MIE 1628: Twitter Toxic Comments Classification

Fan Chen 1004047514 Group 5

1. Introduction

Discussing things you care about can be difficult. The threat of abuse and harassment online means that many people stop expressing themselves and give up on seeking different opinions. Platforms struggle to effectively facilitate conversations, leading many communities to limit or completely shut down user comments.

In this project, we are aimed to develop models to identify different toxic comments including threats, obscenity, insults, and identity-based hate.

As a course project, all of the models are built based on Pyspark environment.

2. Dataset Description & Target Defination

The dataset is twitter context from Kaggle competition. The dataset is 68.8MB, including 159571 columns and 8 rows.

Table1. Sample Dataset

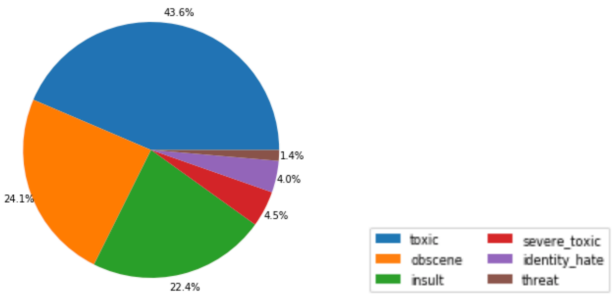
	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	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

Independent variables:1) ID(user ID), which is an unique value in the dataset. 2) comment_text
Response variables: 1) toxic, 2) severe toxic, 3) obscene, 4) threat, 5)insult and 6) identity hat

From table 2, We can see it's unbalanced classes, there are 159,571 twitters in total, 162,25/159,571 of them are labelled. Our target is to assign labels for each twitter by natural

Table 2. Label distribution

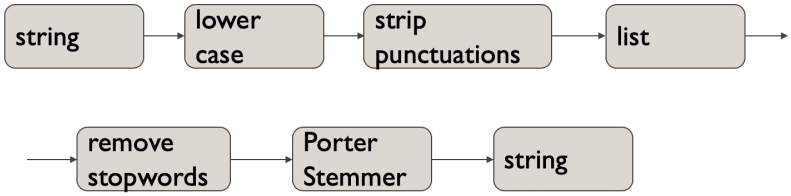
	category	number_of_comments
0	toxic	15294
1	severe_toxic	1595
2	obscene	8449
3	threat	478
4	insult	7877
5	identity_hate	1405



language processing. One instance(text) can belong to several labels in the same time. It's multi-label problem.

When we build our models on school's cluster, it often crashed due to the limitation of memory. Therefore, in our modelling process, we only sample 30000 comments for our tranning set and 20000 comments for test set.

3. Data Cleaning

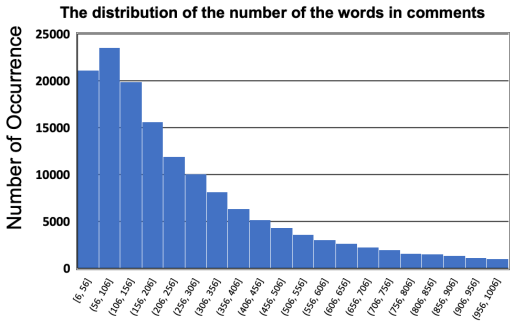


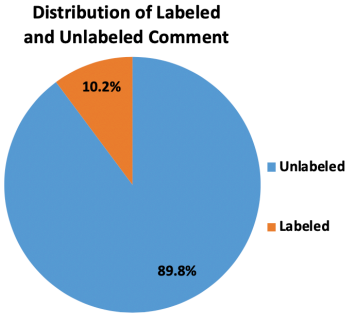
First, we need to clean the comment. String of comments go through the procedure of cleaning to get the principle words. First we lower case the comment, strip punctuations, remove stopwords, and do porter stemmer.

There are three challenges for this project.

1. NLP, feature engineering based on Pyspark
2. Unbalanced dataset. The maximum positive rate of label is 9.58%(toxic) and the minimum positive rate of label is 0.3%(threat).
3. Multi-label classification. One instance(text) can belong to none or several labels in the same time.

4. Data Exploration analysis





Common Toxic Words

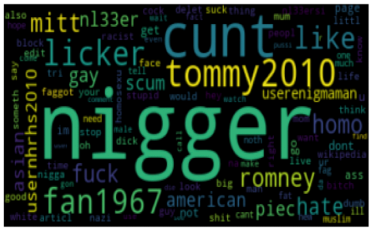
Common Severe Toxic Words

Common Obscene Words

Common Threat Words

Common Insult Words

Common Identity Hate Words



Only 10.2% of comments have label and most of these comments have a length less than 500. The last pictures shows the high frequency word in some classifications.

5. TF-IDF

5.1 TF-IDF Feature Engineering

We choose TF-IDF(which named as HashingTD-IDF in PySpark) as our feature transformation baseline, by this method we transform each comment as a sparse vector containing TF-IDF scores of each words in the comment.

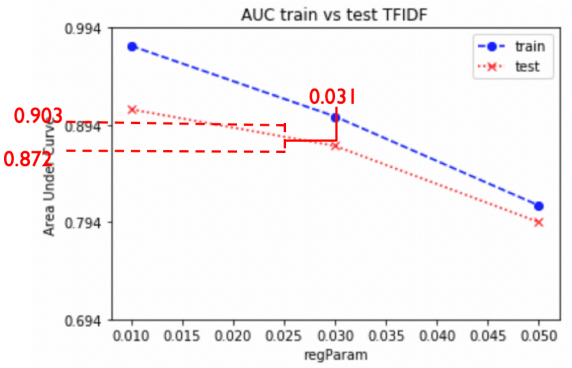
Each comment has been transformed as a sparse vector, there are 262144 features in total.

5.2 Modeling

For modelling, we choose **Logistic Regression**, which is recommended by our mentor, because we have imbalanced classes, thresholds for classification is hard to determine, we choose AUC as the metric to evaluate our model.

To fix imbalanced problem, we assign weights (= **λ** x **negative** label / **positive** label) for each classifications, and we can adjust **λ** to change our imbalance data penalty.

From experience, difference of AUC for train and test data should less than 0.04 to avoid overfitting, and we choose a L2 regularization to restrict hypothesis space.



When regParam(which is C in LR) = 0.03, we get the best result and also avoid over-fitting. The difference of AUC between training set and test set is 0.031.

5.3 Evaluation

Table. AUC and F1 Score for TF-IDF

	AUC	F1 Score
toxic	0.783733	0.334035827
severe_toxic	0.933066	0.279674797
obscene	0.855254	0.499284692
threat	0.890315	0.090909091
insult	0.855999	0.437130802
identity_hate	0.912502	0.242424242

We trained six models for different classifications. The result of AUC shows that the model has a good performace. F1 score is a combination evaluation of recall and precision. “Threat” shows a low F1 score due to only 0.02% positive labels. To further improve our model, we tried Bi-gram method.

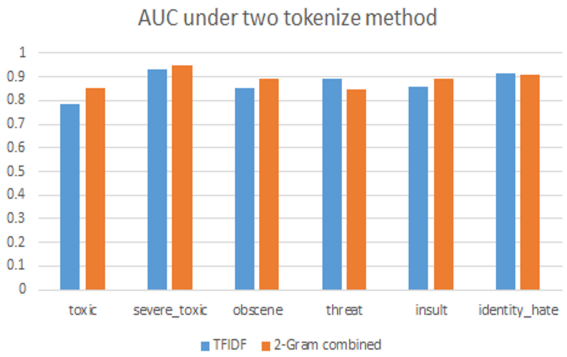
6. Bi-Gram

In the fields of computational linguistics and probability, an n-gram is a contiguous sequence of n items from a given sample of text or speech. The n-grams typically are collected from a text or speech corpus. For example, sentence **“**Hey man, I'm really not trying to edit war”. After, **Uni-gram**(after cleaning): [Hey, man, im, realli, not, tri, edit, war]. After **Bi-gram**: [Hey man, man im, im realli, realli not, not tri, tri edit, edit war].

By combining uni-gram and bi-gram tokenizations, we get almost **520,000** features, almost two times larger comparing to the baseline. We put these features into logistic regression and go through the sample process of TF-IDF. and we see the AUC of each label has increased by applying new feature transformation, that’s about 2% improvement. To improve our model further, we try word2vec.

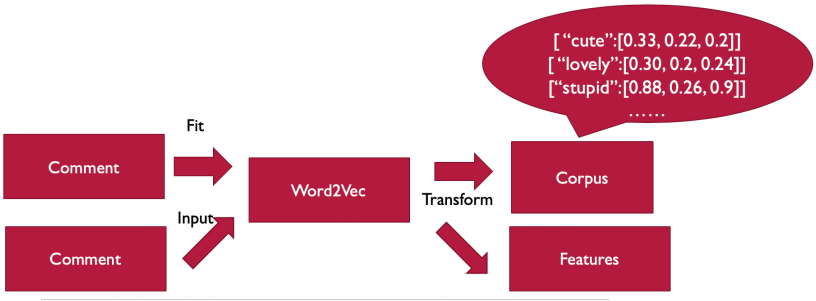
Table. AUC and F1 Score for Bi-Gram

	AUC	F1 Score
toxic	0.559341924	0.192798002
severe_toxic	0.730005946	0.192857143
obscene	0.613286561	0.197894737
threat	0.650330363	0.070175439
insult	0.597146347	0.109514855
identity_hate	0.57564879	0.087912088



7. Word2Vec

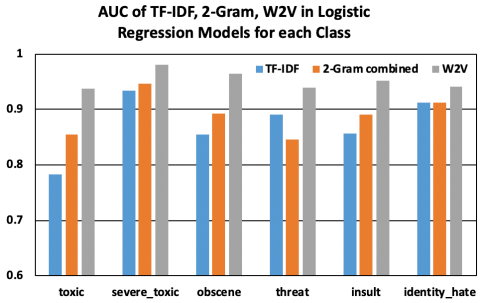
Word2vec takes as its input a large corpus of text and produces a vector space, typically of several hundred dimensions, with each unique word in the corpus being vector in the space.



First, we fit our comments into word2vec model and it will transform the large corpus of text into a vector space with 70 dimensions. Each unique word has a unique vector in the space and similar words will have similar vectors. Then we input our comment into the model and get the features after transformation. The feature is calculated by the average of the vectors of words in the sentence. Therefore, after this process, every comment will has a vector feature of 70 dimensions. We put these features into logistic regression and go through the sample process of TF-IDF.

Table. AUC and F1 Score for Word2Vec

	AUC	F1 Score
toxic	0.937696488	0.651196172
severe_toxic	0.980844552	0.41
obscene	0.964449341	0.647808765
threat	0.939887451	0.133971292
insult	0.951076315	0.575221239
identity_hate	0.941388188	0.228668942



Word2vec shows the best performance for all of classifications. It improved AUC by 6% and F1 score by 11% on average.

8. Conclusion

We have three challenges in the project. First, NLP feature engineering based on Pyspark. We tried three methods including TF-IDF, Bi-Gram, Word2Vec. The result shows that Word2Vec have the best performance. It's an imbalanced dataset. The maximum positive rate of label is 9.58%(toxic) and the minimum positive rate of label is 0.3%(threat). To solve the problem, we adjust class weight when we train the model and evaluate model by AUC & F1 score. Multi-label classification. One instance(text) can belong to none or several labels in the same time. We train six different models for six labels to solve the problem.

9. Expectation

The limitation of the cluster lead to only part of the samples used in the model. If we can have a powerful cluster, the model will have a better performance due to the increase of the training sample and dimensions of the features.

For feature engineering, we may try a new language representation model called BERT [2], which stands for Bidirectional Encoder Representations from Transformers. Unlike recent language representation models, BERT is designed to pre-train deep bidirectional representations by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT representations can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications.

For model, we can try different algorithms such as SVM, CNN, Naive Bayes and compare the results.

10. References:

[1] Kaggle competition, Toxic Comment Classification Challenge

https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge

[2] Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).