

**Design and Implementation of Analytics System**

***Classify toxic online comments to develop a classify system to identify inappropriate comments online***

***Project 4***

*Shaojie Lei*

*Chen-I Huang*

**School of Graduate Professional Studies**

MPS Data Analytics (DAAN)

DAAN 888 – Design and Implementation of Analytics System

Fall, 2020

Supervisor: Youakim Badr, Ph.D.

# Document Control

## Work carried out by Team (Name):

|  |  |  |
| --- | --- | --- |
| **Members** | **Email Address** | **Task description** |
| Shaojie Lei | Spl5620@psu.edu |  |
| Chen-I Huang | Cbh5543@psu.edu |  |

## Revision Sheet

|  |  |  |
| --- | --- | --- |
| **Release No.** | **Date** | **Revision Description** |
| 1 | 2020/09/06 | Group based assignment - Project Proposal |
| 2 | 2020/09/17 | Revise Project Proposal |
| 3 | 2020/09/19 | Revise Project Proposal |
| 4 | 2020/09/20 | Data Collection and Cleaning |
| 5 | 2020/10/04 | Model planning and data exploration |
| 6 | 2020/10/13 | Revise model planning and data exploration |
| 7 | 2020/10/18 | Feature engineering and model training |
| 8 | 2020/11/1 | Model evaluation |
| 9 | 2020/11/14 | Model Deployment |
| 10 | 2020/12/22 | Final Report |

**TABLE OF CONTENTS**

Contents

[Document Control 1](#_Toc56951204)

[Work carried out by Team (Name): 1](#_Toc56951205)

[Revision Sheet 1](#_Toc56951206)

[Project Proposal……………………………………………………………………………………………1](#_Toc56951207)

[Collecting Data and Data Cleaning 5](#_Toc56951208)

[Model Planning and Data Exploration 7](#_Toc56951209)

[Feature Engineering and Model Building 18](#_Toc56951210)

[Model Validation and Visualization 20](#_Toc56951211)

[Model Deployment 28](#_Toc56951212)

[FINAL Report…………………………………………………………………………………………….30](#_Toc56951213)

[Project Presentataion & DeMO 31](#_Toc56951214)

[Reference………………………………………………………………………………………………….32](#_Toc56951215)

**General Guidelines**

1. To complete all the homework assignments for this course please use this template document.
2. Each assignment has to be submitting by Sunday 11:59 PM EST.
3. Each figure should be followed by a brief description about the figure.
4. The figures can be hand drawn and scanned in some circumstances but the hand drawn figure should be clear and legible to obtain full credits. Unclear hand drawn figures will receive partial credits. For constructing figures and diagrams it is advised to use tools.
5. Figures and tables should have appropriate captions. For documenting and referencing styles please follow the APA or MLA writing style.
6. Please make sure that you provide a reference section.
7. Any material text or figure taken from books, journals or Internet should be referenced. If you have a sentence or a figure that does not belong (authorship) to you they need to be clearly referenced. If you fail to do so your report will be considered as a case for plagiarism. It is your duty to make sure that your report is free from any activity related to plagiarism. In case you are suspected of attempting plagiarism then you will be responsible for the cause. The penalty for plagiarism will be a “0” awarded to your report. So it is good to keep simple, always have the principle to acknowledge people for their contributions.

Please go through the following instructions before submitting the report

#### **Academic Integrity**

Academic integrity — scholarship free of fraud and deception — is an important educational objective of Penn State. Academic dishonesty can lead to a failing grade or referral to the [Office of Student Conduct](http://www.sa.psu.edu/ja/).

Academic dishonesty includes, but is not limited to:

* cheating
* plagiarism
* fabrication of information or citations
* facilitating acts of academic dishonesty by others
* unauthorized prior possession of examinations
* submitting the work of another person or work previously used without informing the instructor and securing written approval
* tampering with the academic work of other students

#### How Academic Integrity Violations Are Handled

In cases where academic integrity is questioned, [procedure requires an instructor to notify a student](http://www.psu.edu/oue/aappm/G-9-academic-integrity.html) of suspected dishonesty before filing a charge and recommended sanction with the college. Procedures allow a student to accept or contest a charge. If a student chooses to contest a charge, the case will then be managed by the respective college or campus Academic Integrity Committee. If a disciplinary sanction also is recommended, the case will be referred to the [Office of Student Conduct](http://www.sa.psu.edu/ja/title=).

All Penn State colleges abide by this Penn State policy, but review procedures may vary by college when academic dishonesty is suspected. Information about Penn State's academic integrity policy and college review procedures is included in the information that students receive upon enrolling in a course.

Additionally, Penn State students are expected to act with civility and personal integrity; respect other students' dignity, rights, and property; and help create and maintain an environment in which all can succeed through the fruits of their own efforts. An environment of academic integrity is requisite to respect for oneself and others, and a civil community.

#### For More Information on Academic Integrity at Penn State

Please see the [Academic Integrity Chart](http://www.campuses.psu.edu/CAO.pdf)  for specific college contact information or visit one of the following URLs:

* Penn State Senate [Policy on Academic Integrity](http://www.psu.edu/dept/oue/aappm/G-9.html)
* [iStudy for Success!](http://istudy.psu.edu/tutorials/) — learn about plagiarism, copyright, and academic integrity through an educational module
* [Turnitin](http://tlt.its.psu.edu/turnitin) a web-based plagiarism detection and prevention system

# Project Proposal

1. **Project title**

Classify toxic online comments to develop a classify system to identify inappropriate comments online

1. **Background and Challenges**

**Backgroud**:

Our project will focus on identifying the negative online behavior, like toxic comments.

Then, classify these negative comments into more detailed categories such as toxic,

severe toxic, obscene, threat, insult and identity hate.

* Online news, social, and gaming platforms continues to grow
* 39% expert and leader think that future will be shape by harassment

**Challenges**:

* Online comments are made in a wide variety of context and contain words from many different formal and informal lexicons
* Spelling and grammar mistakes
* Dataset are labeled differently
* Varieties of pre trained models
* Large computation needed for pre-train model
* Never build google chrome extension before

1. **Project Objectives**

* Detect toxic comments
  + - Give the user warning that the comments may contain inappropriate content
    - For websites or social media to detect the improper comments and take action
* High accuracy models
  + - Word's meaning depends on semantic and have buzzword, slang...etc
    - Improve the accuracy, make sure can work on real world
* Implemented with chrome extension or API, etc.
  + - Convenient and real-time detection

1. **Feasibility Studies**

**4.1 Scientific Survey**

* Keywords that describe your project
  + NLP, sentiment analysis, text classification

1. Toxic Comment Detection and Classification - H Li, W Mao, H Liu
   * This paper uses Civil Comments dataset from Kaggle
   * They use Naive Bayes SVM, LSTM Model, BERT, and Ensembling to train the model, and compare the accuracy
   * This paper very helpful to us, the concept and workflow is same as us
2. Deep Learning for Hate Speech Detection in Tweets - Pinkesh Badjatiya,

Shashank Gupta, Manish Gupta, Vasudeva Varma

* + This paper used a dataset of 16k annotated tweets by D.Hovy
  + They used CNN, LSTM, SVM and GBDT
  + Embeddings learned from deep neural network models when combined with gradient boosted decision trees led to the best accuracy value

1. Toxic Comment Detection in Online Discussions-Julian Risch, Ralf Krestel
   * This Paper used datasets collected from Twitter which contains 7 different class, such as, Clean, Toxic, Obscene, Insult, Identity Hate, Severe Toxic and Threat
   * They mainly focused on GRU, and LSTM and combination of algorithm to predict the outcome
   * The cause of misclassification of comments are toxicity without swear words, sarcasm, irony and mislabeled comments

1. [NLP]Toxic Comment Classification-Joyce Chang
   * Dataset are collected from Kaggle (the same dataset that we are using)
   * He used Multinomial NB, Logistic Regression and Linear SVC
   * With simple LR model, we can reach a satisfied result in this case, with boosting and ensembling, it can achieve a more aggressive f-score
2. A Machine Learning Approach to Comment Toxicity Classification-Navoneel Chakrabarty
   * Wikipedia Talk Page Dataset prepared by Jigsaw and now publicly available at Kaggle is used
   * Used removal of punctuation, lemmatization, and removal of stop words during the text preprocessing stage
   * Used Decision Tree and SVM with linear kernel for predicting the comment class, and Decision Tree performed the best
3. Convolutional Neural Networks for Toxic Comment Classification- Spiros.V. Georgakopoulos, Sotiris K. Tasoulis, Aristidis G. Vrahaatis
   * Dataset used are from Kaggle’s competition regarding Wikipedia’s talk page edits.
   * Compare CNN against traditional bag-of-words approach combined with a selection of algorithm
   * CNN can outperform traditional algorithms on toxic comments classification

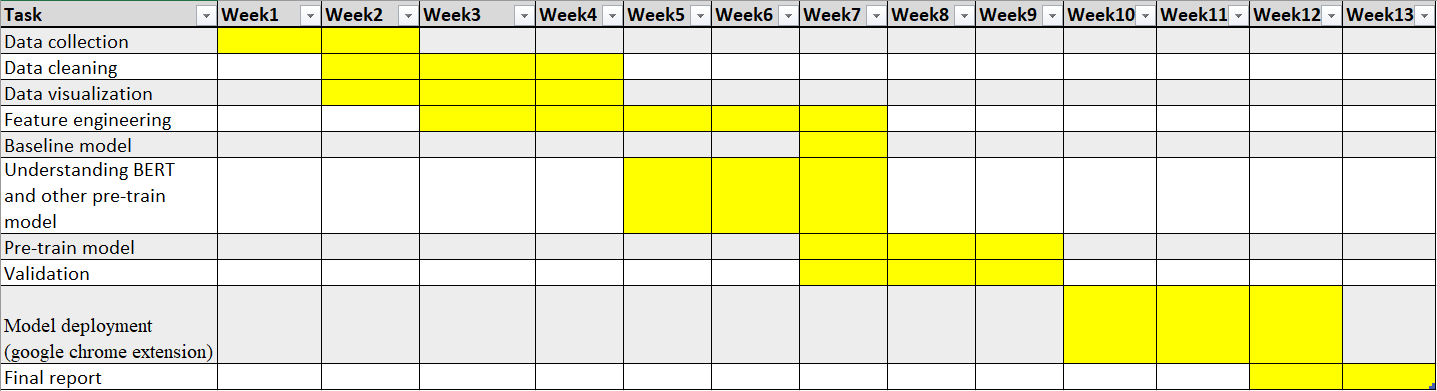
**4.2 Technical Survey**

* Google Jigsaw - Conversation AI
  + A chrome extension from Google’s Conversation AI combined with machine learning technique can filter out toxic comments
* Conversation AI provide API but no code for pre-trained model
  + <https://perspectiveapi.com/#/start>
* The purpose of Conversation AI and our project are the same, but they didn’t release their model or code, so we don’t know their exact solution.

1. **Methodology (Data Analytics System)**

* Data source
  + - Wikipedia comments with different type of toxicity - Kaggle
* Data analytics workflow
  + - Collecting the dataset
      * Separate into train, test, and validation data set
    - Data cleaning
      * Remove stop word
      * Tokenization
      * Stemming
      * Lemmatization
    - Data visualization
      * World cloud
      * TFIDF
    - Model training
      * Only one embedding and LSTM layer as our baseline model
      * Train on training data set and test on testing data set
    - Use pre-train model
      * Use pre-train model from transformers, like BERT
      * Train on training data set and test on testing data set
    - Validation
      * Use the validation set to test the accuracy of baseline model and pre-train model
    - Build a system
      * HTML + JS + CSS to write google chrome extension

1. **Deliverables** 
   * Write-up report
   * Reproducible Data Analytics Workflow
     + Commented Code
     + Datasets
     + Saved trained models
   * Slides (Presentation)
2. **Importance and Impacts**
   * Discussing thing 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.
   * Anti-discrimination and racial equality are getting more and more attention. For example, the BLM recently, we can use the system we develop to prevent the discrimination and keep the harmony online.
   * Reduce the suicide rate caused by cyberbullying, in a recent survey, people who get cyberbullied are 40% more likely to commit suicide, and people who cyberbully others are 16% more likely to commit suicide.
3. **Project Schedule**



1. **Required Technologies**

* Programming language
* Python
* HTML, JS, CSS
* Packages
* Keras/Pytorch - Deep learning model training
* NLTK - Tokenization, POS, Stemmer...etc
* Wordcloud – Word cloud
* Sklearn – bag of word, TFIDF
* Tools
* Huggingface/transformers

# Collecting DaTA and Data Cleaning

**Purpose:**

For this part, we collected the dataset from Kaggle.com, the dataset contains text, id, and different types of toxicity. After doing some simple exploration, our dataset is pretty much cleaned except for imbalanced data. So, handling imbalanced dataset is essential for the project.

**Collecting data:**

1. Data sources: Toxic Comment Classification Challenge – Kaggle (<https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/data?select=test_labels.csv.zip>)
2. Dataset stored: CSV file
3. Data summary:

|  |  |  |  |
| --- | --- | --- | --- |
| Column name | Numbers | Types | Example |
| id | 159571 | String | 000103f0d9cfb60f |
| Comment\_text | 159571 | String | Fuck my stupid muslim ass |
| toxic | 159571 | int | 1 |
| sever\_toxic | 159571 | int | 1 |
| obscene | 159571 | int | 0 |
| threat | 159571 | int | 0 |
| insult | 159571 | int | 1 |
| identity\_hate | 159571 | int | 0 |

1. The data meets our research/business needs, with have sufficient number of data (159571) and already labeled. Moreover, the data was labeled into six category (toxic, sever\_toxic, obscene, threat, insult, identity\_hate), which can train our model classify the comment into these category, not just toxic or not.
2. Additional data: additional data might be required in the future if the model didn’t perform well. For additional dataset, we can find other open source dataset (<http://hatespeechdata.com/>) or crawl down form Twitter, Reddit...etc.

**Data Cleaning:**

Since we are do deep learning on NLP, most of the following data cleaning steps are for data visualization, to have a better understanding of data.

1. Check the missing values and duplications
2. Remove stop word
3. Tokenization
4. Stemming
5. Lemmatization
6. Data visualization
   1. World cloud
   2. TFIDF (unigrams, bigrams)
   3. Correlation between each class
   4. Correlation with the length of comments
7. Handle imbalanced data, which also important for neural network

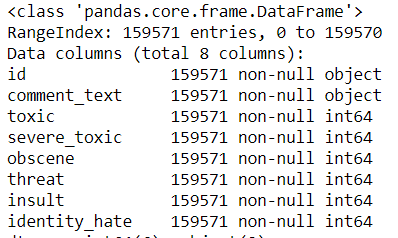
# Model Planning and Data Exploration

**Purpose:**

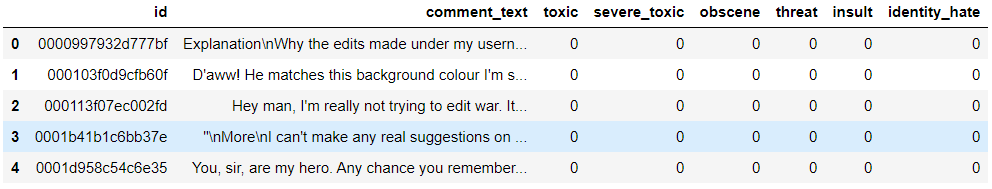
Our goal for this part was doing detailed data exploration and selected appropriate models to train. In Data exploration, we are using bar chart to find the frequency for each types of toxicity, and word cloud to show the most frequent text for that type of toxicity. Then, we will discuss the pretrained model and their advantages

**Data Exploration**

1. **Data summary**
   1. Check data type

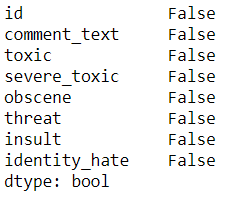


The data has 8 columns and 159571 rows. The data type for six class (toxic, sever\_toxic, obscene, threat, threat, insult, identity\_hate) are int, and object for id and comment\_text. The following is an example of the data:



* 1. Check null value

## train.isnull().any()



The data has no null value.

* 1. Number of clean data and at least one class

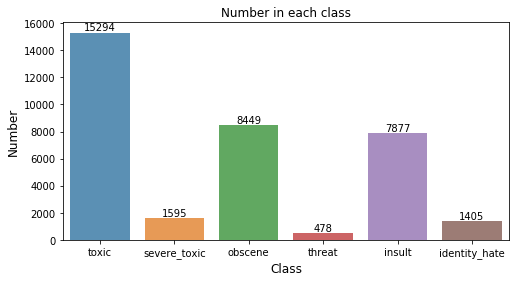
Total comments = 159571

Total clean comments = 143346

Total with at least one class = 35098

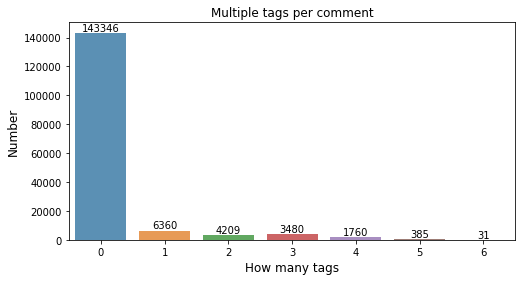
Most of the data are clean, which means not toxic

* 1. Check the numbers in each class

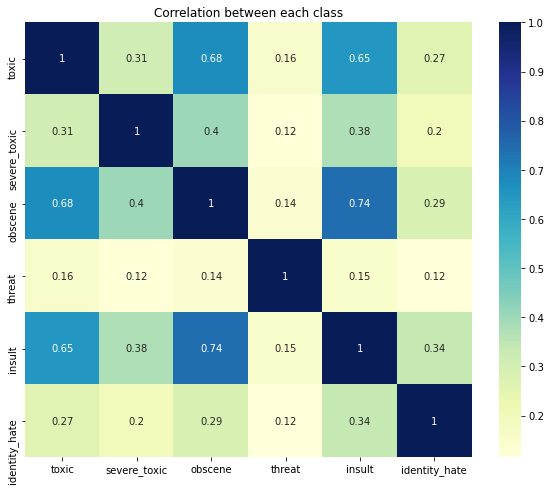


We can see from the plot that toxic has the most numbers, and severe\_toxic, threat, and identity\_have have very less numbers.

* 1. Check multiple tagging



* 1. From the plot, we can see that most of the data belongs to clean comments, which is 0 tags in the plot. And the 6 tags data, which means its toxic, sever\_toxic, obscene, threat, threat, insult, and identity\_hate have the less number. In the plot, we know that our data is imbalance, to many data belongs to clean comments, which is 0 tags in the plot. Therefore, imbalance data will be an important issue, and we need deal with imbalance data.
  2. Check the correlation in each class

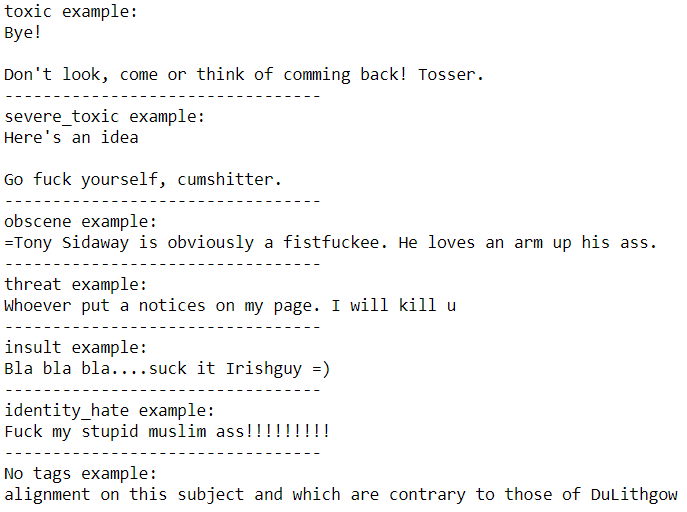


Toxic is highly correlate with obscene and insult, but not with severe toxic. Obscene is highly correlate with insult and toxic. Threat does not have much correlation with other categories. Insult has highly correlate with obscene and toxic.

1. **Visualize data**
   1. NLP preprocess

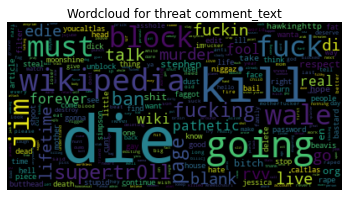
To present our text data, we need to do NLP first and then we can have word cloud and calculate the TF-IDF. The step of our NLP included the following:

* + 1. Lower case: Convert all word into lower case, so that same word with lower case or upper case will not be count into different word. For example, Analytics and analytics are the same.
    2. Remove punctuation: Remove punctuation like ?,!., so that it won’t affect our visualization.
    3. Remove stopword: Remove stopword like I, can, a. because most of the time, these word are meaningless in the text.
    4. Lemmatization: Lemmatization is the process of converting the words of a sentence to its dictionary form. For example, given the words amusement, amusing, and amused, the lemma for each and all would be amuse. Here we did not choose stemming is because sometime stemming will cut the word into only root, not a recognize word.
  1. Example of comments in each class

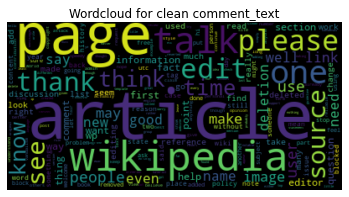


* 1. Word cloud for each class



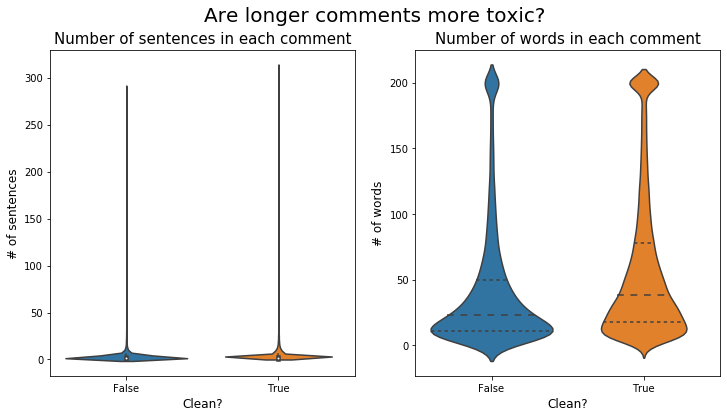






Some words appear frequently in different categories. For example, fuck appear in toxic, severe toxic, obscene, and insult. Insult and identity have many same words. Like nigger, die, jew fat.

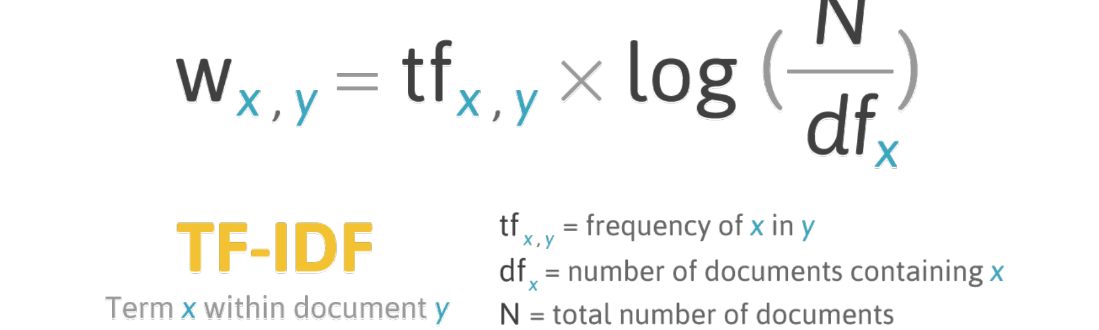
* 1. Relation between toxic and length of comments



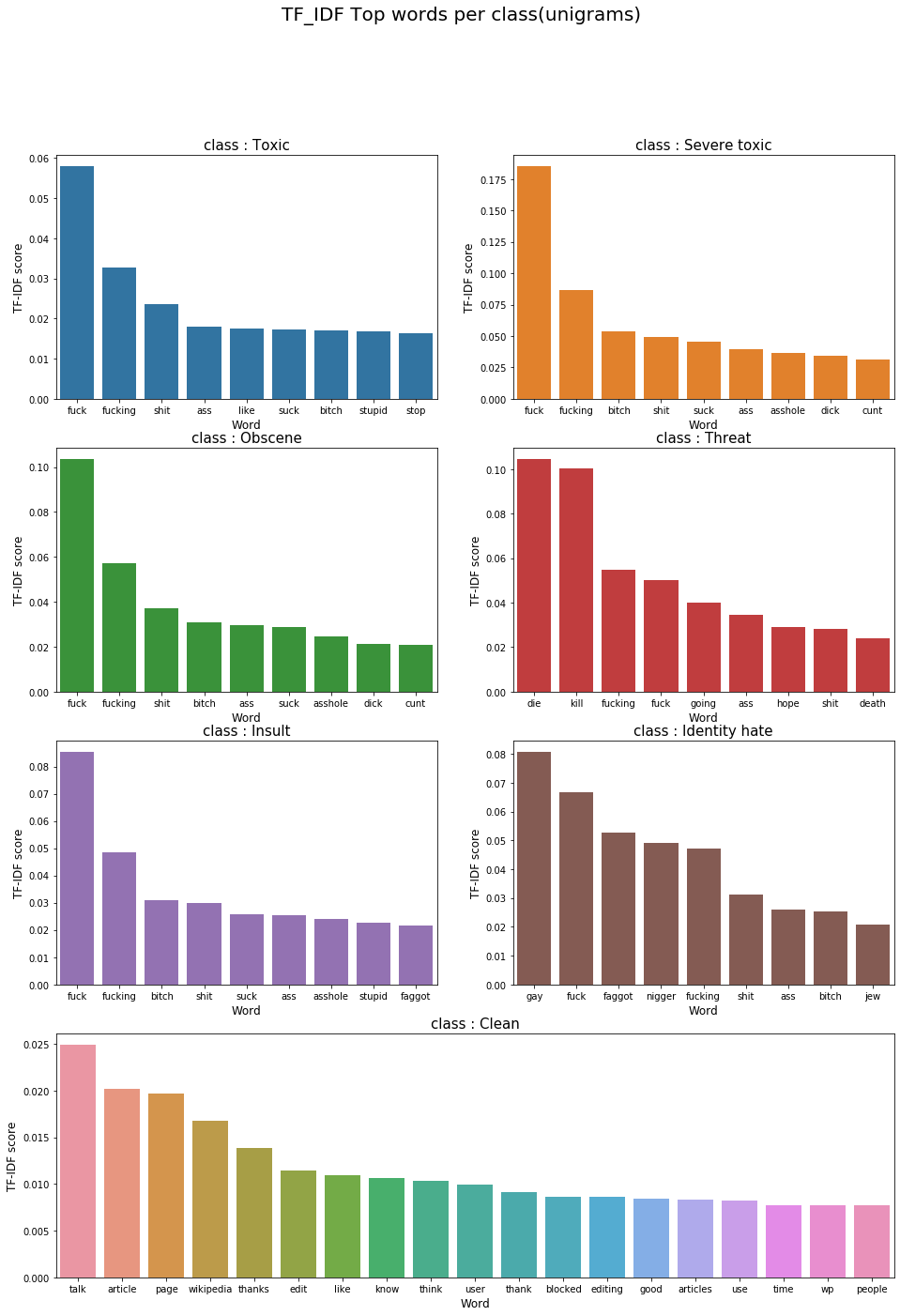
From the plot, we know that the number of sentences in comment and number of words in comment has no correlation with toxic or not. Because the distribution of number of sentences and number of words in clean comment and not clean comment (which means at least belongs to one of toxic, sever\_toxic, obscene, threat, threat, insult, and identity\_hate) are almost the same.

* 1. TFIDF

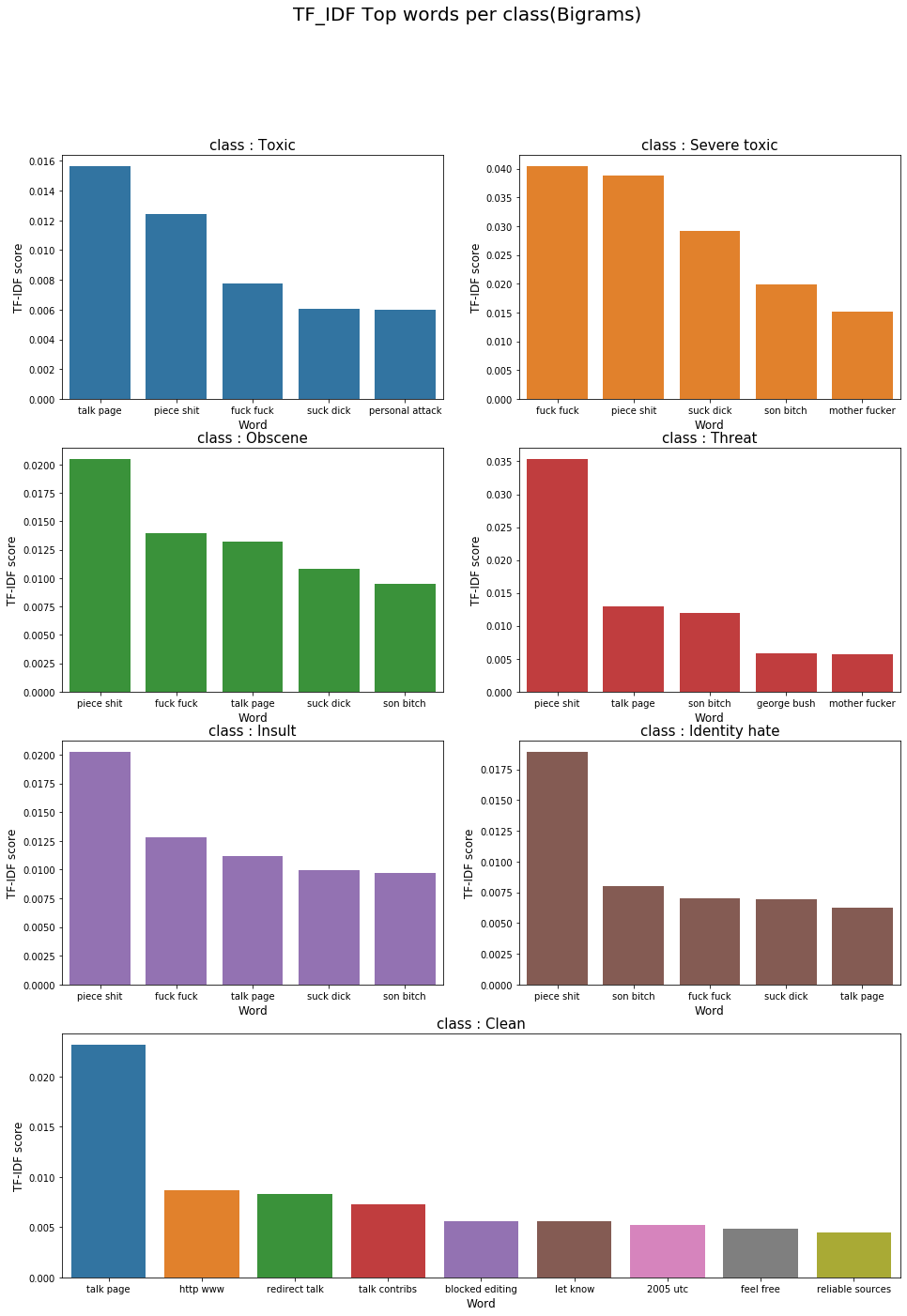
TF-IDF is a statistical measure that evaluates how relevant a word is to a document in a collection of documents. This is done by multiplying two metrics: how many times a word appears in a document, and the inverse document frequency of the word across a set of documents. The higher the score, the more relevant that word is in that document.



The following is the top TF-IDF score in each class.



From the plot, we can know how important each word is in the class. Fuck ranked very high in all class, which is very reasonable. Gay, nigger, bitch, jew appears in identity hate are also very reasonable. Beside consider one word at a time(unigram), we can also consider two words at a time(bigram). Then we can understand the semantic better. The follow is the top TF-IDF score for bigrams in each class.



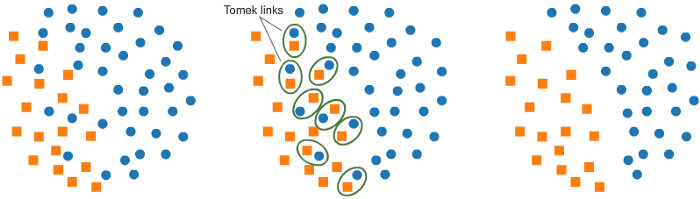
Compare to unigram, bigram will let us see what comes after the word, not just one word. From this plot, the ‘talk page’ appears in many categories which we don’t know why.

1. **CONDLUSION**

From Data Exploration, we had a better understanding of our data. Although we are not using machine leaning model, we didn’t need to extract features. The data visualization still can help us have a clear look of our data. Moreover, it helped us to explain our data to other people.

**Method for Handling Imbalance**

1. **Random Over-Sampling**
   1. Random Over-Sampling can be defined as adding more copies of the exists observations to the minority class. This method works very well when you do not have a lot of data to work with.
2. **Random Under-Sampling**
   1. Random under sampling can be defined as removing some observations until the majority class and minority class are balanced out. This method works better for our dataset since we have 160000 rows of data in our original dataset, but there are draw back for this method, one of which is that we might remove important information from the dataset
3. **Tomek Link**
   1. Tomek Links are pairs of very close instances but of opposite classes. Removing the instanced of the majority class of each pair increases the space between the two classes, facilitating the classification process



1. **SMOTE**
   1. This Technique creates data for the minority class, it picks a point from the minority class and computing the k-nearest neighbor for this point. The synthetic points are added between the chosen point and its neighbor

**Model Planning**

1. **LSTM**
   1. LSTM are a special kind of RNN, capable of learning long-term dependencies. Unlike standard feedforward neural networks, LSTM has feedback connections. It can not only process single data points, but also entire sequences of data (such as speech or video). For example, LSTM is applicable to tasks such as unsegmented, connected handwriting recognition, speech recognition and anomaly detection in network traffic or IDSs. LSTM are now widely used for continuous data especially sentence, which is suitable for our needs.
   2. We will build a simple LSTM model as our baseline model and compare with the following transfer learning that use pre train model. Since LSTM are easy to build compare to another complicate model like the following model. But the accuracy may not be as good as the following model.
   3. Reference:  
      Long short-term memory. (2020, September 27). Retrieved October 04, 2020, from <https://en.wikipedia.org/wiki/Long_short-term_memory>
2. **Bert**
   1. BERT stands for Bidirectional Encoder Representations from Transformer, it is developed by a group of researchers from google. It is a deep learning model in which every output element is connected to every input element, and the weightings between them are dynamically calculated based their connection(Text Classification With Bert Model).
   2. BERT has advantage over other traditional language models, traditional model can only read text from left to right or from right to left, BERT model can read text in both direction, BERT model can be built for NLP without the need to sourcing massive datasets, and it is a pretrained model on a large corpus of unlabeled text , the accuracy and cost of computation for this model should be relatively high.
   3. Reference:   
      Khalid, S., Says:, Y., Says:, S., Says:, J., Says:, P., & Says:, J. (2019, September 17). BERT Explained: A Complete Guide with Theory and Tutorial. Retrieved October 04, 2020, from <https://towardsml.com/2019/09/17/bert-explained-a-complete-guide-with-theory-and-tutorial/>
3. **DeText**
   1. DeTextis an open source framework designed for ranking, language generation works and classification. It supports semantic matching using neural networks to understand the text.
   2. Compare to LSTM, Detext can have better accuracy than simple LSTM. And compare to BERT, the computation need is not that heavy. Therefore, we assume Detext will be a decent balance between LSTM and BERT. We will evaluate the performance in the end.
   3. Reference:  
      Linkedin. (n.d.). Linkedin/detext. Retrieved October 18, 2020, from <https://github.com/linkedin/detext>

1. **Summary**

In the end, we will have a cross table like the following to evaluate the performance between these model.

|  |  |  |  |
| --- | --- | --- | --- |
|  | LSTM | BERT | Detext |
| Original data | Accuracy | Accuracy | Accuracy |
| Original data that down sample the number of clear data | Accuracy | Accuracy | Accuracy |
| Balance data | Accuracy | Accuracy | Accuracy |

The data and model may be adjusted according to our needs in the following weeks.

# Feature Engineering and Model Building

**Purpose:**For this part, we trained our baseline model, which is GloVe and BiLSTM, BERT and we decided to remove Detext.

**Feature Engineering:**

We did not perform feature engineering for our dataset, because we are using deep learning model.

**Model Building:**

1. **Baseline model - GloVe + BiLSTM**
   1. **Reason**

The reason why we chose LSTM model as our baseline model is that LSTM can deal with text and it is easy to train. Instead of simple LSTM, we use BiLSTM. BiLSTM is Bidirectional LSTM, enable additional training by traversing the input data twice from left to right, and from right to left again. Improve the performance compare with simple LSTM. Inaddidtion, we use GloVe word vectors to transfer out text to vector before input to the model.

* 1. **Library**

Python library: Pandas, numpy, matplotlib, keras, tensorflow

GloVe download: https://github.com/stanfordnlp/GloVe

* 1. **Dataset**
     1. Original data set
     2. Reduce the number of data that belongs to no class
     3. Balance data
  2. **Parameters**
     1. Batch size: 256
     2. Epoch: 50
     3. Validation split: 0.3
     4. Earlystopping: if the validation loss didn’t go down for 5 epochs.
     5. ReduceLROnPlateau: Reduce the learning rate if the validationa loss didn’t go down for 3 epochs
     6. Save the best model

1. **BERT**
   1. **Reason**

As the most renowned and powerful model for NLP, we need to apply it to

our classification tasks. Using BERT and Detext are transfer learning, transfer learning is a concept in deep learning where you take knowledge gained from one problem and apply it to a similar problem. It can save us time, data, and computational power. BERT, released by Google, that is so powerful that tackle all kinds of NLP problems and beat many competitions in recent years. We believed BERT can also have great performance on out problems.

* 1. **Library**

Python library: Pandas, numpy, matplotlib, torch, transformers

* 1. **Dataset**
     1. Original data set
     2. Reduce the number of data that belongs to no class
     3. Balance data
  2. **Parameters**
     1. Batch size: 8
     2. Epoch: 5

1. **DeText**

Detext has three advantages: first, it supports the art semantic understanding models (LSTM/BERT). Second, it provides a balance between efficiency and effectiveness. In our case, the cost of computing power will be less, and the accuracy of the outcome will be better than our baseline model. Third, it provides high flexibility on module configuration. However, this week, we decided to remove Detexrt for the following two reasons:

* 1. Because Detext is a package that already wrap up by the author, we need to modify our data into the input type that the Detext can accept. For Detext, it only accepts TFrecord, and it accept multi-class but not multi-label. Therefore, we need to use one-hot encoding to transform our data. The label of our data looks like:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Toxic | severe\_toxic | obscene | threat | insult | identity\_hate |
| 1 | 1 | 0 | 1 | 0 | 1 |

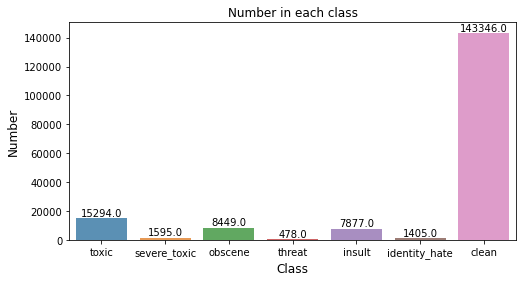
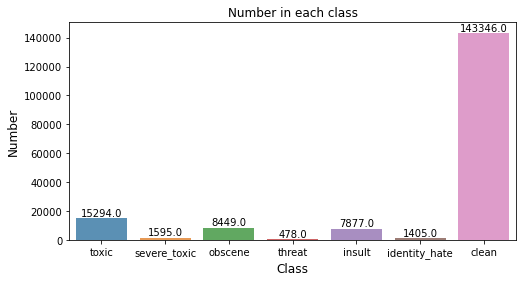
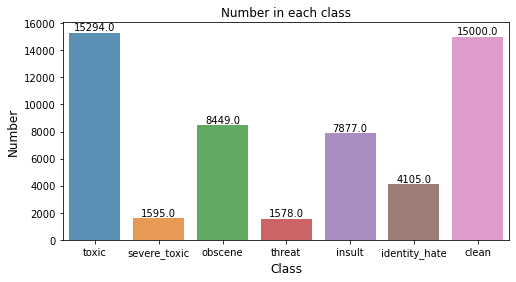
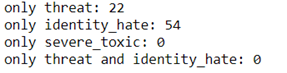
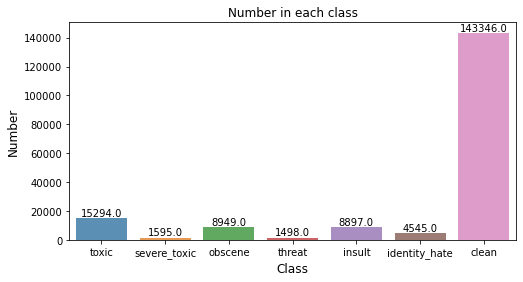
If we transfer it in to one-hot encoding, we will have 2\*2\*2\*2\*2\*2 =64 numbers of combination. There will be label from 0 to 63. However, this is not a good approach because some combination was not appearing in the data. Therefore, some value of labels will be zero. This situation will make the model more complicate and harder to train.

* 1. In the Detext, we can choose CNN, BERT, or LSTM as our model’s encoding layer. Because we already did LSTM and BERT, the outcome might be similar, and we already got well performance on LSTM and BERT. Hence, we decided to remove Detext.

# Model Validation and Visualization

**Purpose:**  For this part, we focused on training and testing different deep learning model on the modified dataset, and continuously learning how to build a ML-powered chrome extension.

**Training Dataset:**

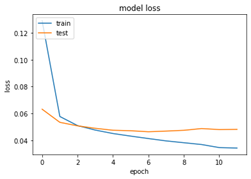
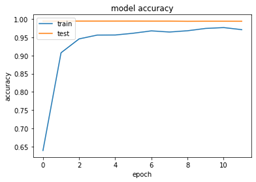
1. **Original Dataset**  
     
   This is the original dataset downloaded from Kaggle, as you can see this dataset is highly unbalanced, so in order to build a model with high accuracy, it is necessary to use different techniques (upsample and downsample)to address the issue.
2. **Lower the clean data**  
     
   In this case, we decided to get rid of some of the clean data, so there won’t be many 0 in the data
3. **Downsample and upsample**  
     
   Downsample the number of clean class and upsample the threat and identity.  
     
   Base on the data distribution we have, it is impossible to upsample and downsample to let each class have similar number, the only thing we can do is to let the class with more quantity drop a little, the class with less quantity rise a little
4. **Augmentation**  
   For image data, we have augmentation skills like scale, rotate, crop...etc. Here, we try to use some augmentation method for our text data. The method we used is to translate the text into another language and translate back into English. Therefore, some words and the order of the sentences may be different from the original text but the meaning will be the same.  
     
   We used this augmentation method to increase the number of threat and identity\_hate.

**Testing Data:**  
The testing data is also downloaded from Kaggle competition. However, because of the scoring way of the competition, some columns in the test data are labeled -1 indicating it was not used for scoring, like the following example:

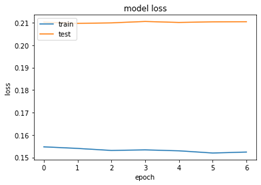
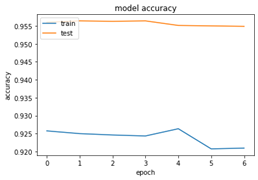


Therefore, we did not know the answer to these columns. We remove the text that all columns are labeled -1 and for the text that only some columns are -1, we will only count the columns that have answers when we calculate the accuracy.

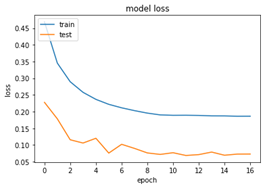
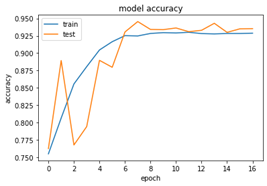
**Model** **Validation and Visualization:**

1. **Baseline Model –LSTM**
   1. **Original Dataset**  
      Model Accuracy and Loss (Batch size:512, Epoch:50, Early stop if the loss didn’t improve)   
      

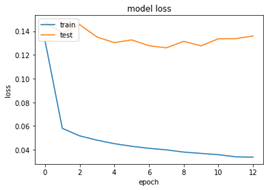
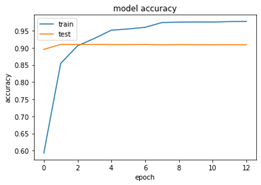
The accuracy of train and test (validation) can go up to near 100% and the loss keep improve to near 15 epochs.

* 1. **Lower the clean data dataset**  
     Model Accuracy and Loss (Batch size:512, Epoch:50, Early stop if the loss didn’t improve)  
     

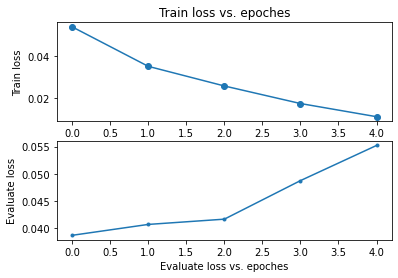
The accuracy of train hovering around 95.5% and test (validation) hovering around 92%. The loss also has the same problem. Which shows that lower the clean data is not a good method in this case.

* 1. **Downsample and upsample**  
     Model Accuracy and Loss(Batch size:512, Epoch:50, Early stop if the loss didn’t improve)  
     

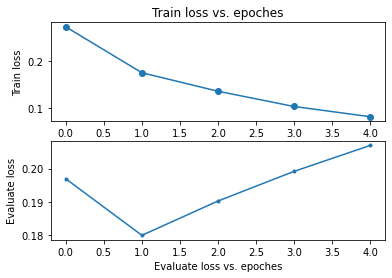
The accuracy of train and test (validation) improve to around 92.5% to 95%. The loss improves to epoch 16 and stop. Compared to the method of lower the clean data, Downsample and upsample perform better but the original data still perform well.

* 1. **Augmentation**  
     Model Accuracy and Loss(Batch size:512, Epoch:50, Early stop if the loss didn’t improve)  
     

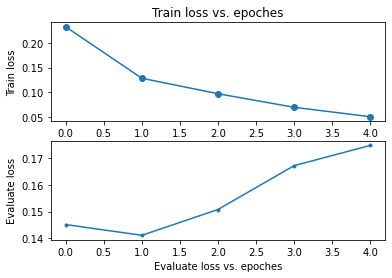
The accuracy of train keeps improve but the test (validation) keeps stay around 90%. And the loss has the same situation.

1. **BERT Model**
   1. **Original Dataset**  
      Batch size:8 Epoch: 5  
      

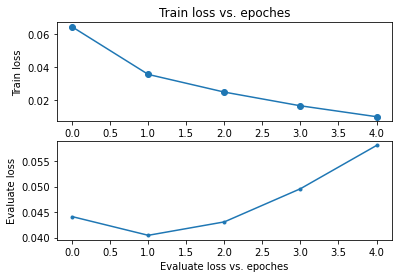
The loss of train improves but the loss of evaluate (validation) goes up.

* 1. **Lower the clean data**  
     Batch size:8 Epoch: 5  
     

The loss of train improves but the loss of evaluate (validation) goes up.

* 1. **Downsample and upsample**  
     Batch size:8 Epoch: 5  
     

The loss of train improves but the loss of evaluate (validation) goes up.

* 1. **Augmentation**  
     Batch size:8 Epoch: 5  
     

The loss of train improves but the loss of evaluate (validation) goes up.

**Evaluation**

We evaluated the model by predicting on testing dataset and see how many correct answers the model get. As we mention in the above, there are –1 in our dataset. Therefore, we did not count the –1. We only count the columns that we know the answer.

Accuracy = (Predict = 0 and True = 0)+(Predict = 1 and True = 1) / (Total number)

1. Baseline Model – LSTM
   1. Original Dataset  
      Accuracy of the testing data:96.9
   2. Lower the clean data  
      Accuracy of the testing data:95.1
   3. Downsample and upsample  
      Accuracy of the testing data:94.3
   4. Augmentation  
      Accuracy of the testing data:96.8
2. Bert Model
   1. Original Dataset  
      Accuracy of the testing data:100
   2. Lower the clean data  
      Accuracy of the testing data:100
   3. Downsample and upsample  
      Accuracy of the testing data:100
   4. Augmentation  
      Accuracy of the testing data:100

**Summary**

Accuracy on testing dataset

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Original Dataset | Lower the clean data | Downsample and upsample | Augmentation |
| LSTM | 96.9 | 95.1 | 94.3 | 96.8 |
| BERT | 100 | 100 | 100 | 100 |

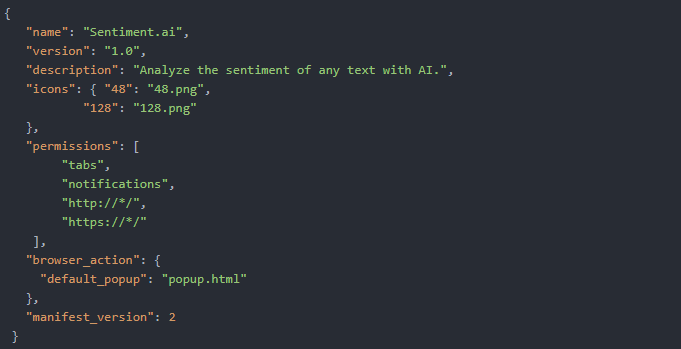
For LSTM model, the original dataset has the best performance on training process and Testing. This may relate to the distribution of the training dataset and testing dataset. Because they have similar composition, and we think this distribution meets the situation in the real scenario. Most of the content on the website are more neutral, toxic is less. If we train a model with more toxic data, then it may imply many contents on the internet are toxic.

Bert model performs relatively well compare our baseline model, and the testing accuracy on all dataset are 100%. This task is easy from BERT, the only problem is that it may be too big to run on Chrome extension. We will see in the following weeks.

**ML-Powered Chrome Extension**  
general steps for model deployment through machine learning powered chrome extension

* Train and test the model in Google Colab
* Upload the model to a public s3 buckets in aws
* Deploy the model as an API using cortext

How cortext configuration file looks like  


* Connect our chrome extension  
  What do the contents of manifest.json file look like:  
  

# Model Deployment

**Purpose:**

For this part, we mainly focused on deploy the machine learning model as a chrome extension based on a tutorial online.

**Implementation:**

* Using a Pre-Trained Model in TensorflowJS
  + In the pretrained model provide by TensorflowJS, it uses supervised learning to classify the text into different categories, such as identity attack, insult, obscene, severe toxicity, threat, etc.
* Creating Angular Elements
  + What are Angular Elements?  
    Angular elements are Angular components packaged as custom elements, a web standard for defining new HTML elements in a framework-agnostic way.
  + Two main library were created in the tutorial , first is a toxicity library, it contains the main code that creates a ShadowDOM which converts the text input and check the types of toxicity using the pretrained model in TensorflowJS. second library is used in the output application and it creates the elements for use by the chrome extension.
* Create a Chrome Extension
  + As the author stated that in the tutorial, chrome extensions are a manifest.json file that lists which JavaScript files should be included.When we are using Angular elements to builds the output application mentioned above, it warps the code in the dist folder. The manifest.json then imports the bundled output application and has a another script that classify the text area and export the output with percentage on the types of toxicity it belongs to.

# FINAL Report

**Purpose:**

Revise the report from week1 to the last week. And discuss the result and provide feedback.

**Discussion of results:**

1. The result of our model is beyond our expectation. The accuracy of our baseline model and BERT can detect toxic comment successfully, baseline model is around 95% and BERT is 100%.
2. The practical implication of our result in society is very wide. The content on the internet is very complicate, website full of dirty text, hater and troll cursing on each other. With the help of toxic comment detection, people can be aware of reading toxic content, parents can help their children filter the toxic text.
3. The limitation of BERT is that it takes too much computation power, we only train 5 epoch.
4. Future work included the following
   1. Replace the pre-train model on our Google Chrome Extension to the model we train, baseline model and BERT.
   2. Try to let the Google Chrome Extension highlight the toxic comments itself.
5. To improve the classifiers, the data needs to be updated continuously and tune the model. Because the language evolves fast and great, especially on the internet, many new slangs, abbrev, hashtag...etc.

**Feedback:**

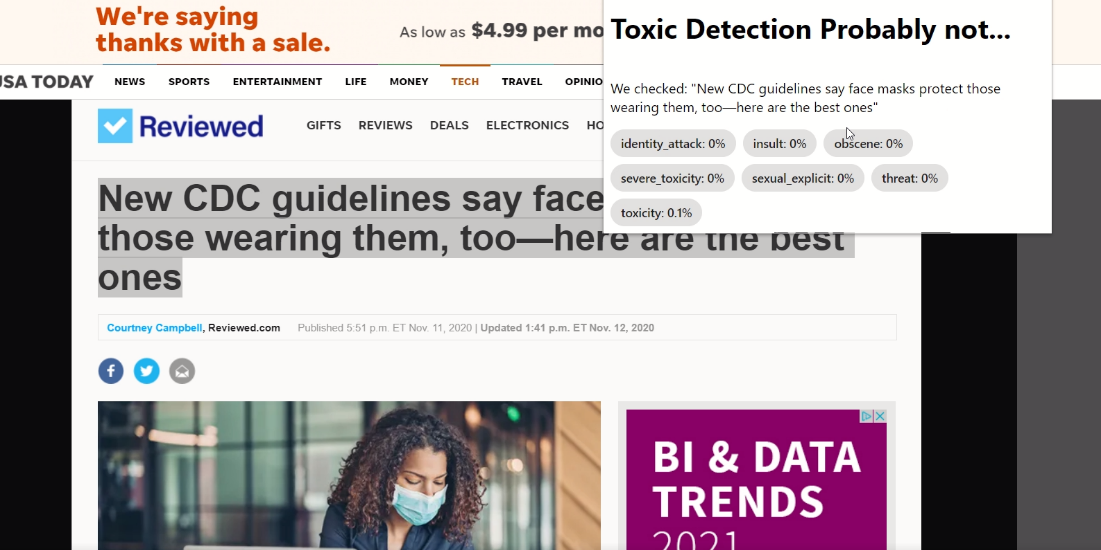
1. In our project, we chose to deploy our model to Google Chrome Extension, which gave us the most challenging time. We can’t say we are profession in JavaScript, Angular, and Google Chrome Extension, but at least we have some basic JavaScript, Angular, and Google Chrome Extension knowledge.
2. The Distinguished Speakers is very helpful, let us know more about what happened in the workplace. Some speaker’s lecture is too technical, hard for us to understand in short time, but it’s good to get more information.
3. Each group have different topic, if they have similar topic, we can compare their methodology. For the topic different from ours, we can learn from them and maybe we can adapt their method in the future when we face the similar problems.

# Project Presentataion & DeMO

**Purpose:**

To provide a demonstration of our team’s toxic comments detection system

**Demo:**



**Sentence which is not toxic at all**



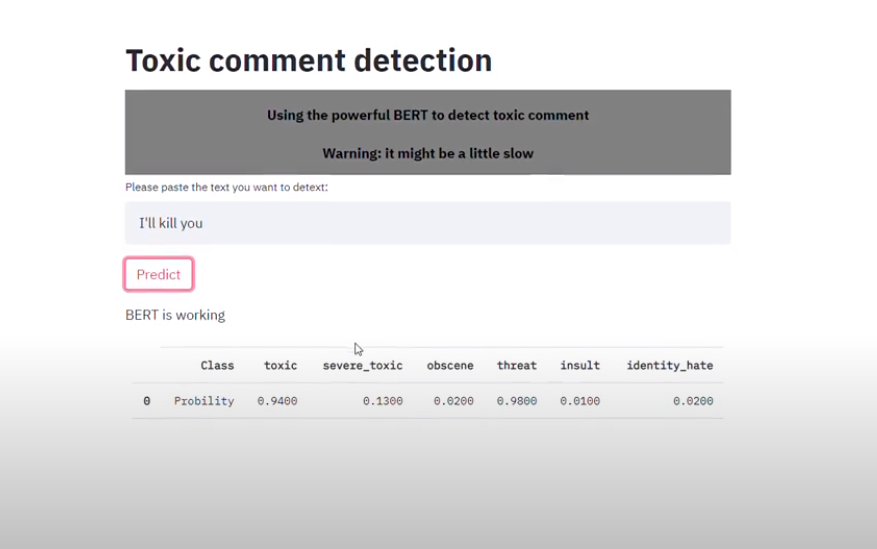
**Sentence which is toxic**

Video: <https://drive.google.com/file/d/1OawHzrep7kP7gmb2qzjD3DnRO_IwFyqF/view?usp=sharing>

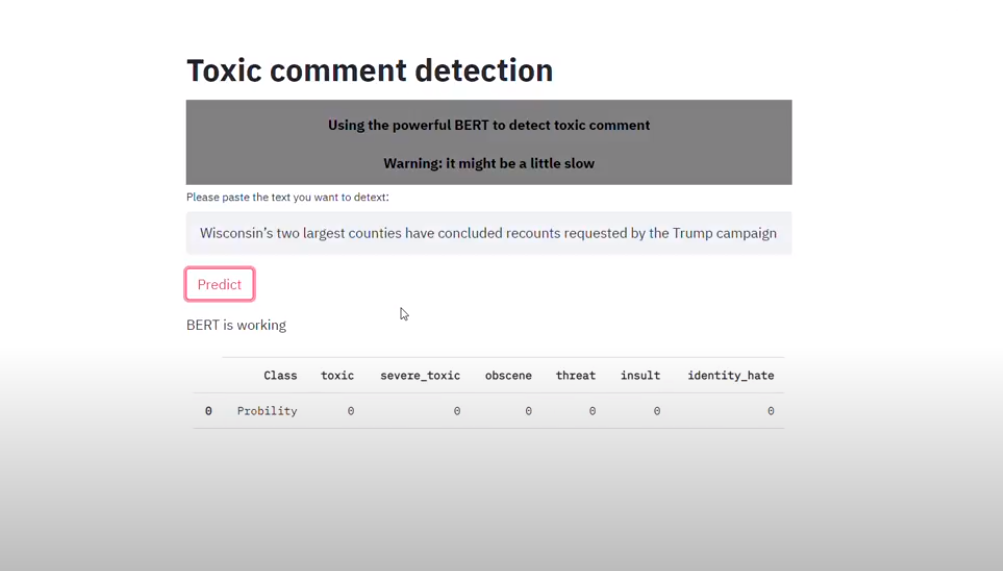
**Streamlit Demo**:

**Purpose**: we tried to use our own model in TensorflowJs, but there is no result coming out, we could not figure out the reasons, so we decided to try to implement it in streamlit

**Demo**:



**Sentence which is toxic**



**Sentence which is not toxic at all**

# Reference

Li, Hao et al. “Toxic Comment Detection and Classification.” (2019).

Badjatiya, Pinkesh et al. “Deep Learning for Hate Speech Detection in Tweets.” Proceedings of the 26th International Conference on World Wide Web Companion - WWW ’17 Companion (2017): n. pag. Crossref. Web.

Risch, Julian & Krestel, Ralf. (2020). Toxic Comment Detection in Online Discussions. 10.1007/978-981-15-1216-2\_4.

Mtjoycechang. “[NLP]Toxic Comment Classification.” Joyce Chang, 25 Feb. 2020, joyceds.com/2020/02/25/nlptoxic-comment-classification/.

Chakrabarty, Navoneel. A Machine Learning Approach to Comment Toxicity Classification. arxiv.org/ftp/arxiv/papers/1903/1903.06765.pdf.

V. Georgakopoulos, Spiros. Convolutional Neural Networks for Toxic Comment Classification. 2018, arxiv.org/pdf/1802.09957.pdf.

Bhangale, N. (2018, April 28). Toxic Comment Classification Models Comparison and Selection. Retrieved October 04, 2020, from <https://medium.com/@nehabhangale/toxic-comment-classification-models-comparison-and-selection-6c02add9d39f>

DeText: A deep NLP framework for intelligent text understanding. (n.d.). Retrieved October 04, 2020, from <https://engineering.linkedin.com/blog/2020/open-sourcing-detext>

Tang, A. (2020, July 26). An Intuitive Comparison of NLP Models (Neural Networks, RNN, CNN, LSTM). Retrieved October 04, 2020, from <https://medium.com/@audreyctang/an-intuitive-comparison-of-nlp-models-neural-networks-rnn-cnn-lstm-fc11bf452923>

Long short-term memory. (2020, September 27). Retrieved October 04, 2020, from <https://en.wikipedia.org/wiki/Long_short-term_memory>

Didi. (n.d.). Didi/delta. Retrieved October 04, 2020, from <https://github.com/didi/delta>

Purva HuilgolTrainee Data Scientist at Analytics Vidhya. Pursuing Masters in Data Science from the University of Mumbai. (2020, April 16). Pretrained Models For Text Classification: Deep Learning Models. Retrieved October 04, 2020, from <https://www.analyticsvidhya.com/blog/2020/03/6-pretrained-models-text-classification/>

Neuml. (n.d.). Neuml/txtai. Retrieved October 04, 2020, from <https://github.com/neuml/txtai>

Khalid, S., Says:, Y., Says:, S., Says:, J., Says:, P., & Says:, J. (2019, September 17). BERT Explained: A Complete Guide with Theory and Tutorial. Retrieved October 04, 2020, from <https://towardsml.com/2019/09/17/bert-explained-a-complete-guide-with-theory-and-tutorial/>

Cheng, R. (2020, July 22). BERT Text Classification Using Pytorch. Retrieved October 04, 2020, from <https://towardsdatascience.com/bert-text-classification-using-pytorch-723dfb8b6b5b>

Pennington, J. (n.d.). Retrieved October 18, 2020, from <https://nlp.stanford.edu/projects/glove/>

Ganegedara, T. (2020, October 18). Light on Math ML: Intuitive Guide to Understanding GloVe Embeddings. Retrieved October 18, 2020, from <https://towardsdatascience.com/light-on-math-ml-intuitive-guide-to-understanding-glove-embeddings-b13b4f19c010>

How Google's Open Source BERT Model is Enhancing NLP. (n.d.). Retrieved October 18, 2020, from <https://blueorange.digital/how-language-processing-is-being-enhanced-through-googles-open-source-bert-model/>

Linkedin. (n.d.). Linkedin/detext. Retrieved October 18, 2020, from <https://github.com/linkedin/detext/blob/master/notebooks/text_classification_demo.ipynb>

Martz, M. (2019, May 05). Are You a Jerk? Retrieved November 15, 2020, from <https://medium.com/@martzcodes/are-you-a-jerk-dea5d1b2d7cc>

TensorFlow. (2019, March 15). Text classification using TensorFlow.js: An example of detecting offensive language in browser. Retrieved November 15, 2020, from <https://medium.com/tensorflow/text-classification-using-tensorflow-js-an-example-of-detecting-offensive-language-in-browser-e2b94e3565ce>

Blog, G. (2020, July 24). Imbalanced Classification: Handling Imbalanced Data using Python. Retrieved October 18, 2020, from <https://www.analyticsvidhya.com/blog/2020/07/10-techniques-to-deal-with-class-imbalance-in-machine-learning/>

Pennington, J. (n.d.). Retrieved October 18, 2020, from <https://nlp.stanford.edu/projects/glove/>

Ganegedara, T. (2020, October 18). Light on Math ML: Intuitive Guide to Understanding GloVe Embeddings. Retrieved October 18, 2020, from <https://towardsdatascience.com/light-on-math-ml-intuitive-guide-to-understanding-glove-embeddings-b13b4f19c010>

How Google's Open Source BERT Model is Enhancing NLP. (n.d.). Retrieved October 18, 2020, from <https://blueorange.digital/how-language-processing-is-being-enhanced-through-googles-open-source-bert-model/>

Linkedin. (n.d.). Linkedin/detext. Retrieved October 18, 2020, from <https://github.com/linkedin/detext/blob/master/notebooks/text_classification_demo.ipynb>