Twitter Sentiment Analysis by Fendi Halim Tjoa

Part 1: Data Cleaning

For sentiment analysis dataset, the first and the easiest thing to do is to convert every characters on the tweet column into lowercase using **.str.lower()**. Doing so will keep the dataset consistent and can prevent sparsity issue when the computer cannot differentiate the same word being capitalized or not. Next step is the huge chunk of data cleaning using **Regex**. For both datasets, mentions and hashtags are being moved to its own column for EDA purpose on part 2. ***@[A-Za-z0-9_]+*** can be used to select the username (always start with **@**) and ***#[A-Za-z0-9_]+*** can be used to select hashtags and the characters followed by it. After that, the column now contains pure tweets and there is more freedom in removing non-alphanumerical keywords. The important thing that needs to be considered is the order of which thing need to be removed first. For example, URL, and HTML tags need to be removed first before removing independent backlash. If done in reverse, the regex wouldn't be able to identify the criteria of an URL or HTML tags. The list of regex can be inferred to the jupyter notebook function **clean_text_sa(text)**. Then, all the words in the column will be tokenized and have the stopwords removed. At this point, the tokenized words can be converted into its root word to reduce number of unique features using stemming or lemmatizing methods. For creating word cloud, lemmatized tokens are used because its root words are meant to be legible for human. For machine learning, stemming will be used because it is more efficient. The column that is ready to be used is under raw_sa[['combo']].

Canadian_election dataset also follows the same setting as above. However, there are extra things that need to be removed such as there is a letter b in front of every tweets, and there are many escaping characters such as '\' and '\n' (e.g., Figure 4). The updated function to removed those can be found on jupyter notebook function **clean_text_ce(text)**. The result can be depicted on Figure 5. The column that is ready to be used is under raw_ce[['stem_combo']].

```
'rt @immarygracee: i so love the aura of her room. very warm nung colors and it matched so well. ❤️ #pushawardskathniels https://t.co/vg
```

Figure 1: Unclean Tweet (sentiment_analysis File)

```
['love', 'aura', 'room', 'warm', 'nung', 'colors', 'matched']
```

Figure 2: Alphanumerical Tokenized Words Only with Stopwords Removed (SA File)

```
['pushawardskathniels', 'love', 'aura', 'room', 'warm', 'nung', 'color', 'match']
```

Figure 3: Clean Token Ready to be Used as Features with Stemmed Words (SA File)

```
'b"That so many people Others that this is a fake account is exactly why we need more exposure to people of different cultures, and more civility and humanity in politics. #elxn43 http
s://t.co/lIhpu6ookD"'
```

Figure 4: Unclean Tweet (canadian_election File)

```
['elxn43', 'peopl', 'fake', 'account', 'exposur', 'peopl', 'cultur', 'civil', 'human', 'polit']
```

Figure 5: Clean Token Ready to be Used as Features with Stemmed Words (CE File)

Part 2: Exploratory Analysis

A simple procedure to determine what party each tweet belongs to can be obtained by using **if** and **in**. For example, if there is a word trudeau, that tweet will belong to liberal party. The same apply to the other party such as conservative and NDP. List of the keywords for each party can be observed on jupyter notebook under Part 2 Canadian Election. However, there could be a problem when the tweet is talking about multiple parties. If the number of keyword counts of the parties are the same (e.g., trudeau mentioned twice and scheer also mentioned twice), then that tweet belongs to *mutual* party. If one of the keyword count is higher then the other (e.g., trudaeu mentioned twice and scheer is mentioned once), then that highest keyword will be the chosen party (that tweet belongs to liberal). If the tweet contains none of these keywords or maybe contains other party such as green party or Bloc Québécois, then that tweet belongs to *none* party. From observing figure 6, the tweet mostly contains nothing from these 3 major parties while conservative ranks 2nd, liberal 3rd and NDP 4th.

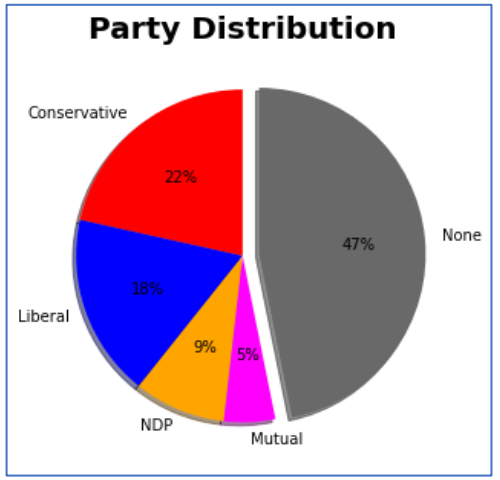

Figure 6: Pie Chart of Party Distribution

By looking at the word cloud on Figure 7, majority of the tweets of canadian_election dataset contains political terms such as elxn43, cdnpoli, canada, election, vote, etc. There are not many sentiment keywords such as happy, sad, great, wonderful, and so on. On figure 8, the ratio of negative and positive sentiment are very close with 47% negative tweet and 53% positive tweet.

On the contrary, word cloud on Figure 8 contains mostly sentiment keywords and they all look positive. Love, happy, good, great, and amazing. These words are humongous, and they stand out more than other keywords.

The pie charts corresponding to the dataset shows the same quick analysis from the wordcloud. Canadian_election dataset is more complex in determining whether the tweet is positive or not. There are 10 reasons why the tweet has negative sentiment. While on sentiment_analysis dataset, the keywords are more obvious.

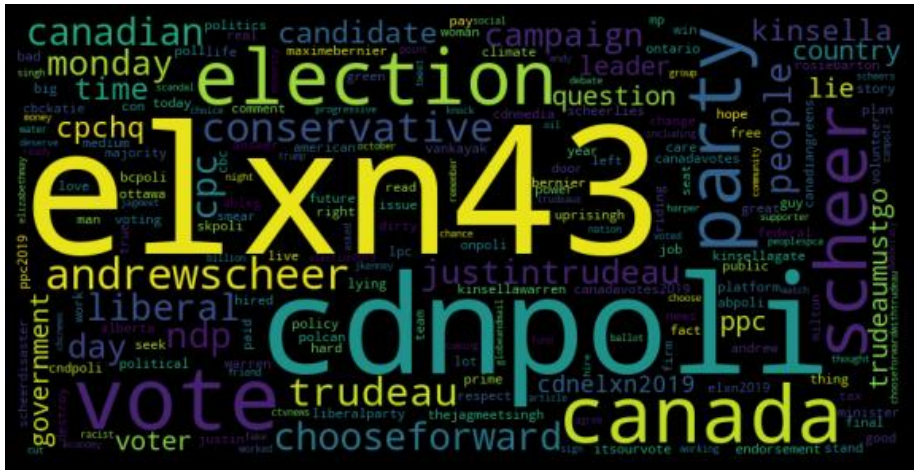

Figure 7: Word Cloud of Most Tweeted Words from canadian_election Dataset

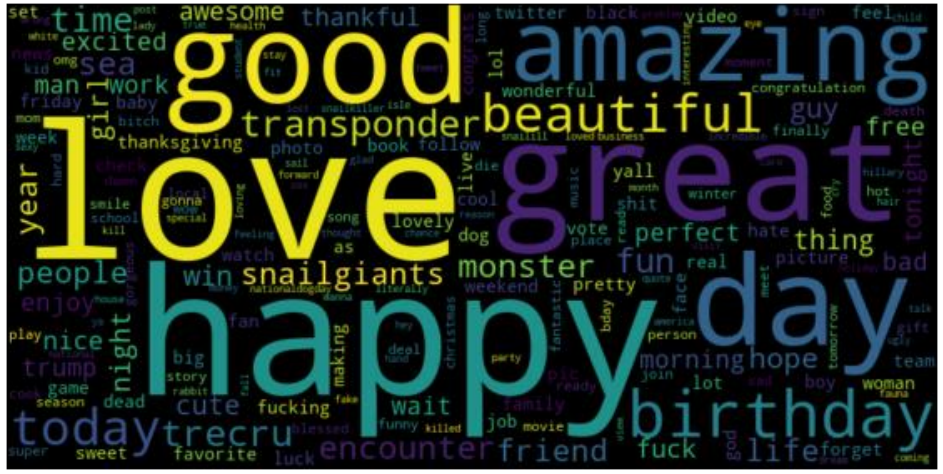

Figure 8: Word Cloud of Most Tweeted Words from sentiment_analysis Dataset

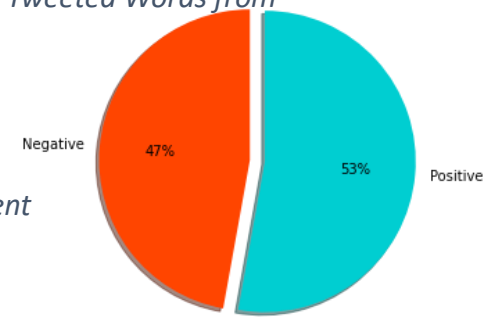

Figure 9: Pie Chart of Sentiment Distribution (CE File)

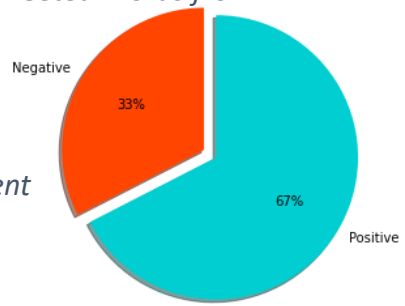

Figure 10: Pie Chart of Sentiment Distribution (SA File)

Part 3: Model Preparation and Implementation

Tokenized words under column raw_sa[['combo']] and raw_ce[['stem_combo']] will be converted back to string again. Then, train and test split can be performed. Train and test split must be performed before building the feature of Bag of Words (BoW) and TF-IDF. The reason is that train and test dataset should be treated independently. That way, there would be no data leakage. After splitting, the column will be converted into corpus (collection of documents) and then application of BoW and TF-IDF can be performed. **CountVectorizer()** is the function to create BoW and **TfidfVectorizer()** is the function to create TF-IDF. Since the data is humungous, with over 225,000 features, max_feature needs to be set up in order to prevent curse of dimensionality. Not only that, that many features are computationally costly and google colab and user's local computer are not able to handle such massive dimensions. Hence, the **max_feature** can be set up to 1000. When set to 1000, BoW and TF-IDF are taking the top 1000 most frequent words tweeted and therefore the model will have better generalization. After the features have been converted to BoW and TF-IDF, the X_train files will be in sparse matrix form.
The top 10 most frequent words tweeted is depicted on Figure 11.

love	63740
happi	51972
great	29533
good	27866
day	26206
amaz	24792
birthday	24197
beauti	16592
today	14419
transpond	14206

Figure 11: Top 10 Most Frequent Words Tweeted (Stemmed)

There are seven classification algorithms to be tested in order to get the best model in predicting sentiment analysis data. Some of the algorithms run fast and some of them run extremely slow such as KNN, decision tree, and random forest. In order to efficiently tune the hyperparameter, **GridSearchCV()** is extremely helpful as it also helps cross-validate the model. As a result, it will tell the mean best score and the best setting to get that score. When these models are used to predict the test set, most of the accuracy are around 80-90% range. However, XGBoost has the worst performance of 67.47%. The best model is to use **BoW LogisticRegression(C = 1, max_iter = 1000, penalty = 'l2', solver = 'newton-cg')** with training accuracy of 90.29%, it performs 90.32% on test set.

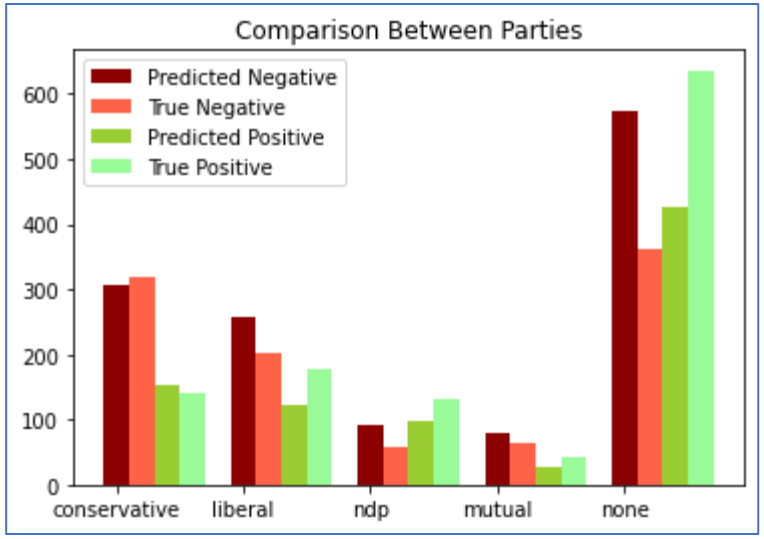

Figure 12: Distributions of Predicted Target vs True Target

Part 4: Model Implementation and Tuning

Using the best model above, the accuracy is 62.03% in predicting the sentiment on canadian_election dataset. The model above is not robust enough to predict the sentiment of canadian_election dataset. However, the prediction result can be used to analyze the overall sentiment of each party. If the similar color's (darker red and lighter red or darker green and lighter green) count are close to each other, means the model is successful in determining the overall sentiments for each party shown on Figure 12. For example, the total predicted negative and true negative on conservative are very close to each other. That means this model is good at identifying the overall negative sentiments for conservative. From these bars, it can be concluded that the model agrees there are more negative sentiments on conservative and liberal party while more positive are shown on NDP.

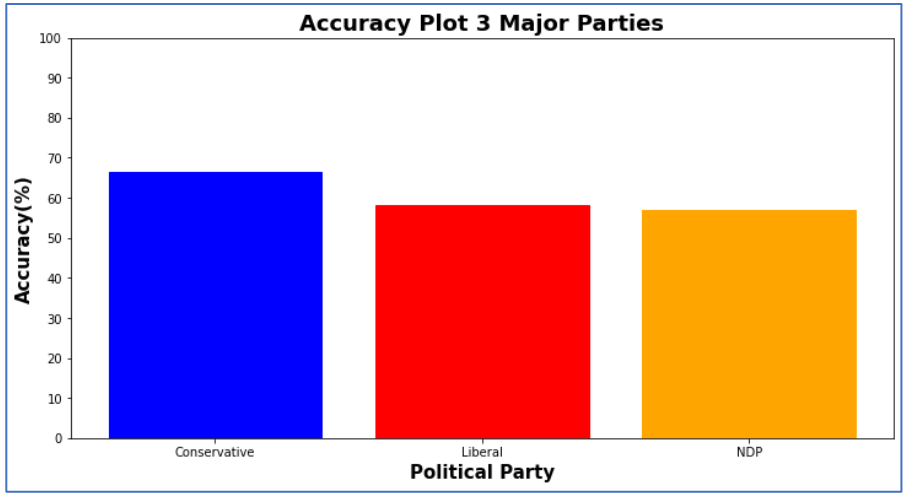

Figure 13: Prediction Accuracy of Each Party

Part 4: Model Implementation and Tuning (cont')

The next problem is to create a model that can predict the 10 negative sentiment reasons from canadian_election dataset. By looking at the pie chart on Figure 14 and the percentage of each class, there is a clear sign of imbalanced data. Moreover, there are 4 groups that has low counts such as Healthcare, Healthcare and Marijuana, Privilege, and Separation. Since these groups contribute 4% of the total counts in the target label, they can be moved to category *others*. The total classes have been reduced from 10 to 6. Theses labels need to be encoded to numerical.

In order to create the second model to predict negative reasons, the same procedure from model 1 can be followed. Train and test data will be split first before applying BoW and TF-IDF. This is an unbalanced multiclass problem and the method in determining the model's performance is to use macro F1 score. This scoring method will treat each class equally. The three classification algorithms chosen are **LogisticRegression()**, **LinearSVC()**, and **GaussianNB()**. These algorithms were chosen because they show promising results on the previous dataset. **GridSearchCV()** are also used across the models to tune hyperparameters and cross-validate. The best model out of these three is BoW Linear SVC with the test score of 52.80%

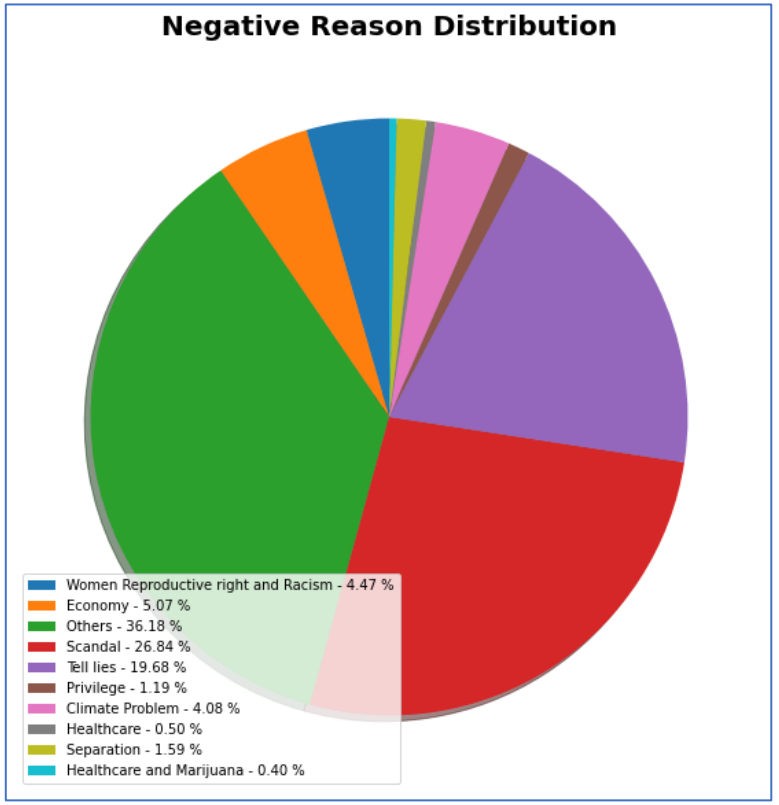

Figure 14: Negative Reason Distribution

Best BOW LogRes Parameter: {'C': 100.0, 'max_iter': 500, 'multi_class': 'multinomial', 'solver': 'newton-cg'}

Best BOW LogRes Training CV F1 Score: 47.76%

Best TFIDF LogRes Parameter: {'C': 68.66488450042998, 'max_iter': 500, 'multi_class': 'multinomial', 'solver': 'newton-cg'}

Best TFIDF LogRes Training CV F1 Score: 45.54%

Figure 15: Logistic Regression Hyperparameter Tuning Results

Best BOW LinearSVC Parameter: {'C': 0.29470517025518095, 'max_iter': 10000, 'penalty': 'l2'}

Best BOW LinearSVC Training CV F1 Score: 52.80%

Best TFIDF LinearSVC Parameter: {'C': 2.329951810515372, 'max_iter': 10000, 'penalty': 'l2'}

Best TFIDF LinearSVC Training CV F1 Score: 51.19%

Figure 16: LinearSVC Hyperparameter Tuning Results

Best BOW GaussianNB Parameter: {'var_smoothing': 0.657933224657568}

Best BOW GaussianNB Training CV F1 Score: 50.30%

Best TFIDF GaussianNB Parameter: {'var_smoothing': 0.006579332246575682}

Best TFIDF GaussianNB Training CV F1 Score: 38.36%

Figure 17: GaussianNB Hyperparameter Tuning Results

Part 5: Results

First Model Analysis:

On the sentiment analysis dataset, both training and testing accuracy are around 90% using Logistic Regression (with proper hyperparameter tuning).
The result of using the same model to predict the sentiment of canadian_election is mediocre. The result of the accuracy is 62.03%.
Other than using this model to predict the sentiment of each tweet, this model also can be used to analyze general sentiment towards each party. The barplot depicted on Figure 12 tells that the model is good at identifying how each party is viewed in the public eye based on the sentiment value. The distribution of the counts of prediction are close to the distribution of true counts. Conservative party and Liberal party are viewed negatively in the public eye while NDP is leaning towards positive.

The possible reason the accuracy is not as high as the training model is because most of the words in the canadian_election dataset contain too many political terms and less of sentiment. By comparing the wordclouds on Figure 7 and Figure 8, clearly there are more political terms than sentiment adjectives in generic tweets dataset. The canadian_elections features were also transformed according to **CountVectorizer()** from generic tweets dataset. Therefore, there could be loss of important features from canadian_election dataset.

To improve the accuracy of the model, the model needs to be trained in a dataset that contains more of the political sentiment keywords. Keywords in sentiment analysis dataset are very generic (love, good, wonderful, sad,...) while sentiment keywords in canadian election dataset are not really expressed in keywords (forward, stand, unpredictable, support, vote, ...)

Second Model Analysis:

On the canadian_election dataset, the training Macro F1 Score using **LinearSVC** is 52.80% and the test score is 54.09%.

The second model is more difficult to configure and predict because the datasets contain unbalanced target variable and they are multiclass. Initially, there are 10 negative reasons of why the tweets have negative sentiment. When analyzing the distribution of these classes, there are 4 classes that contain less then 4% of the dataset. For example, there is only 5 samples for class Healthcare, and 12 samples for class Privilege. Because of this, these 4 classes can be moved to class Others. As a result, there are 6 classes remaining.

There are many reasons that the F1 score is not that high:

1. The curse of dimensionality. Here we have more features than the samples (704x2994). When these happen, it is very difficult for our model to generalize. The max_features of countvectorizer or tfidfvectorizer can be reduced to around 700 (to match the sample size) but there is a potential loss of information.
2. There are not enough good keywords that can be identified by the models. Exactly like depicted on Figure 7, most of the tweets contain political party keywords and less words regarding negative reasons. Negative reasons do not specify if the tweet is negative based on political parties.
3. There could be noises in the feature that is not removed. For instance, keywords such as names, typos, and unimportant hashtags.

One way to improve this accuracy is to increase the number of samples just like the sentiment analysis dataset. By having more samples like sentiment analysis dataset, the features can be reduced to only the necessary ones. Therefore, the model can generalize better and will perform good in both training and testing.