Introduction:

Sentiment Analysis is one of the most researched problems in NLP. It involves analyzing the given data and predicting the sentiment (opinion) of it. It has many names — Opinion Mining, Sentiment Mining, and Subjectivity Analysis. The data could be text, images, tabular etc. SA helps companies in making good decisions. It has a lot of applications: chatbot, product reviews, aspect reviews.

There are many types of sentiment analysis: Document Level Sentiment Analysis, Sentence level sentiment analysis, Aspect based Sentiment Analysis, and Fine-grained Sentiment Analysis etc.

For this project, we focus on Sentence level sentiment analysis. It involves understanding the sentence word by word to obtain the semantic meaning of a sentence. Traditional machine learning and deep learning methods, like SVM, CNN, RNN, treat the sentences as "bag-of-words" without considering the positions of the words in the sentences. However, the sentiment of a sentence may change if we change the positions of the words. My final model Roberta Transformer also considers the positions of the words using positional word embeddings along with the contextual word embeddings that are pretrained on other data, but tuned on the current training data. Transformers use multiple attention heads (my model BERT has 12) and each of them focus on different positions in a sentence and thus capture different aspects of the sentence like semantics and syntactics etc. Hence, the decoder will get a representation that is an accumulation of representations of different aspects and thus if given a good representation of the sentence, it can predict the sentiment well.

Problem Definition:

Given a tweets dataset, we must predict whether a tweet has positive, negative, or neutral sentiment. It is like the text classification problem; the given sentence (tweet) must be preprocessed and should be classified as one of the three classes.

Solution Pipeline:

a) Data preprocessing
b) Data Analysis (class distribution)
c) Model Building and hyperparameter tuning

Data Preprocessing:

Note: (I followed the preprocessing steps and used some of the code mentioned in https://github.com/rrajath/tweet-sentiment-classifier/blob/master/src/PreprocessData.py)

Steps involved in DP:

a. Removal of rows with completely missing tweets
b. Converted all the characters to lower case
c. Expanded abbreviations and contractions:
   Abbreviations like lol (laugh out loud) and contractions like isn't (is not) convey more meaning when expanded and might change the sentiment.
d. Removal of URLs:
   URLs were replaced by a single space as they do not change the sentiment of a tweet significantly. They are of the form: https://www.linkedin.com/
e. Removal of Html tags, hash, punctuation, and special characters:
   Hash (#), Html tags (<,>,), punctuation (;,"), and special characters (&,%,_) were also replaced by single space as they do not contribute much to the sentiment of a tweet
f. Removal of usernames:

Usernames that are referred to in the tweets do not add to the sentiment hence they are parsed and replaced by blank spaces

g. Removal of words with more than one-digit, non-Ascii characters and new line characters:
Words with more than one digit are mostly usernames which do not alter the sentiment much. Hence, they were replaced by blank spaces.

h. Removal of additional white spaces:
Stripped off extra white spaces

i. Removal of two or more repetitions of a character:
Ex: “Gooood” becomes “Good”

j. Split camel case words:
Camel case words are two or more words that got merged.
Ex: voteforobama becomes vote for obama

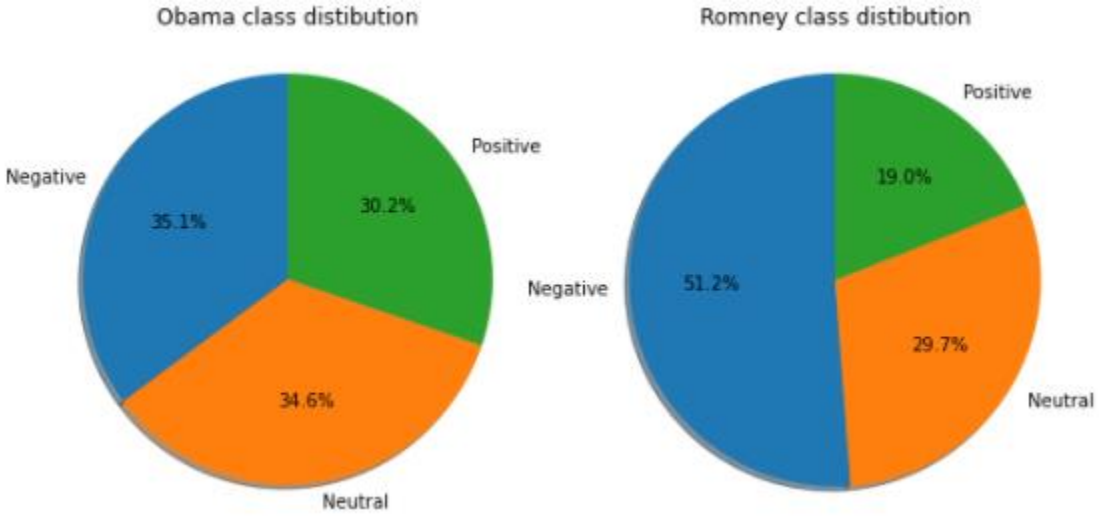
k. Removal of stop words:
Stop words (like is, are) do not change the sentiment they just increase the size of the vocabulary which causes overhead. I extracted a stop words list from the NLTK library and used it to remove the stop words in my tweets.

l. Replaced some weird words:
Words like “hahaha…..haha” and “lolo….lol” were replaced with haha and lol respectively.

m. Lemmatized the words:
Finally, converted all the words into their base forms
Ex: ‘Winning” becomes “Win”

Data Analysis: (forgot to include the DA ipynb file in the zip file)

Through data analysis, I found that the classes are imbalanced for both datasets particularly for the Romney dataset. Hence, I decided to use cross-validation F1 positive score and accuracy to tune the hyperparameters.



Techniques/Models Tried:

a) Machine Learning models
b) Deep Learning Models

Note: I used the Hyperopt library for hyper-parameter optimization (tuning).

Machine Learning Models: (used dimensionality reduction technique PCA for the time consuming models)

All the ML models were trained on both the TF-IDF datasets of the original datasets and PCA datasets of the TF-IDF datasets, with the no of principal components as a hyper-parameter. For models that take a lot of

time like RBF SVM, I used only PCA datasets to build models. I tuned the hyper-parameters of each model with the cross-validation F1 positive score and accuracy.

I started with basic ML models and went on to ensemble models, such as Random Forest, and Light gradient boosting.

1. Logistic Regression (hyperparameters: C)
2. SVC – Support Vector Classifier (hyperparameters: C)
3. KNN – (hyperparameters: k)
4. Naïve Bayes (NB)
5. Random Forest (hyperparameters: no of estimators, max_depth, min_samples_split, min_samples_leaf)
6. Light Gradient Boosting (hyperparameters: boosting_type, n_estimators, max_depth, learning_rate, subsample, min_split_gain)

Deep Learning Models: (used Google Colab Pro with fastest GPU Tesla V100)

For RNNs and CNN models, I used Stanford glove twitter word embeddings of 200 dimensions. For transformer models, I used the 'cardiffnlp/twitter-roberta-base-sentiment' pretrained model that has contextual word embeddings.

1. CNN (hyperparameters: epochs, batch size, maximum sentence length, learning rate)
2. RNNs - LSTM and GRU (hyperparameters: epochs, batch size, maximum sentence length, learning rate, embedding dimension
3. Roberta Transformer (hyperparameters: batch size, maximum sentence length)

Results:

Note: all the scores below are the best cross-validation scores on the given data except for LSTM and GRU which are just the best validation scores.

For Obama tweets:

S. No	Model	Accuracy	F1 Positive	F1 Neutral	F1 Negative
1	Logistic Regression	0.60	0.61	0.55	0.63
2	Logistic Regression PCA	0.57	0.60	0.52	0.59
3	SVC PCA	0.58	0.61	0.54	0.61
4	KNN	0.51	0.57	0.41	0.54
5	KNN PCA	0.48	0.51	0.47	0.45
6	Naïve Bayes	0.48	0.54	0.36	0.50
7	Naïve Bayes PCA	0.48	0.47	0.45	0.53
8	Random Forest	0.57	0.59	0.53	0.58
9	Random Forest PCA	0.54	0.56	0.50	0.57
10	Light GB	0.55	0.57	0.51	0.56
11	Light GB PCA	0.55	0.58	0.51	0.57

S. No	Model	Accuracy	F1 Positive	F1 Neutral	F1 Negative
1	LSTM	0.56	0.61	0.45	0.61
2	GRU	0.57	0.59	0.49	0.62
3	CNN	0.54	0.57	0.44	0.59
4	Roberta Transformer	**0.64**	**0.66**	**0.58**	**0.67**

For Romney tweets:

S. No	Model	Accuracy	F1 Positive	F1 Neutral	F1 Negative
1	Logistic Regression	0.55	0.50	0.43	0.64
2	Logistic Regression PCA	0.54	0.50	0.44	0.63
3	SVC PCA	0.56	0.49	0.44	0.66
4	KNN	0.53	0.43	0.36	0.65
5	KNN PCA	0.51	0.36	0.4	0.62
6	Naïve Bayes	0.4	0.37	0.36	0.45
7	Naïve Bayes PCA	0.45	0.35	0.44	0.50
8	Random Forest	0.57	0.3	0.26	0.71
9	Random Forest PCA	0.56	0.33	0.32	0.70
10	Light GB	0.51	0.45	0.45	0.58
11	Light GB PCA	0.53	0.46	0.44	0.62

S. No	Model	Accuracy	F1 Positive	F1 Neutral	F1 Negative
1	LSTM	0.60	0.43	0.35	0.74
2	GRU	0.57	0.51	0.44	0.67
3	CNN	0.54	0.45	0.27	0.67
4	Roberta Transformer	**0.64**	**0.54**	**0.49**	**0.74**

Conclusion:

I selected an ensemble of the best transformers model and the best SVC model (built on the PCA TFIDF datasets) as the final model for both Obama and Romney tweets. Logistic Regression also produced good results. But all other ML models, such as Naïve Bayes, KNN, Random Forest, Light GB did not perform well for both datasets particularly for the Romney for which classes are quite imbalanced. Deep Learning models, such as CNN's and RNNs performed slightly better than the ML models but Transformer models gave the best results which use contextual word embeddings, that are pretrained on other datasets, instead of traditional word embeddings used by CNNs and RNNs.

Lessons Learned:

Transfer Learning is the reason for the better performance of the transformers. It is again proven with this project that the word embeddings learned from the given data are not sufficient enough to produce good results i.e. they cannot capture the correct meanings of the words if only trained on the given training data. Contextual word embeddings that are pretrained on other data and tuned on the given training data were able to capture the contextual meanings of the words, which resulted in better accuracy and the F1 scores.

Models built on the traditional word embeddings come under single-task learning or closed-world learning whereas Transfer learning comes under multi-task learning and open-world learning. Multi-task learning recently overthrown many current state-of-the-art models in NLP. (https://www.aclweb.org/anthology/P19-1441.pdf).

One more thing to note is if the classes are highly imbalanced or skewed then even the best models like Transformers cannot capture the right patterns in the data, this is evident in the accuracy and F1 scores of the Roberta transformer in the case of the Romney dataset. We can try sampling techniques like Under sampling and Oversampling (smote). In case of under-sampling, the models might underfit the data and they can overfit the data when over-sampling is used.