Inspiring Excellence

CSE 440: Natural Language Processing II

Project: Clickbait Detection

Section:02

No.	Name	ID
1	Samanta Binte Taher	21201837
2	Ashraful Alam Nirob	20101514
3	Tahmidul Karim Takee	20101609
4	Orin Akter	20301234

Submission Date:

11 May 2023

Table of Contents

No	Topic	Page No.
1	Introduction	3
2	Dataset Description	4
3	Methodology	5
4	Conclusion	6
5	Reference	7

1. Introduction

A sensationalized headline called "clickbait" drives us to click on a link to an article, picture, or video. User security is seriously threatened by the harmful presence of clickbait in the broad digital world, where information deals with viewers' attention. Clickbait leads readers into accessing apathetic or false material by using charming yet false headlines.

This project aims to provide an enticing solution that uses machine learning to identify and reveal clickbait's sneaky strategies. We have executed a Clickbait Detection utilizing Machine Learning data classification methods. This project will classify the clickbait type categorized in phrase, passage, and multi. Since it's a classification problem we have used three different classification models which are XGBClassifier, LinearSVC, and Recurrent Neural Network(RNN).

Our algorithm will figure out the complex patterns by evaluating huge datasets of labeled clickbait and authentic headlines, enabling people to traverse the online environment confidently and encouraging a more authentic, dependable, and fulfilling digital experience.

2. Dataset Description

The given dataset demonstrates the extent of the clickbait based on distinctive parameters like user id, postId, targetMedia, targetUrl, provenance, spoilerPositions, spoiler postText, postPlatform, targetParagraphs, targetTitle, targetDescription, target keywords, and Tags. The column Tags are our desired output label. The output label comprises three classes, multi, phrases, and passage.

Here are some data visualization results to visualize how our current data looks like:

A kernel density estimate (KDE) plot is a method for visualizing the distribution of

observations in a dataset, analogous to a histogram. KDE represents the data using a continuous probability density curve in one or more dimensions.

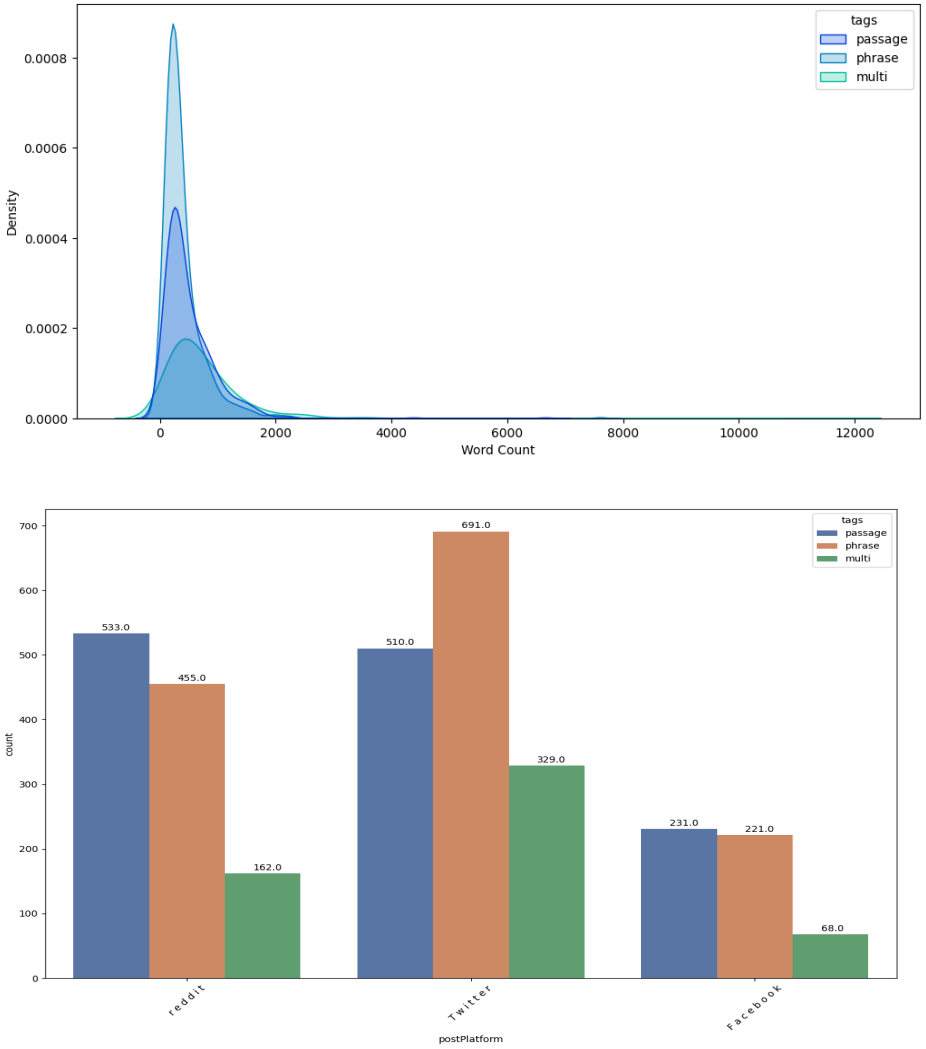

Fig: KDE plot for Word Count per Tag

Another data visualization technique called, Word Cloud, is used to see the most frequent words in a feature. For instance, the most frequently used words on Twitter, Facebook, and Reddit are '**one', 'said', 'will', and 'people**'.

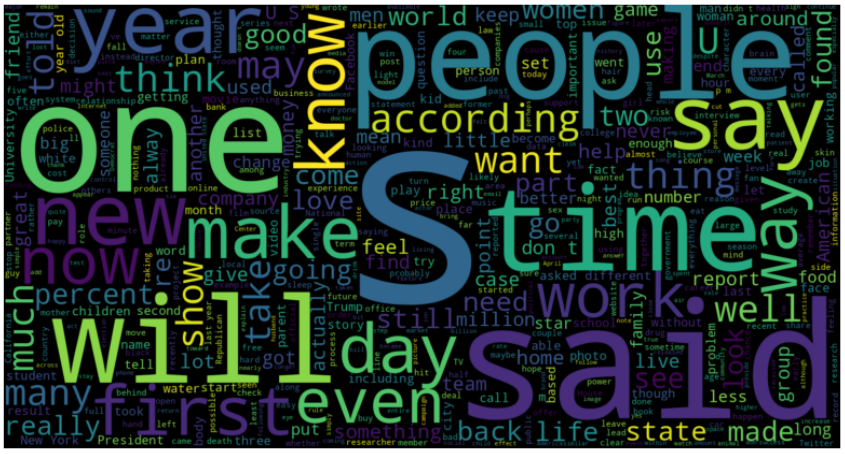

Fig: Word Cloud(Twitter)

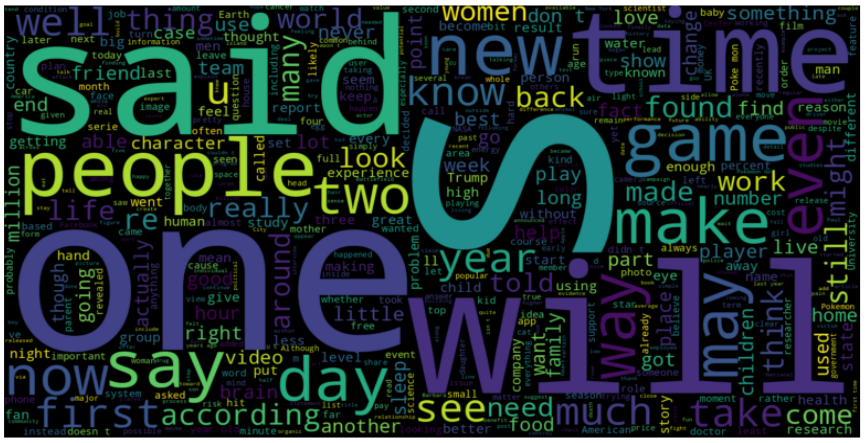

Fig: Word Cloud(Facebook)

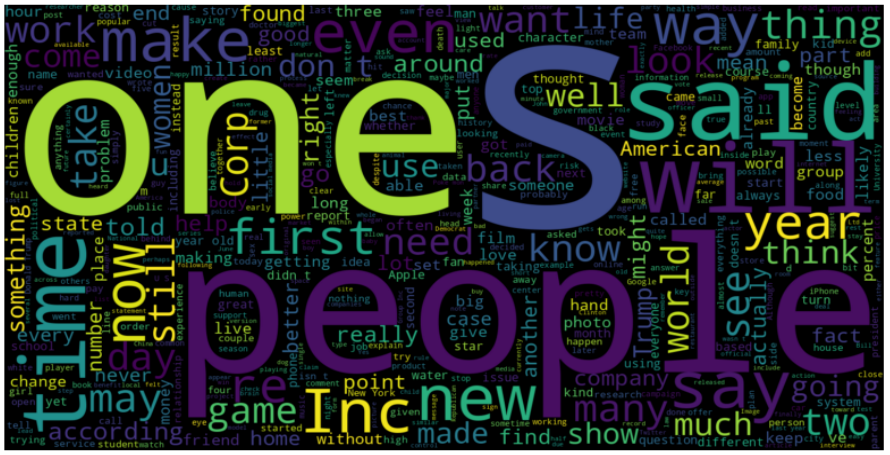

Fig: Word Cloud(Reddit)

3. Methodology

Data Preprocessing and Feature Engineering on Dataset:

i) Dropping Unnecessary columns:

The columns uuid, postId, targetMedia, targetUrl, provenance, and spoiler Positions due to having a lower correlation with the output label.

ii). Dropping Duplicate Values

Each column contained multiple duplicate values, hence they were dropped. As they can confuse our model and lead to a bad performance issue.

iii). Measuring the Percentage of Missing values in every column and deciding whether to drop or feature engineer:

Only two columns TargetKeyword and Target Description had missing values and the percentage of missing values was above 30%. Since the percentage measure was this high we didn’t feature engineered, but rather dropped them entirely.

iv). Cleaning the textual data

The segment's textual data contained space, punctuation marks, URLs, hashtags, and characters that were removed, as well as lemmatization was performed to convert each word into its original form.

v). Vectorizing data:

Using TFIDF Vectorizer and ColumnTransformer the textual data are converted into numerical data.

Model Selection:

Sparse matrices can be quite different from dense matrices, and some machine learning algorithms may perform better or worse on sparse matrices than on dense ones. Generally, linear models and tree-based models tend to perform well on sparse data, while some other algorithms like XGboost and neural networks may require dense data.

That is why chose to run XGboost, Linear Support Vector Classification, and LSTM.

4. Conclusion:

Our models performed decently in comparison to standards. Among LSTM, XGboost , Linear Support Vector Classification this three model expectedly **LSTM** performed better as it is a Recurrent Neural Network with an accuracy score of **0.67** whereas **SVC** performed better with **0.50** accuracy but **XGboost** performed very badly with **0.29** only.

The main reason for the performance drop is less Feature Engineering and a lack of hardware resources. With A decent amount of Feature Engineering, we can get a performance boost of up to 20-25% increase.

In the future, we are hoping to add a feature where we will store the number of Words in cleaned data.

5. Reference:

https://www.kaggle.com/code/shivanimalhotra91/gensim-word2vec-lstm-95-accuracye
https://scikit-learn.org/stable/modules/generated/sklearn.compose.ColumnTransformer.html
https://xgboost.readthedocs.io/en/stable/python/index.html