Comparative Analysis of Machine Learning Classifiers for Text Classification

Kaggle Team name: pied piper

Keval Pipalia (20241329) *and Divya Nagpal (20241274) †

Professional Masters in Machine Learning, MILA,
University of Montreal, Quebec, Canada.

Submitted on: 18th December 2022

1 INTRODUCTION

The field of natural language processing research is moving in an important direction, and one of the hottest research areas right now is sentiment classification. In this paper, we try to learn accurate models to classify the Text Classification Challenge Dataset. We experiment with several classification algorithms on a dataset of 1 million text sentences belonging to 3 sentimental categories. LSTM is a type of recurrent neural network. It is one of the representative algorithms of deep learning. A majority voting using the 5 different models including LSTM, complement Naive Bayes, Multinomial Naive Bayes, SVM and Logistic Regression performed the best on our validation test set. The results can be reproduced by using the code at GitHub.

This report is in addition to the Kaggle Competition for Text Classification Challenge in IFT6390 Fall 2022.

1.1 Problem

The task is to classify text sentences belonging to 3 sentimental categories i.e. positive, neutral or negative as shown in Figure 1. Formally, this is called the problem of text classification. It aims at identifying to which of a set of categories a new text document belongs, on the basis of a training set of data containing text documents whose category memberships is known. We aim to learn patterns across these categories which differentiate them the most.

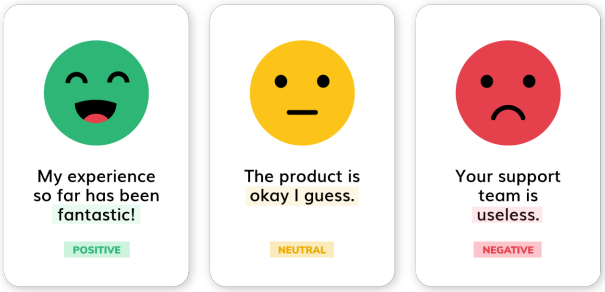

Figure 1: Sentiment analysis illustration

1.2 Dataset

The Dataset represents over 1 millions texts which are annotated as negative, positive and neutral. The training and held-out test set for this data consists of 1,030,487 samples and 556,129 samples respectively. The task is to classify text sentences belonging to 3 sentimental categories i.e. positive, neutral or negative.

The text documents may have frequent grammatical mistakes as the source or distribution that generated the data

*Electronic address: keval.pipalia@mila.quebec
†Electronic address: divya.nagpal@umontreal.ca

is unknown which makes the problem even harder. Furthermore, each text document can significantly vary in length. For instance, the mean number of words per document across all the categories is 52 whereas the corresponding median values is 30 words, indicating the presence of outliers in the data. We will use 20% of training data for validation and later will include them and retrain the model with learned parameters to get the maximum accuracy on kaggle test dataset.

2 FEATURE DESIGN

2.1 Preprocessing

Preprocessing the data is a crucial step in any text classification pipeline since human language is intrinsically complicated, making it impossible for a straightforward machine learning algorithm to comprehend the intricacy of unprocessed language text.

2.1.1 Stopwords Removal

Stop Words are the words that do not have any characteristics that most clearly distinguish the classes from one another. We used NLTK library to remove the stopwords from the documents. But apparently we found that stopwords add information to sentiment analysis classification. Hence we are not going to remove the stopwords [1].

2.1.2 STEMMING AND LEMMATIZATION

To reduce the complexity of the data, we perform an additional preprocessing step of trying to reduce each word to its root or lemma, called stemming and lemmatization respectively. Stemming is much simpler as compared to lemmatization as lemmatization depends on correctly identifying the intended part-of-speech and meaning of a word in a sentence, as well as within the larger context surrounding that sentence. We choose to perform lemmatization as it is known to perform better than stemming in practice.

2.2 Advance text cleaning pipeline

We experimented with many text-cleaning techniques, including stemming, lemmatizing, stopword removal, punctuation removal, etc. Figure 2 illustrates the comparative study of different techniques along with their accuracy scores on the validation set that were leveraged. All of them were compared and combined into a single preprocessing workflow. The pipeline comprises of lemmatizing, replacement of contractions, lowering of alphabets, and removal of some punctuation.

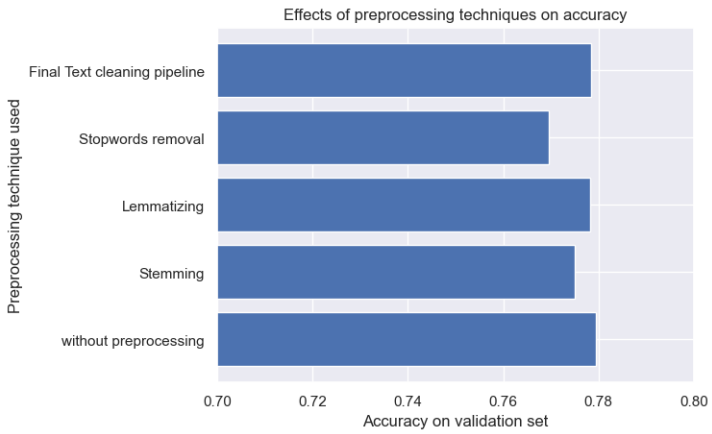

Figure 2: Comparison of different text cleaning approaches. (values refer to accuracy on validation data with naive bayes bag of words classifier)

2.3 Feature Transformation

Textual data must always be transformed to its numerical representation for machine learning algorithms to function, even after comprehensive preparation of the data. There are several methods for doing this. To determine which one performs the best on our dataset, we test the most popular options. The effectiveness of these techniques is compared after a description of them.

2.3.1 Bag of Words

To make it simpler for our models to learn the language, we assume that the order of the words in a sentence does not affects the class predictions. This is called a Bag-of-words model wherein each document vector contains the count of each word in the vocabulary.

2.3.2 TF-IDF

TF-IDF (term frequency-inverse document frequency) 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 [2].

$$TFIDF(t) = TF(t) * IDF(t)$$

$$TF(t) = \frac{\text{Number of times term t appears in a document}}{\text{Total number of terms in the document}}$$

$$IDF(t) = log_e \frac{\text{Total number of documents}}{\text{Number of documents with term t in it}}$$

2.3.3 Word Embeddings

Word embeddings are basically a form of word representation that bridges the human understanding of language to that of a machine. They have learned representations of text in an n-dimensional space where words that have the same meaning have a similar representation. Meaning that two similar words are represented by almost similar vectors that are very closely placed in a vector space. These are essential for solving most Natural language processing problems [3].

2.3.4 Word2Vec

The word2vec tool takes a text corpus as input and produces the word vectors as output. It first constructs a vocabulary from the training text data and then learns vector representation of words. The resulting word vector file can be used as features in many natural language processing and machine learning applications.

A simple way to investigate the learned representations is to find the closest words for a user-specified word. The distance tool serves that purpose.

2.3.5 String Kernels

We will be using custom Word2Vec function kernel as a string kernel. It has been trained on given dataset and does not use pretrained weights.

3 ALGORITHMS

We experiment with several different classification algorithms to get the best prediction accuracies on our dataset.

3.1 Naive Bayes

The Naive Bayes classifier, which is the most often used for text, serves as the starting point for our search for the best classification method (NBC). The great performance, high scalability, and simplicity of training of the probabilistic classifier NBC have made it popular. Text data is often high-dimensional due to the large number of possible word combinations. Although it is predicated on the independence of words, it appears to function quite well in actual use. With Bag of Words, this algorithm was run with an accuracy of 0.7784.

3.1.1 Hyperparameters tuning

Another common technique to further improve the accuracy of a NBC is to smooth the text data using Laplace Smoothing, also known as Additive Smoothing. Additive smoothing allows the assignment of non-zero probabilities to words which do not occur in the sample. Recent studies have proven that additive smoothing is more effective than other probability smoothing methods in several retrieval tasks. We incorporated Laplace Smoothing in our implementation of the NBC and applied it on the training set. Using grid search, we found out that the best value of the hyperparam α to be 2.0.

3.1.2 Complement Naive Bayes

Complement Naive Bayes is particularly suited to work with imbalanced datasets. In complement Naive Bayes, instead of calculating the probability of an item belonging to a certain class, we calculate the probability of the item belonging to all the classes. This is the literal meaning of the word, complement and hence is called Complement Naive Bayes. After tuning it we received the accuracy of 0.7724. This is slightly less than Naive Bayes but as mentioned this model is unbiased to the underrepresented neutral class [4].

3.2 Support Vector Machine

Since there can be many hyperplanes separating the classes in our data, one reasonable choice as the best hyperplane is the one that represents the largest separation, or margin, between any two classes. A support vector machine classifier chooses a hyperplane so that the distance from it to the nearest data point on each side is maximized.

3.2.1 Hyperparameters tuning

We could also apply the kernel trick to learn a non-linear hyperplane but since our data is already very high-dimensional we went ahead with a linear classifier. We chose regularization parameter C by using grid search.

3.3 LOGISTIC REGRESSION

Logistic regression is a Supervised Machine Learning algorithm that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. for example, Softmax regression is a generalization of logistic regression. Logistic regression is used for binary classification, while Softmax regression is used for multi-class classification.

3.3.1 Hyperparameters tuning

Since there are many hyperparameters we can tune with this algorithm heuristic (grid search) can take very long time so we chose to use bayesian search instead. We found values of tolerace: 2.2+e4 , solver: 'saga', max iterations: 300 and regularization parameter C: 0.5

The implementation for Logistic Regression can be found in the github repo. We have used stochastic gradient descent with logistic(softmax) function to train the algorithm. We received the accuracy of 0.8249 on test set and 0.8254 with the Kaggle.

3.4 XGBOOST

XGBoost and Gradient Boosting Machines (GBMs) are both ensemble tree methods that apply the principle of boosting weak learners (CARTs generally) using the gradient descent architecture. However, XGBoost improves upon the base GBM framework through systems optimization and algorithmic enhancements. XGBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. We received accuracy of 0.7686.

3.5 NEURAL NETWORK

A neural network is a series of algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates.

3.6 RECURRENT NEURAL NETWORK

Ordinary feedforward neural networks are only meant for data points that are independent of each other. However, if we have data in a sequence such that one data point depends upon the previous data point, we need to modify the neural network to incorporate the dependencies between these data points. RNNs have the concept of "memory" that helps them store the states or information of previous inputs to generate the next output of the sequence [5].

3.6.1 LSTM

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network capable of learning order dependence in sequence prediction problems. This neural network has neurons and synapses that transmit the weighted sums of the outputs from one layer as the inputs of the next layer. A backpropagation algorithm will move backwards through this algorithm and update the weights of each neuron in response to he cost function computed at each epoch of its training stage.

By contrast, here is what an LSTM looks like:

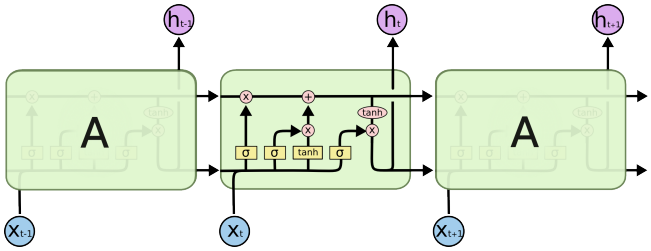

Figure 3 : LSTM architechture

The complete architecture of the model is given in Figure 4.

All layers will use the ReLU activation function except the last layer which is using a softmax activation to classify.

We achieved accuracy of 83.48% on test set after training on 100 iterations. (early stopping at 15)

3.7 ENSEMBLED MODEL

Although we have received a very good accuracy with Logistic Regression, we wanted to better classify the misclassified texts, but after certain epochs the LSTM model started

overfitting and there were no improvements w.r.t. validation data. So we used one simple yet effective approach named 'ensembling'.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 30, 100)           1000000

bidirectional (Bidirectiona  (None, 30, 128)           84480
l)

dropout (Dropout)            (None, 30, 128)           0

bidirectional_1 (Bidirectio  (None, 64)                41216
nal)

dropout_1 (Dropout)          (None, 64)                0

dense (Dense)                (None, 64)                4160

dropout_2 (Dropout)          (None, 64)                0

dense_1 (Dense)              (None, 3)                 195

=================================================================
Total params: 1,130,051
Trainable params: 130,051
Non-trainable params: 1,000,000
_________________________________________________________________
```

Figure 4: LSTM architechture

Ensembling is the process of combining multiple learning algorithms to obtain their collective performance i.e., to improve the performance of existing models by combining several models thus resulting in one reliable model. Ensemble Learning uses multiple models for prediction and gives class that represents most of the votes from classifiers' predictions.

We used best 5 models with a voting classifier to get the most accurate classifier. We received accuracy of 82.65% with the test set.

4 METHODOLOGY

A few approaches were used to find an optimal model. A few of them are discussed below.

4.1 USAGE OF TRAIN/TEST DATA

The training and held-out test set for this data consists of 1,030,487 samples and 556,129 samples respectively. We will use 20% of training data for validation. All hyperparameters and models were compared after training and testing with these splits. After deciding the optimal values, models were retrained on complete dataset for better accuracy on kaggle test set.

4.2 DATA IMBALANCE

A classification data set with skewed class proportions is called imbalanced. Classes that make up a large proportion of the data set are called majority classes. Those that make up a smaller proportion are minority classes. Looking at out given training data we can see in the Figure 5, we have 'neutral' class that is minority class. There are only 80 samples of neutral class compared to 500,000 of other positive and negative classes.

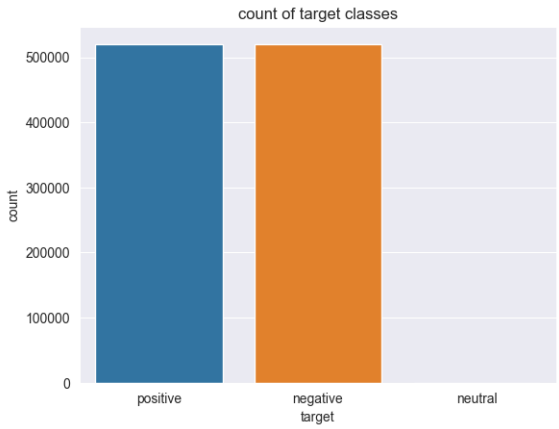

Figure 5: Label Distribution

4.2.1 SMOTE

SMOTE (synthetic minority oversampling technique) is an oversampling technique where the synthetic samples are generated for the minority class. It aims to balance class distribution by randomly increasing minority class examples by replicating them. SMOTE synthesises new minority instances between existing minority instances. It generates the virtual training records by linear interpolation for the minority class. These synthetic training records are generated by randomly selecting one or more of the k-nearest neighbors for each example in the minority class.

4.3 REGULARIZATION

Regularization is a technique used to reduce the errors by fitting the function appropriately on the given training set and avoid overfitting. In RNN, we have set dropout layer or we can freeze some parameters, that way we can get similar results as regularization.

4.3.1 L2 Regularization

L2 regularization can deal with the multicollinearity (independent variables are highly correlated) problems through constricting the coefficient and by keeping all the variables. L2 technique forces the weight to reduce but never makes them zero. Also referred to as ridge regularization, this technique performs best when all the input features influence the output, and all the weights are of almost equal size.

4.3.2 DROPOUT REGULARIZATION

Dropout regularization is one technique used to tackle overfitting problems in deep learning. Dropout is a regularization method approximating concurrent training of many neural networks with various designs. During training, some layer outputs are ignored or dropped at random. This makes the layer appear and is regarded as having a different number of nodes and connectedness to the preceding layer. In practice, each layer update during training is carried out with a different perspective of the specified layer. Dropout makes the training process noisy, requiring nodes within a layer to take on more or less responsible for the inputs on a probabilistic basis.

4.4 OPTIMIZATION

4.4.1 RMSprop

Root Mean Squared Propagation, or RMSProp, is an extension of gradient descent and the AdaGrad version of gradient descent that uses a decaying average of partial gradients in the adaptation of the step size for each parameter. The use of a decaying moving average allows the algorithm to forget early gradients and focus on the most recently observed partial gradients seen during the progress of the search, overcoming the limitation of AdaGrad [6].

4.4.2 ADAM

Adam is an alternative optimization algorithm that provides more efficient neural network weights by running repeated cycles of "adaptive moment estimation." Adam extends on stochastic gradient descent to solve non-convex problems faster while using fewer resources than many other optimization programs. It's most effective in extremely large data sets by keeping the gradients "tighter" over many learning iterations. Adam combines the advantages of two other stochastic gradient techniques, Adaptive Gradients and Root Mean Square Propagation, to create a new learning approach to optimize a variety of neural networks [7].

4.5 Monitoring Performance

While training LSTM networks we found they were overfitting the data and were not able to generalize well if trained for too long, hence we applied the below strategies to prevent overfitting.

4.5.1 EARLY STOPPING

In machine learning, early stopping is a form of regularization used to avoid overfitting when training a learner with an iterative method, such as gradient descent. Such methods update the learner so as to make it better fit the training data with each iteration. Up to a point, this improves the learner's performance on data outside of the training set. Past that point, however, improving the learner's fit to the training data comes at the expense of increased generalization error. Early stopping rules provide guidance as to how many iterations can be run before the learner begins to over-fit. Early stopping rules have been employed in many different machine learning methods, with varying amounts of theoretical foundation [8].

4.5.2 MODEL CHECKPOINT

The notion of the "best" model during training may conflict when evaluated using different performance measures. We try to choose models based on the metric by which they will be evaluated and presented in the domain. In a balanced classification problem, this will most likely be classification accuracy. Therefore, we will use accuracy on the validation in the ModelCheckpoint callback to save the best model observed during training [9].

4.6 HYPER PARAMETER TUNING

We trained model by experimenting different hyper-parameters to find optimal model. Many Hyper-parameters were tested during the experiment like learning rate, optimizer, number of epochs, batchsize, architecture of neural network, etc. We optimized them by watching pattern of change in accuracy, for example for deciding number of epochs we plotted epochs vs loss graph.

4.7 ENSEMBLING

After trying to train individual models, we tried to utilize the potential of ensembling different models. Ensemble methods use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone. Evaluating the prediction of an ensemble typically requires more computation than evaluating the prediction of a single model, so ensembles may be thought of as a way to compensate for poor learning algorithms by performing a lot of extra computation [10]. Fast algorithms such as decision trees are commonly used in ensemble methods (for example, random forests), although slower algorithms can benefit from ensemble techniques as well. Empirically, ensembles tend to yield better results when there is a significant diversity among the models. We created the final ensembled model with Naive Bayes, complement Naive Bayes, LSTM, SVM and Logistic Regression model.

5 RESULTS

After comparing results with other approaches and algorithms, we found ensembling model scored highest accuracy. Please notice, this results were received after training on 80% dataset and keeping 20% for testing, later on some models were retrained with 100% examples to get maximum accuracy on Kaggle competition.

Results of Naive Bayes were pretty surprising, consistent and after tuning the hyper parameters we were able to push the test accuracy to 0.7929. One big advantage of naive bayes is smaller training time, which gave us access to testing multiple text preprocessing techniques. During our experimentation we found Naive Bayes to be more than 200 times faster than LSTM network with even 5 epochs. Later on we also trained Complement Naive bayes which worked well for under represented neutral class.

Logistic Regression helped us push the rank in Kaggle Competition, this simple yet effective classifier was able to score accuracy of 0.8249 on test set after using bayesian search for hyperparameter tuning. Logistic Regression with TF-idf encodings marked 0.825 accuracy on kaggle platform.

We trained two separate Support Vector machine classifiers, one using string kernels and other with tfidf and smote. For string kernels we used custom trained word2vec embeddings and for further improvement of dimentionality we used absolute mean of the word embeddings generated by the word2vec model. We achieved 0.5482 accuracy with this approach. To further improve it we can find better correlated words and embed them using any embeddings with PCA to reduce the dimentions of the embeddings. We achieved 0.8155 accuracy with tfidf approach.

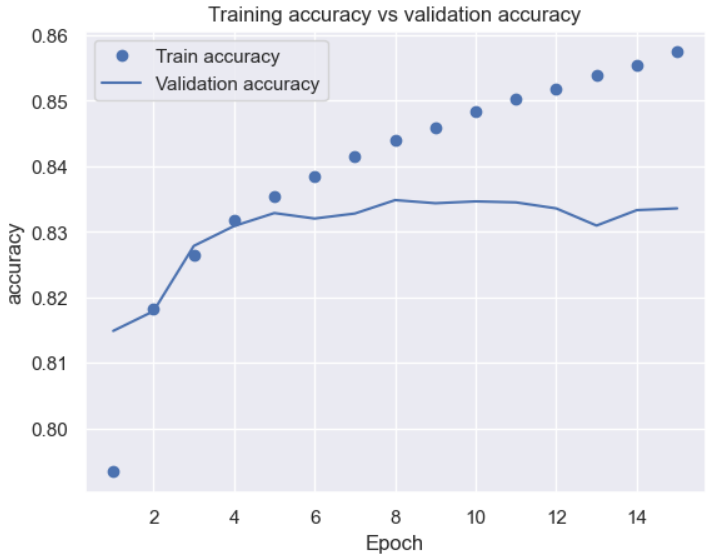

Figure 6: Accuracy Plot

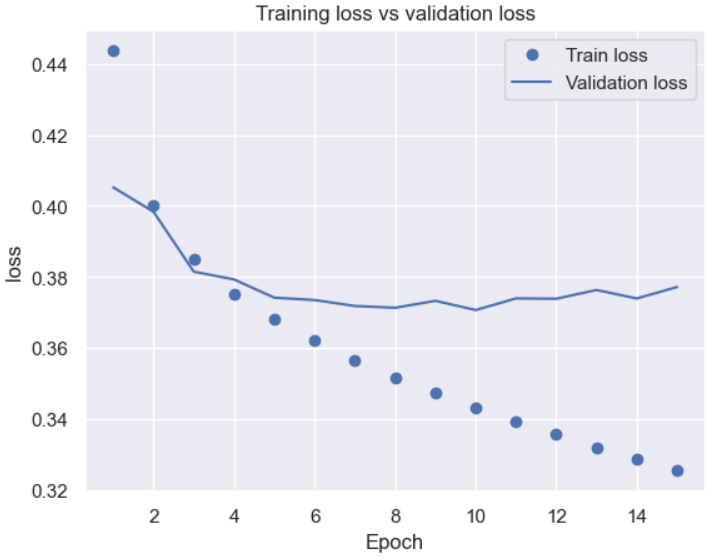

Figure 7: Loss Plot

We also trained XGBoost with default parameters to check its accuracy on text data, we use tfidf encoding for that and received accuracy of 0.7686.

Experimenting with LSTM networks took a lot of effort and we found that the accuracy did not improve as compared to simple models like logistic regression and support vector machines. However after handling the overfitting issue, Bidirectional LSTM architecture took a lead in comparison of validation accuracy. We first tried to implement without regularizar but it did not perform well and to our surprise it started to overfit after first couple of epochs. This shallow network was then replaced with more complex architechture with word2vec embeddings trained on given dataset.

To better monitor the accuracy and overcome the issue of overfitting, we used Dropouts, Early stopping and model checkpoints. The training curve of this architechture is given in Figure 6 and Figure 7.

For Ensembled Training, we used 5 trained models that we optimized earlier. So, same hyperparameters were used during trainig. We found that LSTM model gave best performance after training it for 5 epochs. To overcome the issue of overfitting we used dropouts and early stopping as discussed above. Complement Naive Bayes was used with given training dataset and all other models were trained with smote augmented dataset to overcome the issue of data imbalance. Given model performed the best with the hold out test set.

TABLE 1: COMPARISON OF ACCURACY ON TEST DATA

Classifier	Accuracy Test Data	Feature Transformation	other properties
Ensembled Model	**0.8265**	Tf-idf	best HP
Naive Bayes	**0.7783**	BOW	-
Naive Bayes	**0.7749**	BOW	Stemming
Naive Bayes	**0.7782**	BOW	Lemmatizing
Naive Bayes	**0.7696**	BOW	Stopwords
Naive Bayes	**0.7784**	BOW	ATC
Naive Bayes	**0.7929**	Tfidf	ATC
Logistic Regression	**0.8172**	Tfidf	-
Logistic Regression	0.8249	Tfidf	ATC + HP Tuning
Logistic Regression	0.8249	Tfidf	above + SMOTE
Support Vector Machines	**0.8155**	Tfidf	-
Support Vector Machines	**0.5482**	string kernels	-
XGBoost	**0.7686**	Tfidf	-
LSTM (Above Mentioned Architecture)	**0.8265**	Keras Embeddings	epochs=1000
LSTM (Above Mentioned Architecture)	**0.8348**	Word2Vec	epochs=100
LSTM (Shallow Architecture)	0.9793	Tfidf	epochs=500
LSTM (Shallow Architecture)	0.9700	Tfidf	epochs=100

.
.

TABLE 2: COMPARISON OF ACCURACY ON KAGGLE PLATFORM

Classifiers	Accuracy on Kaggle	Feature Transformation	other properties
Naive Bayes	**0.7574**	BOW	-
Naive Bayes	**0.7644**	BOW	ATC
Naive Bayes	**0.7798**	BOW	ATC + optimized HP
Logistic Regression	**0.82542**	Tf-idf	ATC + optimized HP
Logistic Regression	**0.81954**	Tf-idf	ATC

** Here ATC means Advanced Text cleaning pipeline which is mentioned in 2.2

5.1 EXPLAINABILITY

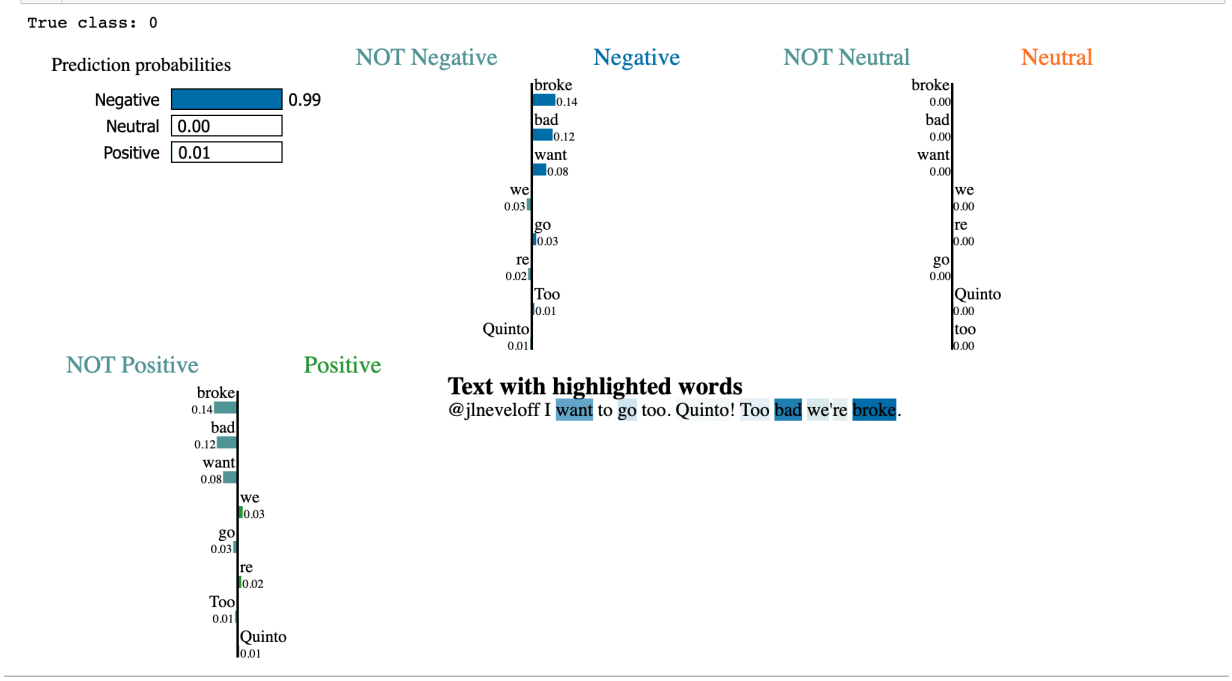

Figure 8: Lime Explainability for Logistic Regression

Local interpretable model-agnostic explanations (LIME), is a technique that approximates any black box machine learning model with a local, interpretable model to explain each individual prediction. It manipulates the input data and creates a series of artificial data consisting of a subset of original attributes. Several versions of the original text are created, in which a certain number of different, randomly selected words are removed. This new artificial data is then assigned to different categories (classified). Observing the presence or absence of certain keywords we can see their influence on the classification of the selected text [12].

Tools like LIME, Gradcam, shap helps us understand what are the forces behind our blackbox algorithms that are affecting the prediction of a particular class. It is also equally important to look at falsely classified examples, it can open many ways for improvement and unnoticeable biases that are present in the model. Here in given example lime clearly explains and highlights the words responsible for prediction to a negative class. Looking at this results we can plan for preparations to reduce False Positive and False Negative samples.

6 DISCUSSION

There some disadvantages of these models implemented, which are discussed below.

As observed above, Ensembled model was the best performer, but this comes with what cost?

Cons of Ensembled model:

- The model is almost 3 times slower than when compared with other model's inference time.
- The model training is too costly compared to improvement from Logistic Regression results.
- The model is not iterpretable at all, as a human it's beyond our capacity to understand models of this complexity.

6.1 FUTURE WORK

There are many other approaches we can try to better understand the problem and solving it logically like transfer

learning, using pretrained word embeddings, etc.

One of such approach can be implemented by using following steps,

1. The Transformer in NLP is a novel architecture that aims to solve sequence-to-sequence tasks while handling long-range dependencies with ease. It relies entirely on self-attention to compute representations of its input and output WITHOUT using sequence-aligned RNNs or convolution.
2. Using a powerful pretrained attention based transformer, and using transfer learning to train it on given dataset, we can get good predictions in sentiment analysis task.
3. Another issue is of limited understandings of words given our dataset. There are many pretrained dictionaries available which are very accurate and has been trained on a huge text corpus.
4. After doing the exploratory data analysis we found that the dataset contains of human written textual posts taken from social media (possibly twitter). One of the biggest challenges of performing NLP tasks on social media data and on text messages is that internet English is vastly different from standard English. There are slangs, acronyms, abbreviations, initialisms, emojis, hashtags, URLs, and among other things, misspellings in posts, tweets, messages, etc that do not appear in standard English. Since it is human generated data, it is bound to many inaccuracies in semantical, syntaxtual and spelling errors. Also given social media texts, it is very important to invest some time in data preprocessing steps that are specific to internet English data, or require special attention [11].

Also, We can apply oversampling for targeted classes that does not have enough data.

7 STATEMENT OF CONTRIBUTION

1. Defining the Problem: Both
2. Developing the methodology: Both
3. Coding the solution:

   Keval: Coded Machine learning experiments.
   Divya: Coded deep learning experiments.
5. Performing the data analysis

   Divya: Worked on exploratory data analysis
   Keval: Worked on Embeddings
6. Writing the report: Both

We hereby state that all the work presented in this report is that of the authors.

8 REFERENCES

1. https://medium.com/@limavallantin/why-is-removing-stop-words-not-always-a-good-idea-c8d35bd77214
2. https://monkeylearn.com/blog/what-is-tf-idf/
3. https://www.mygreatlearning.com/blog/word-embedding/
4. https://www.geeksforgeeks.org/complement-naive-bayes-cnb-algorithm/
5. https://machinelearningmastery.com/an-introduction-to-recurrent-neural-networks-and-the-math-that-powers-them/
6. https://machinelearningmastery.com/gradient-descent-with-rmsprop-from-scratch/
7. https://deepai.org/machine-learning-glossary-and-terms/adam-machine-learning
8. https://en.wikipedia.org/wiki/Early_stopping/
9. https://machinelearningmastery.com/how-to-stop-training-deep-neural-networks-at-the-right-time-using-early-stopping/
10. https://en.wikipedia.org/wiki/Ensemble_learning
11. Wilson, Steven & Magdy, Walid & Mcgillivray, Barbara & Garimella, Kiran & Tyson, Gareth. (2020). Urban Dictionary Embeddings for Slang NLP Applications. Url: https://aclanthology.org/2020.lrec-1.586
12. https://www.steadforce.com/blog/explainable-ai-with-lime