sentence to construct a multi-hot vector. We calculate the TF-IDF value of a word using the following formula, where t is the target word, d is the sentence containing the target word and D is the set of all sentences:

$$TFIDF(t,\ d)\ =\ TF(t,d)\ *\ IDF(t)$$

$$TF\ =\ log(1\ +\ freq(t,d)$$

$$IDF\ =\ log(\frac{n}{(d \subseteq D\ where\ t \subseteq d)})$$

With this method, we get vector v_d for each sentence that:

$$v_d = (e_1, e_2, ..., e_n),\ v_d \subseteq R^n$$

$$e_{i,d}\ =\ TFIDF(i,\ d)\ if\ i\ in\ d\ else\ 0$$

Where n is the size of the vocabulary. The embedding acquired by this method can reflect the statistical importance of the words in the corpus, but the vector is sparse: for every sentence the vector can have more than several ten thousand dimensions, with most of them being 0. Therefore, we perform word2vec (Mikolov #) to obtain a low dimension dense representation of the sentence.

Word2vec Method The preliminary exploration of our dataset revealed that the domain we are dealing with has a large deviation from the corpus that most of the pre-trained vectors are trained based on. Therefore, we decide to train word2vec on our dataset, instead of using pre-trained word2vec vectors for the embedding. After getting the word2vec model in which it stores the embedding of all the words, we use the average pooling strategy to get the vector of the sentence:

$$v_d = \frac{\sum_{i=0}^{n} w_i}{n}$$

Where n is the total number of words in the sentence and w_i is the word embedding of word i which belongs to sentence d.

Pre-trained Language Model Encoding Method With above methods we view sentences as bags of words, and train embeddings on our own corpus. While two million is a non-trivial number, it's actually not that big compared to the corpus used when training massive deep language models like BERT[16]. Besides, using a bag of words ignores the information contained in the order among words in the sentence. Therefore, we choose to apply pre-trained language models that can provide embedding for the whole sentence directly. Here, we deploy Google universal sentence encoder[17] to transform sentences. The universal sentence encoder uses

[16] Devlin J, Chang M-W, Lee K & Toutanova K (2018) BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv [csCL]*

[17] Yang Y, Cer D, Ahmad A, Guo M, Law J, Constant N, Abrego GH, Yuan S, Tar C, Sung Y-H, et al (2019) Multilingual Universal Sentence Encoder for Semantic Retrieval. arXiv [csCL]

the transformer which is trained on large corpus to provide meaningful sentence-level embedding for several downstream tasks. After pre-training, the encoder of the transformer model is used to provide encodings.

4.1.2 Regression Models

Three different types of regression models are selected to perform our tasks: Linear model, SVM and tree method.

Linear Regression Firstly, a least squares linear regression model is used. Given the embedding of sentence and corresponding target, the model tries to find a function with coefficients $w = (w1, \ldots, wp)$ to minimize the residual sum of squares between the prediction and the given value.

SVM Regression. Generally speaking, svm methods are robust and have a stable performance on a large set of machine learning tasks. We choose the regression method derived from the standard SVM classification method, which is Support Vector Regression, to perform the regression task.

Random Forest Regression. The most commonly used tree method is the decision tree model. However, the decision tree method tends to overfit the data and does not perform as well as its ensemble counterparts. Therefore, we use random forest - the ensemble method built on top of the decision tree to perform the task. The random forest has a good capability to fit a large amount of data, and generally yield good results.

4.2 Ranking Method

The goal of the project is to mine the relative toxicity between two sentences. While this can be achieved by considering all the sentences as a list and giving a score directly to each element, it's very natural to just view the problem as a pair-wise ranking problem in information retrieval. The question can be seen, given a set of documents $D = \{d_1, d_2, \ldots, d_n\}$, we should build a function f under the query q so that it returns an ordering r^* such r^* is as close to the ordering given by the relative toxicity as possible. We focus on the toxicity relativity between two sentences, that is, given two sentences d_i and d_j, if the relative toxic level of d_j > d_i,we want:

$$(d_i, d_j) \in f\overline{w}(q) \Leftrightarrow \overline{w}\phi(q, d_i) > \overline{w}\phi(q, d_j)$$

In our context, the query q is the same for the document set D. ϕ is the feature of sentence under the given query q and w is the parameter for the model. The question can be seen as a binary classification problem where the model learns w so that number of pairs satisfying above condition is the maximum in D. We use pairwise SVM rank[18] to achieve this.

4.3 Classification Method

Since the task is not to accurately predict the value of toxicity level, but to compare two sentences according to their toxicity level, we can view this problem as a

[18] https://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html

classification problem to better utilize the information that the model outputs. The idea is to manually divide the continuous value of toxicity level into discrete ordinal categories of toxicity levels, and to predict which toxicity level the sentence falls into. To do this, we first normalize the continuous toxicity level to range [0, 1], and we make a category of 11 classes with a step 0.1, namingly:

$$c = [0,\ 0.1,\ 0.2,\ 0.3,\ ...,\ 0.9,\ 1].$$

Then, we map the value of toxicity level to the responding class. When doing comparison between sentences, we just need to compare their toxicity level instead of comparing the real value score. The advantage of this method is that we can make full use of the model's output: we can utilize the information not only about the class with the highest prediction score, but also we can take advantage of the probability that model gives to other classes. The probability can serve as a weight of the prediction, and we get the toxicity level by the following formula:

$$T = \sum_{i=0}^{n=10} p_i * c_i$$

Where $c_i \in c$, p_i is the corresponding probability that model predicts for c_i.

3.3.1 Embedding methods

For the feature extraction from the texts, we use the same methods as described in 3.1.1.

3.3.2 Classification Model

Logistic Regression. For comparison, we choose logistic regression as the linear classification model. The same as linear regression, it also learns a linear function with coefficients $w = (w1,\ ...,\ wp)$, but uses a sigmoid function to convert the output to a range of [0, 1], so that a decision boundary can be set to get classification results.

SVM Classification. SVM method is very commonly used for classification, and we use it to compare with its counterpart in regression.

Random Forest Classification. The random forest is used for classification.

5. Implementation Details

The scikit-learn library is employed to implement the machine learning models and TF-IDF embedding method. We use gensim[19] to training word2vec models, during training, the embedding dimension is set to 150 and window is set to 5. For universal encoder, we apply the English language universal encoder[20] in version 5. Default parameter set is used for linear and svm models. For the random forest method, the estimator parameter is set to 25 and the tree depth is not limited. For the svm rank method, using the default set of parameters does not work, because the convergence could be very slow if epsilon is not carefully chosen. We use a svm rank model with c equaling to 20 and tolerance equaling to 0.15.

[19] https://radimrehurek.com/gensim_3.8.3/models/word2vec.html
[20] https://tfhub.dev/google/universal-sentence-encoder-large/5

6. Results

	Regression TF-IDF	Regression Word2vec	Regression + Sentence Encoder	Svm rank + word2vec	Classification TF-IDF
Logit	0.636	0.587	N/A	N/A	0.41
SVM	0.648	0.618	0.667	0.593	0.43
RF	0.646	0.606	N/A	N/A	0.34

Figure 2 Experimental results using the metrics described in project overview

As can be seen from the table, SVM method has a robust performance, especially when paired with the use of a large pre-trained model as encoder. Among the three strategies, the classification method has the worst performance, and the regression method has the best performance.

The classification method produced relatively poor results. Thus, we have decided to drop this method at an early stage. We concluded that the reason that the classification method performs poorly is likely due to the improper selection of toxicity intervals for splitting toxicity levels. As seen from the data exploration section, our dataset has a tailed distribution, with most of the data on the left-side of the scale (value = 0). While less than 1% data is found between the range 0.9 to 1.0. If we split the range of toxicity into intervals of equal lengths, we could end up with extremely imbalanced datasets where data from certain classes are overrepresented and the others underrepresented. In our project, we merged contiguous intervals with too few samples together so that the distribution of the number of samples per interval was well-balanced. However, we encountered the issue that data falling into the merged class became hardly distinguishable. Furthermore, the min-max normalization of the toxicity level scores may also influence the result, as it compresses the information: the distance between data samples that have different levels of toxicity actually shrinks, making it harder to compare between data points.

However, the method of transforming the continuous toxicity level to discrete range can still be useful, for it preserves the model's confidence about the sample over different levels of toxicity, which can be inferred from the probability of each class the model predicts. As a result, more experiments can be done on this part, such as trying different strategies of selecting intervals between ranges, different strategies of selecting the total classes in the range etc., to see how this method can help improve the model's ability on comparing the toxicity level between comment texts.

The performance of the ranking approach also falls behind that of the regression method. This may be due to the fact that the convergence of the model with large-scale data under the same query is sensitive to the selection of parameters, e.g. the selection of tolerance, and the regularization parameter C. For example, if

the tolerance parameter is set to low, the model will take too long to converge or not coverage at all. A way to combat the problem is to conduct a cross-validation and exhaustive search. However, this is highly time-demanding for a large-scale dataset, and therefore not always feasible.

The word2vec method was shown to have worse performance than TF-IDF. This may be due to the reason that the dimension of the word2vec is too low (in our case 150) to express a data size of over 2 million samples. For future improvement, we could train several models with different dimensions to select the one with the best performance. Furthermore, since the semantic of word2vec vectors does not vary with changes of context, we could use pre-trained word embeddings for common words and use embeddings trained on our corpus for domain-specific words, taking advantage of the power of large pre-trained language model while keeping the domain information.

The ranking method did not perform as well as the regression method but the result is comparable. The narrow discrepancy in performance might be due to the selection of hyperparameters. The result is inspiring to the future design of the models: we can potentially use the strategy of ranking directly in training of deep models under the frame of multi-task learning.

The best performing model under our experimental setting was Universal sentence Encoder embedding. However, this model is pre-trained and fixed, the domain of corpus on which it is trained is different from the task domain. This leaves room for fine-tuning the models and achieving better results.

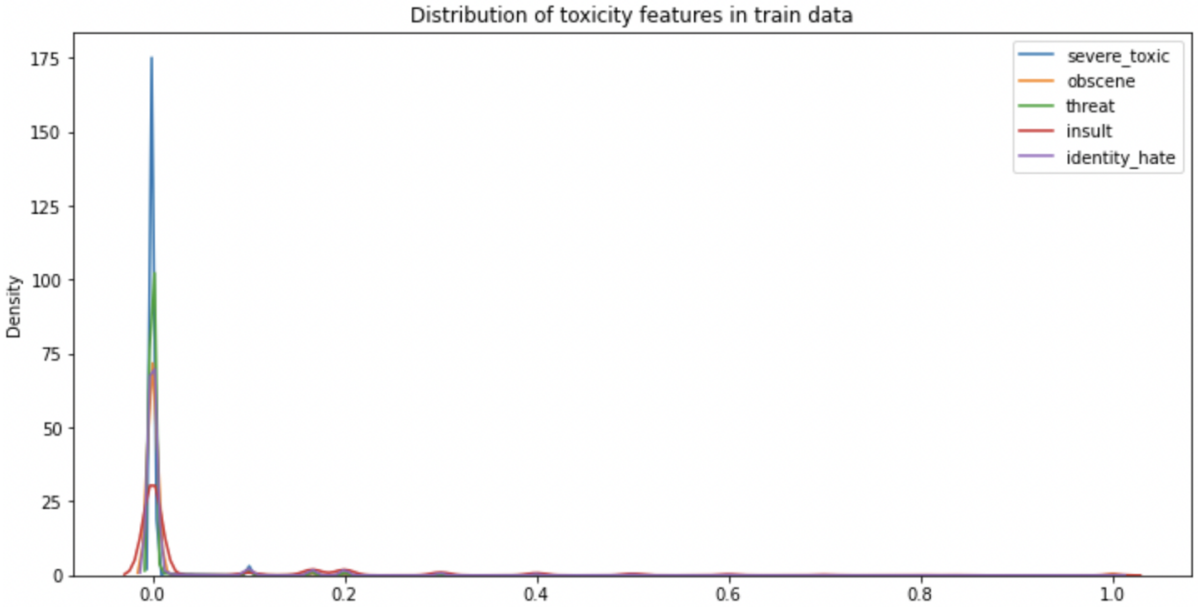
7. Conclusions

Our experiment has shown that using pre-trained large models for text embedding leads to decent results in the task of relative toxicity ranking of comments. In our project, the regression methods perform better than the classification and ranking methods.

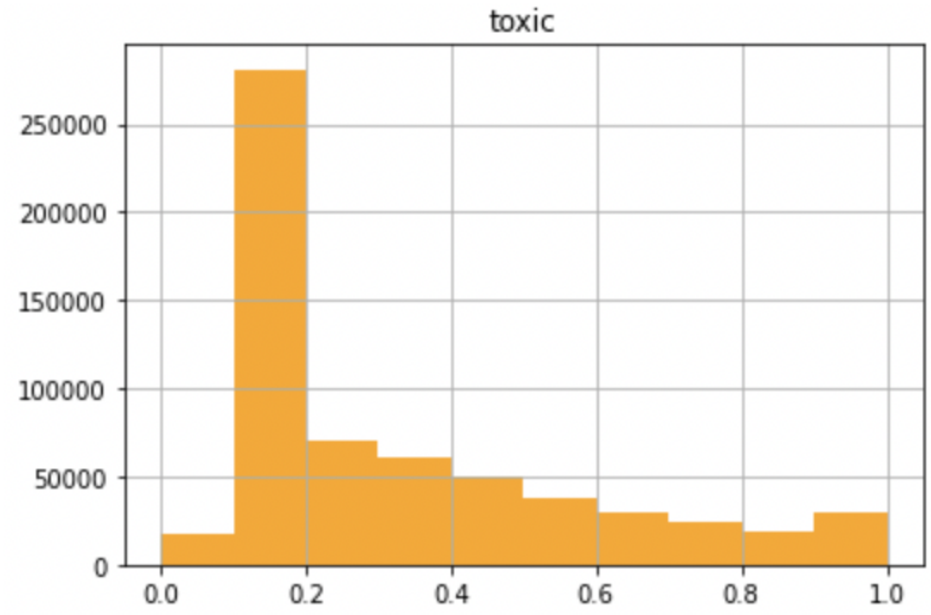
For future work, we would like to explore how the classification methods and ranking methods can help to design the model architecture and to improve the model's performance to compare the toxicity level of different comment texts. Besides, we plan to finetune deep language models according to the domain feature of our specific task to get better results.

Appendix

Appendix 1 a) The distribution of toxicity features in training dataset

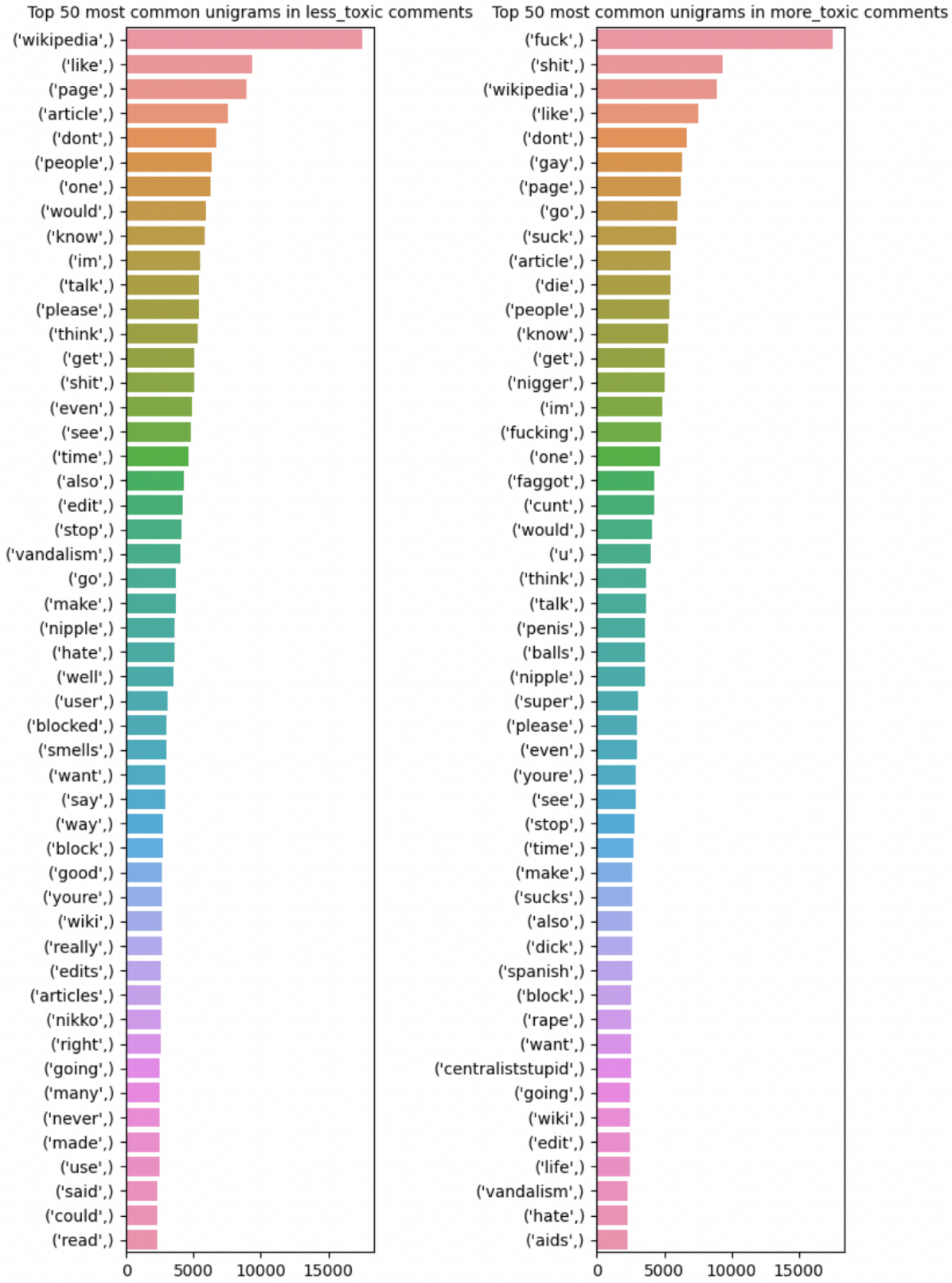


b) The distribution of levels of toxicity value among toxic comments in training dataset

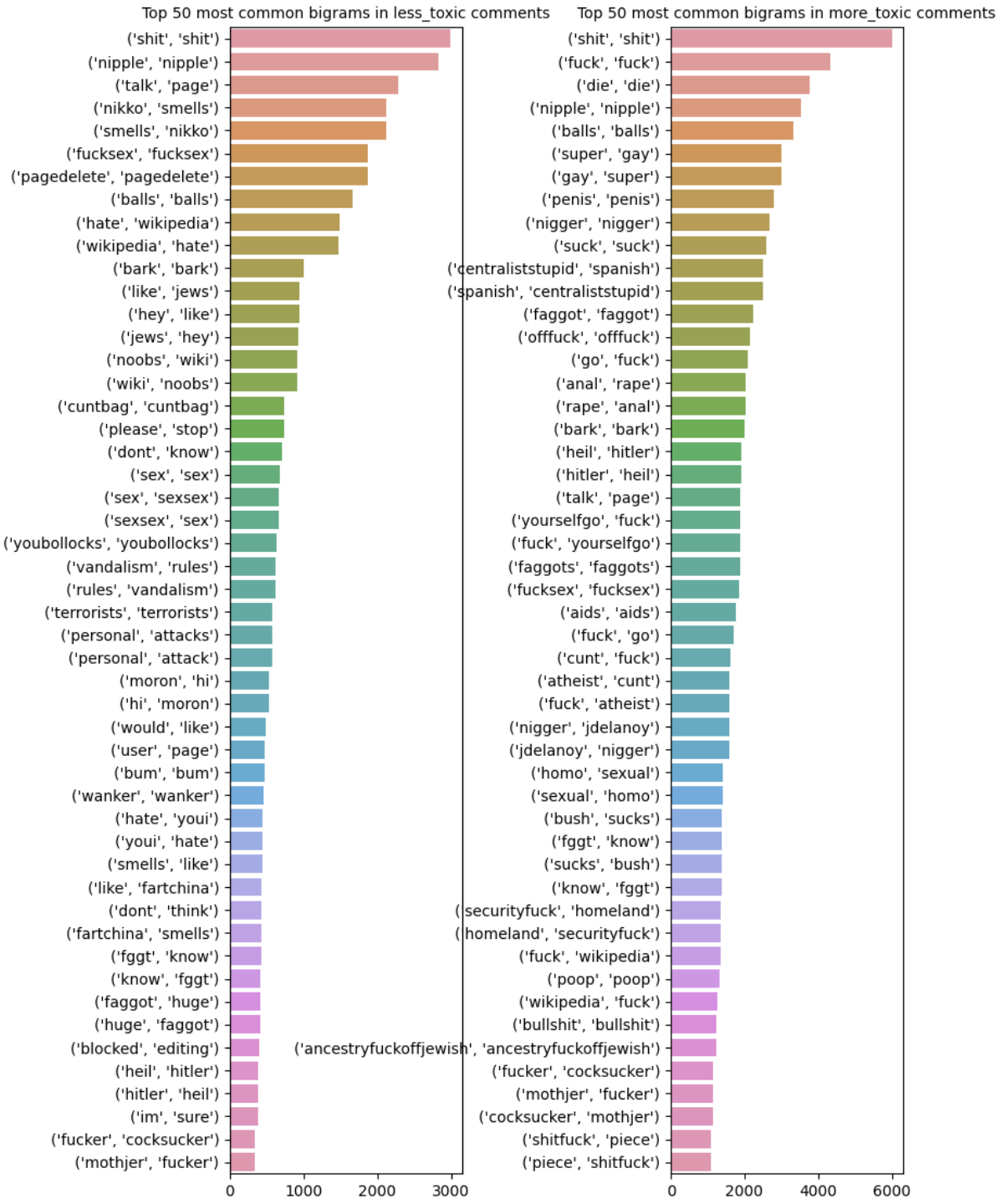


Appendix 2 N-gram Feature Visualisation

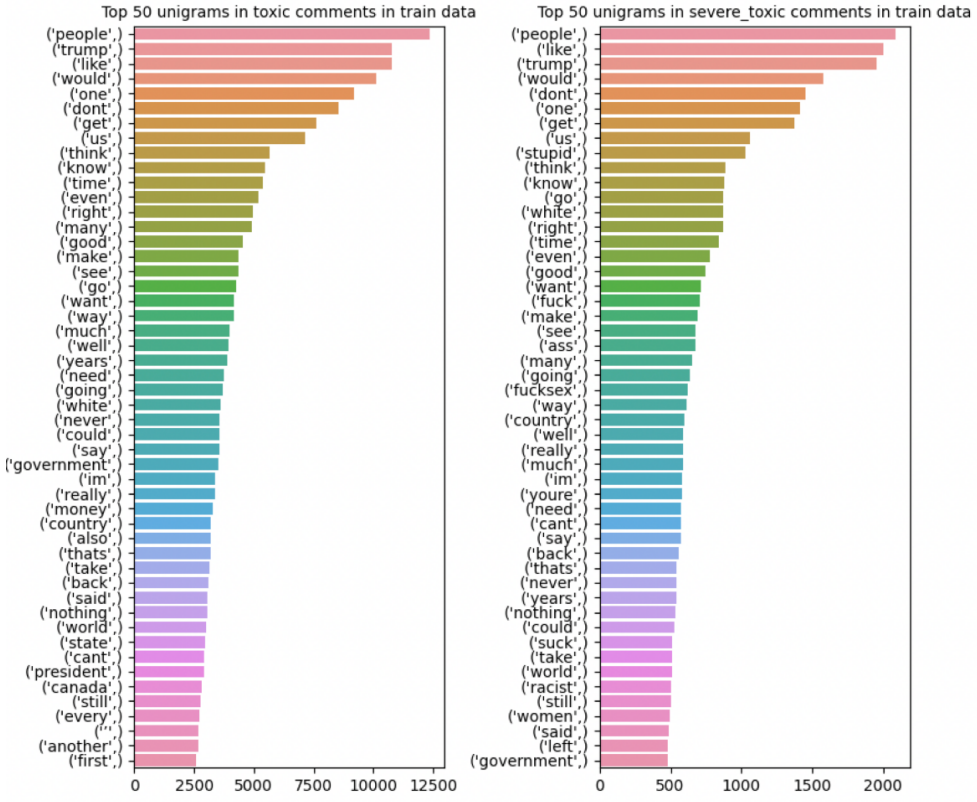
a) Top 50 unigrams in “less toxic” and “more toxic” comments in test dataset

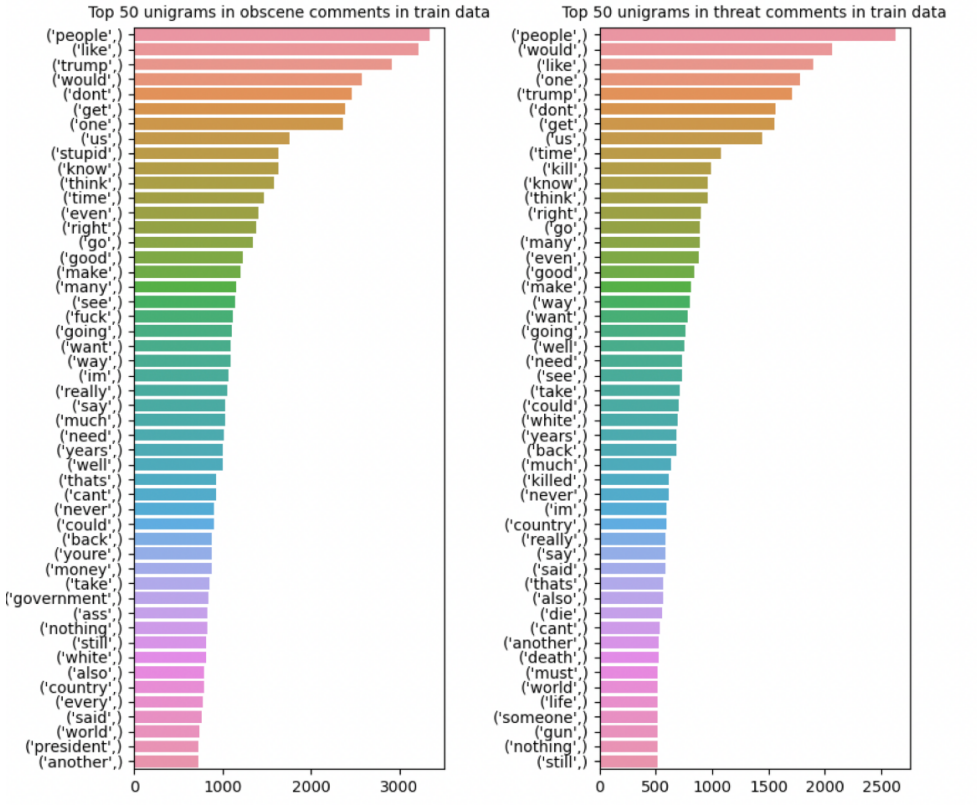


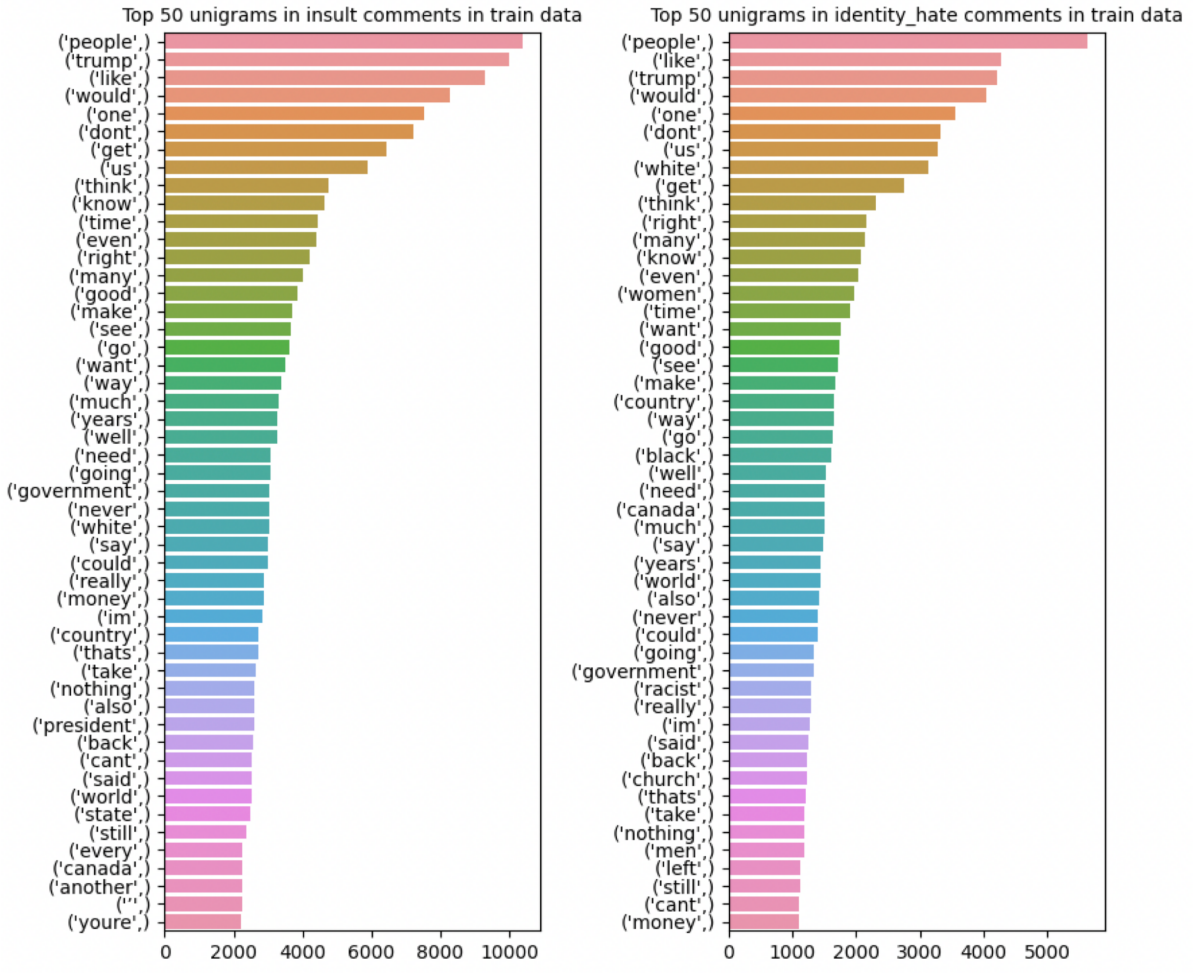
b) Top 50 bigrams in “less toxic” and “more toxic” comments in test dataset



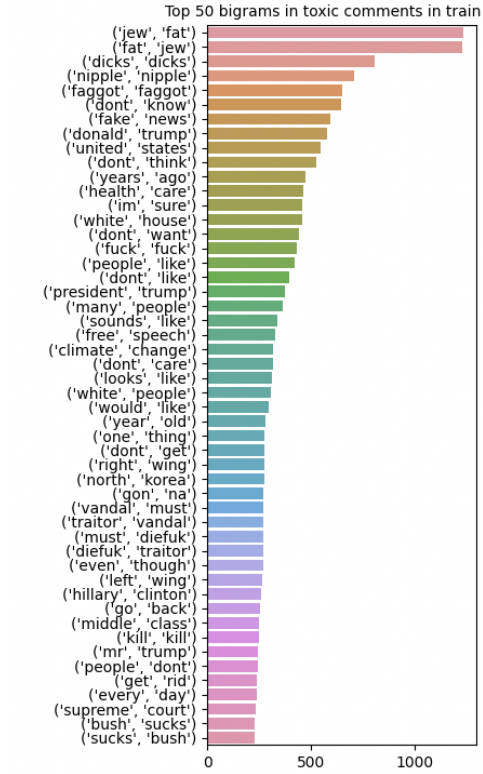
c) Top 50 unigrams in test dataset for each of six toxicity attributes

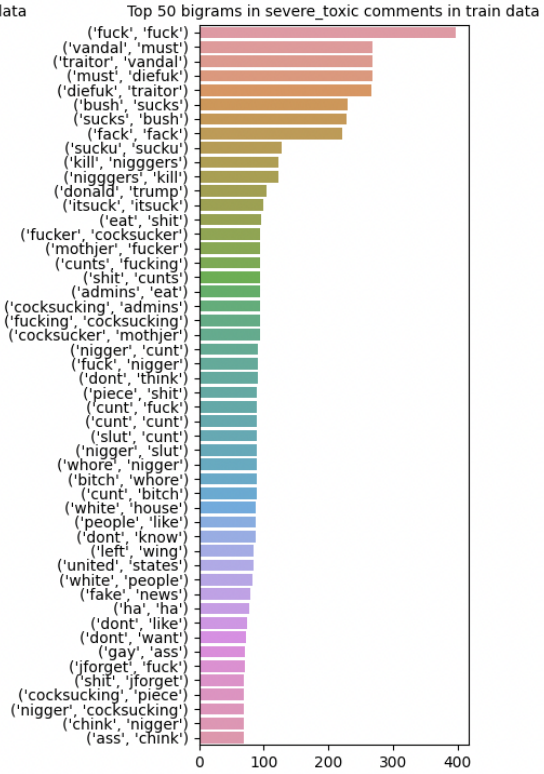


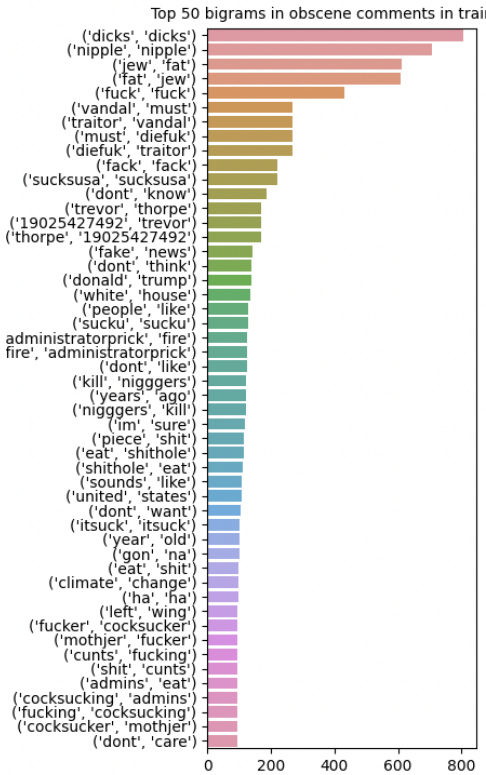


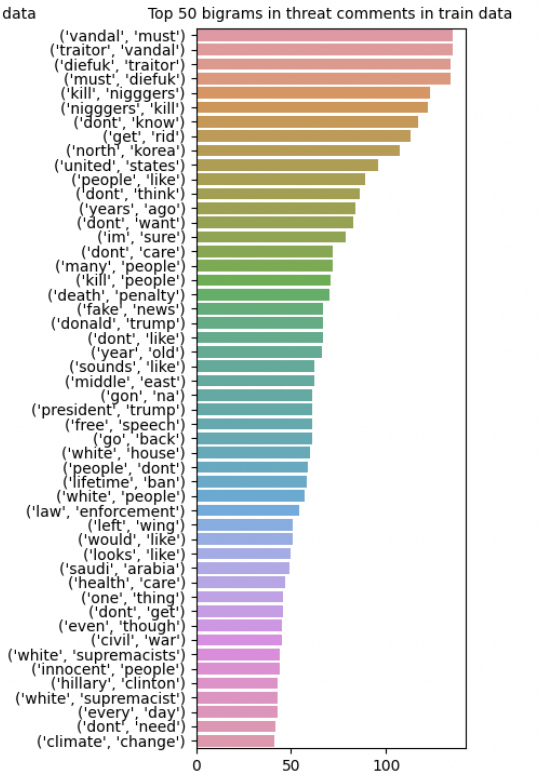

Top 50 unigrams in insult comments in train data
('people',)
('trump',)
('like',)
('would',)
('one',)
('dont',)
('get',)
('us',)
('think',)
('know',)
('time',)
('even',)
('right',)
('many',)
('good',)
('make',)
('see',)
('go',)
('want',)
('way',)
('much',)
('years',)
('well',)
('need',)
('going',)
('government',)
('never',)
('white',)
('say',)
('could',)
('really',)
('money',)
('im',)
('country',)
('thats',)
('take',)
('nothing',)
('also',)
('president',)
('back',)
('cant',)
('said',)
('world',)
('state',)
('still',)
('every',)
('canada',)
('another',)
('"',)
('youre',)
0
2000
4000
6000
8000
10000
Top 50 unigrams in identity_hate comments in train data
('people',)
('like',)
('trump',)
('would',)
('one',)
('dont',)
('us',)
('white',)
('get',)
('think',)
('right',)
('many',)
('know',)
('even',)
('women',)
('time',)
('want',)
('good',)
('see',)
('make',)
('country',)
('way',)
('go',)
('black',)
('well',)
('need',)
('canada',)
('much',)
('say',)
('years',)
('world',)
('also',)
('never',)
('could',)
('going',)
('government',)
('racist',)
('really',)
('im',)
('said',)
('back',)
('church',)
('thats',)
('take',)
('nothing',)
('men',)
('left',)
('still',)
('cant',)
('money',)
0
1000
2000
3000
4000
5000

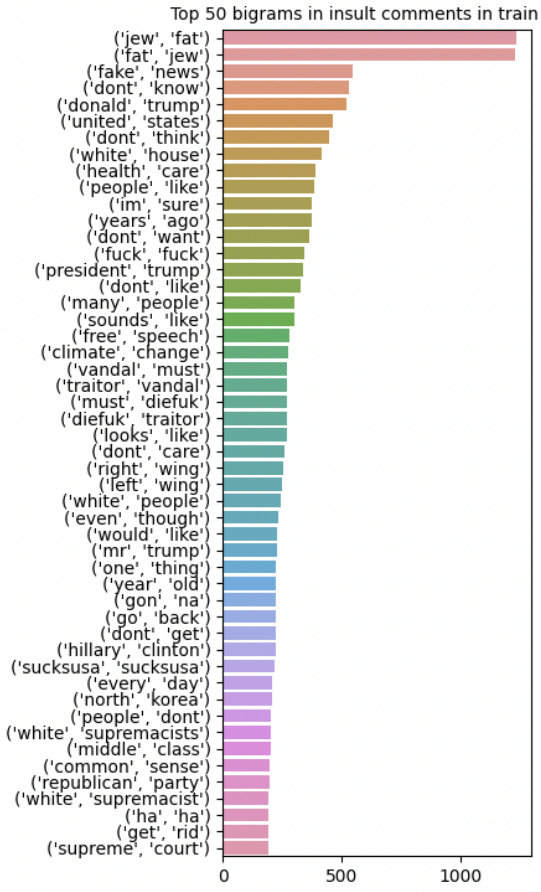
d) Top 50 bigrams in test dataset for each of six toxicity attributes

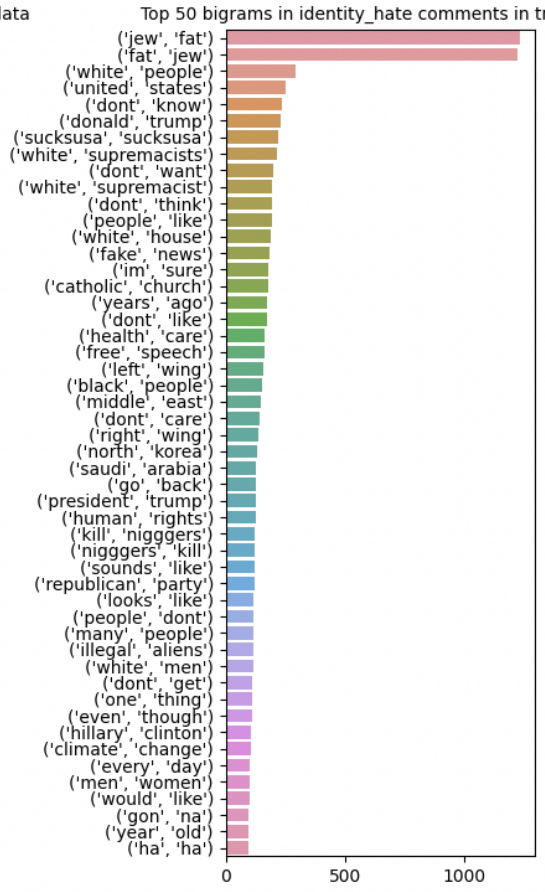









Top 50 bigrams in insult comments in train data
('jew', 'fat')
('fat', 'jew')
('fake', 'news')
('dont', 'know')
('donald', 'trump')
('united', 'states')
('dont', 'think')
('white', 'house')
('health', 'care')
('people', 'like')
('im', 'sure')
('years', 'ago')
('dont', 'want')
('fuck', 'fuck')
('president', 'trump')
('dont', 'like')
('many', 'people')
('sounds', 'like')
('free', 'speech')
('climate', 'change')
('vandal', 'must')
('traitor', 'vandal')
('must', 'diefuk')
('diefuk', 'traitor')
('looks', 'like')
('dont', 'care')
('right', 'wing')
('left', 'wing')
('white', 'people')
('even', 'though')
('would', 'like')
('mr', 'trump')
('one', 'thing')
('year', 'old')
('gon', 'na')
('go', 'back')
('dont', 'get')
('hillary', 'clinton')
('sucksusa', 'sucksusa')
('every', 'day')
('north', 'korea')
('people', 'dont')
('white', 'supremacists')
('middle', 'class')
('common', 'sense')
('republican', 'party')
('white', 'supremacist')
('ha', 'ha')
('get', 'rid')
('supreme', 'court')
0
500
1000


Top 50 bigrams in identity_hate comments in train data
('jew', 'fat')
('fat', 'jew')
('white', 'people')
('united', 'states')
('dont', 'know')
('donald', 'trump')
('sucksusa', 'sucksusa')
('white', 'supremacists')
('dont', 'want')
('white', 'supremacist')
('dont', 'think')
('people', 'like')
('white', 'house')
('fake', 'news')
('im', 'sure')
('catholic', 'church')
('years', 'ago')
('dont', 'like')
('health', 'care')
('free', 'speech')
('left', 'wing')
('black', 'people')
('middle', 'east')
('dont', 'care')
('right', 'wing')
('north', 'korea')
('saudi', 'arabia')
('go', 'back')
('president', 'trump')
('human', 'rights')
('kill', 'nigggers')
('nigggers', 'kill')
('sounds', 'like')
('republican', 'party')
('looks', 'like')
('people', 'dont')
('many', 'people')
('illegal', 'aliens')
('white', 'men')
('dont', 'get')
('one', 'thing')
('even', 'though')
('hillary', 'clinton')
('climate', 'change')
('every', 'day')
('men', 'women')
('would', 'like')
('gon', 'na')
('year', 'old')
('ha', 'ha')
0
500
1000

Appendix 3 LDA

a) Top 5 words in each topic for test data

Topics	Top 5 words
1	"nipple","penis","say","make","use"
2	"cunt","fucksex","damn","http","faggots"
3	"suck","hey","dick","jews","love"
4	"wiki","hate","noobs","bitches","bullshit"
5	"faggot","bark","gay","huge","aids"
6	"user","blocked","block","edit","edits"
7	"shit","vandalism","nigger","rules","stop"
8	"smells","nikko","pagedelete","super","gay"
9	"sex","ve","good","way","time"
10	"fuck","fucking","balls","die","ass"

b) Top 5 words in each topic for train data

Topics	Top 5 words
1	"canada","city","trudeau","canadian","know"
2	"did","party","news","mr","media"
3	"article","use","does","think","church"
4	"page","talk","want","ll","wikipedia"
5	"com","right","www","http","comment"
6	"years","time","women","long","school"
7	"money","tax","pay","state","oil"
8	"country","government","law","states","american"
9	"year","new","need","000","change"
10	"trump","president","obama","man","clinton"