Email Text Classification Using Machine Learning Techniques

Shashank Gupta
I17MA048

Supervisor : Dr. Indira P. Tripathi
Administrative Supervisor : Dr. Raj Kamal Maurya

Department of Mathematics and Humanities
Sardar Vallabhbhai National Institute of Technology, Surat

May 19, 2022

Table of Contents

Introduction

Text classification is a way for categorising any type of content into or out of a specified category. Machine learning is used to classify text documents into a collection of predetermined classes. The classification is done based on selected documents and features consisting in text documents. The classes are chosen prior to the experiment analysis, and this is referred to as supervised machine learning approach. Organisations and enterprises use numerous text documents for industrial service and government records. In text and archive associations, individual correspondence and records likewise existed. Since there is a huge measure of text data that exists arbitrarily, text arrangement needs are expanding. An AI strategy is expected to sort this information.

In the text classification, statistical analysis is used to give analytical and comparative study for selecting features using Support Vector Machines, Naive Bayes, Decision Trees and Neural Networks[1].

Introduction

Figure: Text classification

Dataset

A methodology is being developed by machine learning techniques on email text classification in this study . **The dataset contains some text messages and labels spam/ham associated with it**. The datasets is chosen from the kaggle and it contains 4993 rows and 2 columns after removing duplicates and null values. Column label contains ham/spam text associated with it while column text contains raw email messages.

First 5 rows of dataset

index	label	text
0	ham	Subject: enron methanol ; meter # : 988291 this is a follow up to the note i gave you on monday , 4 / 3 / 00 { preliminary flow data provided by daren } . please override pop ' s daily volume { presently zero } to reflect daily activity you can obtain from gas control . this change is needed asap for economics purposes .
1	ham	Subject: hpl nom for january 9 , 2001 (see attached file : hplnol 09 . xls) - hplnol 09 . xls
2	ham	Subject: neon retreat ho ho ho , we ' re around to that most wonderful time of the year - - - neon leaders retreat time ! i know that this time of year is extremely hectic , and that it ' s tough to think about anything past the holidays , but life does go on past the week of december 25 through january 1 , and that ' s what i ' d like you to think about for a minute . on the calender that i handed out at the beginning of the fall semester , the retreat was scheduled for the weekend of january 5 - 6 . but because of a youth ministers conference that brad and dustin are connected with that week , we ' re going to change the date to the following weekend , january 12 - 13 . now comes the part you need to think about . i think we all agree that it ' s important for us to get together and have some time to recharge our batteries before we get to far into the spring semester , but it can be a lot of trouble and difficult for us to get away without kids , etc . so , brad came up with a potential alternative for how we can get together on that weekend , and then you can let me know which you prefer . the first option would be to have a retreat similar to what we ' ve done the past several years . this year we could go to the heartland country inn (www . . com) outside of brenham . it ' s a nice place , where we ' d have a 13 - bedroom and a 5 - bedroom house side by side . it ' s in the country , real relaxing , but also close to brenham and only about one hour and 15 minutes from here . we can golf , shop in the antique and craft stores in brenham , eat dinner together at the ranch , and spend time with each other . we ' d meet on saturday , and then return on sunday morning , just like what we ' ve done in the past . the second option would be to stay here in houston , have dinner together at a nice restaurant , and then have dessert and a time for visiting and recharging at one of our homes on that saturday evening . this might be easier , but the trade off would be that we wouldn ' t have as much time together . i ' ll let you decide . email me back with what would be your preference , and of course if you ' re available on that weekend . the democratic process will prevail - - majority vote will rule ! let me hear from you as soon as possible , preferably by the end of the weekend . and if the vote doesn ' t go your way , no complaining allowed (like i tend to do !) have a great weekend , great golf , great fishing , great shopping , or whatever makes you happy ! bobby
3	spam	Subject: photoshop , windows , office . cheap . main trending abasements darer prudently fortuitous undergone lighthearted charm orinoco taster railroad affluent pornographic cuvier irvin parkhouse blameworthy chlorophyll robed diagrammatic fogarty clears bayda inconveniencing managing represented smartness hashish academies shareholders unload badness danielson pure caffein spaniard chargeable levin
4	ham	Subject: re : indian springs this deal is to book the teco pvr revenue . it is my understanding that teco just sends us a check , i haven ' t received an answer as to whether there is a predermined price associated with this deal or if teco just lets us know what we are giving . i can continue to chase this deal down if you need .

Figure: First 5 rows of dataset

Lower case conversion

- Since lowercase and uppercase letters are handled differently by the machine, it is easy for machine to read the text in the same case.
- We converted all the text into lower case.

```
[42]  data['text'] = data['text'].str.lower()
```

```
[44]  data.text[0]

'subject: enron methanol ; meter # : 988291\r\nthis is a follow up to the note i gave you on monday , 4 / 3 / 00 { preliminary\r\nflow data provided by
daren } .\r\nplease override pop ' s daily volume { presently zero } to reflect daily\r\nactivity you can obtain from gas control .\r\nthis change is ne
eded asap for economics purposes .'
```

Figure: Lower case conversion code

Remove punctuations

- There are total 32 main punctuations that need to be taken care of.
- We used the string module with a regular expression to replace any punctuation in text with an empty string.

```
[45]  data['text'] = data['text'].apply(lambda x: re.sub('[%s]' % re.escape(string.punctuation), '' , x))

[46]  data.text[0]

'subject enron methanol  meter   988291\r\nthis is a follow up to the note i gave you on monday  4  3  00  preliminary\r\nflow data provided by daren \r\nplease override pop  s daily volume  presently zero  to reflect daily\r\nactivity you can obtain from gas control \r\nthis change is needed asap for economics purposes '
```

Figure: Remove punctuations code

Remove words and digits containing digits

- When words and digits are combined in a document, it causes a challenge for machines to grasp.
- We must exclude words and numbers that are mixed,such as cat55 or cat5ts5.
- We use regular expressions to remove this with empty string.

```
#remove words and digits
data['text'] = data['text'].apply(lambda x: re.sub('\w*\d\w*','',x))
```

```
data.text[0]
```

```
'subject enron methanol  meter   \r\nthis is a follow up to the note i gave you on monday       preliminary\r\nflow data provided by daren  \r\nplease override pop  s daily volume  presently zero  to reflect daily\r\nactivity you can obtain from gas control \r\nthis change is needed asap for economics purposes '
```

Figure: Remove words and digit containing code

Remove Stop Words

- In the text we have a lot of repeated words like "and" , "or", "at" etc. These are frequently occuring words in any english text.
- These words add no value to our classifier while training. Hence these are called stop words.
- We use NTLK library to remove theses words.

```
[16]
#remove stopwords
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))
stop_words.add('http')
stop_words.add('subject')
def remove_stopwords(text):
    return " ".join([word for word in str(text).split() if word not in stop_words])
data['text'] = data['text'].apply(lambda x: remove_stopwords(x))
```

```
[17]
data.text[0]
```

```
'enron methanol meter follow note gave monday preliminary flow data provided daren please override pop daily volume presently zero reflect daily activit
y obtain gas control change needed asap economics purposes'
```

Figure: Remove stopwords code

Stemming

- Stemming is a process for deleting affixes from words to get the base form, for example doing, done, and did, all of which are originated from the word do.
- NTLK pakage is used to stemmed the words. We have two common examples of stemming algorithms as porter and snowball stemmer. Porter stemmer is widely used tool from the NLTK library.

```
[18]
#stemming
from nltk.stem import PorterStemmer
stemmer = PorterStemmer()
def stem_words(text):
    return " ".join([stemmer.stem(word) for word in text.split()])
data["text"] = data["text"].apply(lambda x: stem_words(x))
```

```
[19]
data.text[0]
```

```
'enron methanol meter follow note gave monday preliminari flow data provid daren pleas overrid pop daili volum present zero reflect daili activ obtain g
a control chang need asap econom purpos'
```

Figure: Stemming code

Lemmatization

- The stemming technique is not utilised in production since it is inefficient and frequently stems undesired words. As a result, another approach known as lemmatization was introduced to the market to overcome the problem.
- Actually, Lemmatization is a systematic way to reduce the words into their lemma by matching them with a language dictionary.

[20]
```python
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
def lemmatize_words(text):
    return " ".join([lemmatizer.lemmatize(word) for word in text.split()])
data["text"] = data["text"].apply(lambda text: lemmatize_words(text))
```

[21]
```python
data.head()
```

1 to 5 of 5 entries Filter

index	label	text
0	ham	enron methanol meter follow note gave monday preliminari flow data provid daren plea overrid pop daili volum present zero reflect daili activ obtain ga control chang need asap econom purpos
1	ham	hpl nom januari see attach file hplnol xl hplnol xl
2	ham	neon retreat ho ho ho around wonder time year neon leader retreat time know time year extrem hectic tough think anyth past holiday life go past week decemb januari like think minut calend hand begin fall semest retreat schedul weekend januari youth minist confer brad dustin connect week go chang date follow weekend januari come part need think think agre import u get togeth time recharg batteri get far spring semest lot troubl difficult u get away without kid etc brad came potenti altern get togeth weekend let know prefer first option would retreat similar done past sever year year could go heartland countri inn www com outsid brenham nice place bedroom bedroom hous side side countri real relax also close brenham one hour minut golf shop antiqu craft store brenham eat dinner togeth ranch spend time meet saturday return sunday morn like done past second option would stay houston dinner togeth nice restaur dessert time visit recharg one home saturday even might easier trade would much time togeth let decid email back would prefer cours avail weekend democrat process prevail major vote rule let hear soon possibl prefer end weekend vote go way complain allow like tend great weekend great golf great fish great shop whatev make happi bobbi
3	spam	photoshop window offic cheap main trend aba darer prudent fortuit undergon lightheart charm orinoco taster railroad affluent pornograph cuvier irvin parkhous blameworthi chlorophyl robe diagrammat fogarti clear bayda inconvenienc manag repres smart hashish academi sharehold unload bad danielson pure caffein spaniard chargeabl levin
4	ham	indian spring deal book teco pvr revenu understand teco send u check receiv answer whether predermin price associ deal teco let u know give continu chase deal need

Figure: Lemmatization code

Remove Extra Spaces

- The majority of text data is unstructured and includes extra gaps between words. To overcome this problem, we use regular expression to remove extra space.

```
[22]  data["text"] = data["text"].apply(lambda x: re.sub(' +', ' ', x))

[23]  data.text[0]

'enron methanol meter follow note gave monday preliminari flow data provid daren plea overrid pop daili volum present zero reflect daili activ obtain ga
control chang need asap econom purpos'
```

Figure: Remove extra space code

Counting word occurence(CWO)

- In this feature selection technique, the whole corpus is broken down into separate words and the count of the each word is considered as the features of the model.
- The reason for this methodology is that a keyword or key signal will emerge again. So, if the frequency of recurrence indicates the value of a term, more often signifies more significance.

```python
doc = "This movie is very scary and long. This movie is not scary and is slow. This movie is spooky and good"

count_vec = CountVectorizer()
count_occurs = count_vec.fit_transform([doc])
count_occur_df = pd.DataFrame((count, word) for word, count in zip(count_occurs.toarray().tolist()[0], count_vec.get_feature_names_out()))
count_occur_df.columns = ['Word', 'Count']
count_occur_df.sort_values('Count', ascending=False, inplace=True)
count_occur_df.head()
```

1 to 5 of 5 entries Filter

index	Word	Count
2	is	4
0	and	3
4	movie	3
9	this	3
6	scary	2

Figure: CWO code

Normalized count occurence(NCO)

- If we believe that high frequency will dominate the outcome, resulting in model bias. Pipeline normalisation is simple to implement.

```
doc = "This movie is very scary and long. This movie is not scary and is slow. This movie is spooky and good"

norm_count_vec = TfidfVectorizer(use_idf=False, norm='l2')
norm_count_occurs = norm_count_vec.fit_transform([doc])
norm_count_occur_df = pd.DataFrame((count, word) for word, count in zip(
    norm_count_occurs.toarray().tolist()[0], norm_count_vec.get_feature_names_out()))
norm_count_occur_df.columns = ['Word', 'Count']
norm_count_occur_df.sort_values('Count', ascending=False, inplace=True)
norm_count_occur_df=norm_count_occur_df.round(decimals=2)
norm_count_occur_df.head()
```

1 to 5 of 5 entries Filter

index	Word	Count
2	is	0.55
0	and	0.41
4	movie	0.41
9	this	0.41
6	scary	0.27

Figure: NCO code

Term frequency inverse document frequency(TFIDF)

- TFIDF takes a different strategy, believing that high frequency may not be capable of providing significant information gain. To put it another way, unusual words give the model greater weight.

$$\text{Term frequency(TF)} = \frac{\text{Number of repetitions of word in a sentence}}{\text{Total words in a sentence}}$$

$$\text{Inverse document frequency(IDF)} = \text{Log}\left[\frac{\text{Number of Sentences}}{\text{Number of sentences containing the word}}\right]$$

$$\text{Final score} = \text{TF} * \text{IDF}$$

Figure: TFIDF formula

```python
doc = "This movie is very scary and long. This movie is not scary and is slow. This movie is spooky and good"
tfidf_vec = TfidfVectorizer()
tfidf_count_occurs = tfidf_vec.fit_transform([doc])
tfidf_count_occur_df = pd.DataFrame((count, word) for word, count in zip(
    tfidf_count_occurs.toarray().tolist()[0], tfidf_vec.get_feature_names_out()))
tfidf_count_occur_df.columns = ['Word', 'Count']
tfidf_count_occur_df.sort_values('Count', ascending=False, inplace=True)
tfidf_count_occur_df=tfidf_count_occur_df.round(decimals=2)
tfidf_count_occur_df.head()
```

1 to 5 of 5 entries Filter

index	Word	Count
2	is	0.55
0	and	0.41
4	movie	0.41
9	this	0.41
6	scary	0.27

Figure: TFIDF code

Bag of n-grams

- A 2-gram (or bigram) is a two-word series of words like "What is", "is your", "your name".
- A 3-gram (or trigram) is a three-word sequence of words like "What is your", "is your name".
- N-gram models are used to determine the probability of the n-final gram's word given the previous words, as well as to assign probabilities to whole sequences.
- In our research, we have implemented not beyond 2-gram approach for feature engineering. Since k unique words can mean k^2 unique bigrams, implementing more than 2-grams will cost very high computational cost.

Support vector classifier

- Support Vector Machine or SVM is supervised learning algorithms, which is used for classification as well as regression problems. However, primarily, it is used for classification problems in machine learning.
- The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.
- SVM chooses the extreme points/vectors that helps in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:

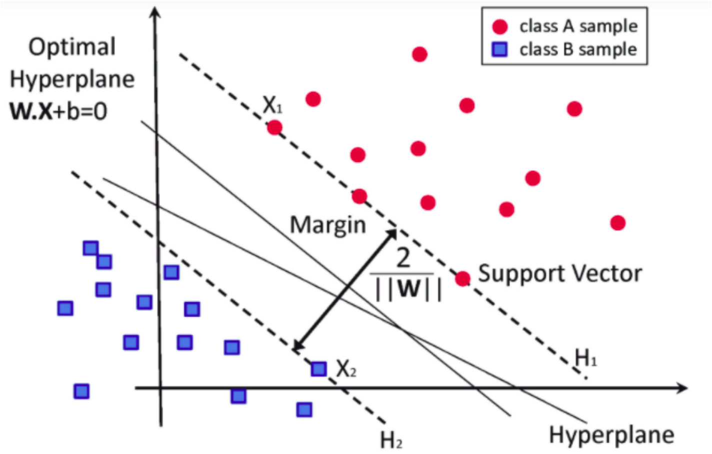

Figure: SVM using hyperplane

Naive Bayes Classifier

- Naive Bayes classifier is generated by Bayes Theorem and utilized to perform classification tasks . It's a collection of algorithms that all act on the same principle i.e. each pair of features being classified is independent to the others. The Bayes Theorem is expressed as follows:

$$P(\text{spam}|x) = (P(x|\text{spam}) * P(\text{spam}))/P(x)$$

- Where x is a feature vector containing the words coming from the Spam (or Ham) emails. in other terms:

$$\text{Posterior} = \frac{\text{likelihood} * \text{prior}}{\text{evidence}}$$

Logistic regression classifier

- A statistical model that employs the logistic function to predict a binary dependent variable. A sigmoid function is another term for the logistic function, which is defined as:

$$F(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1}$$

- This function helps the logistic regression model in compressing the entries from (-k,k) to (0,1).
- Logistic regression, like linear regression, starts with a linear equation. This equation, on the other hand, is formed of log-odds that are then processed through a sigmoid function that reduces the result of the linear equation to a probability between 0 and 1

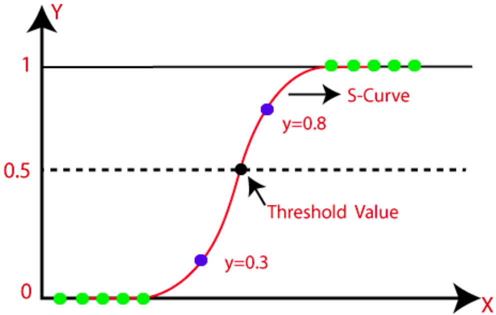

Figure: Logistic regression classification

Decision Tree Classifier

- Decision tree is a tree structure like a flowchart in which each leaf node symbolises the outcome and each inner node shows a characteristic or attribute. The uppermost node is the root node of tree.
- It determines how to split based on the value of an attribute. This framework assists us in drawing conclusions. It's a flowchart graph that slightly approximates human reasoning. As a result, decision trees is easy to understand and interpret.

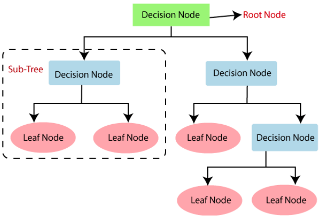

Figure: Decision Tree

K nearest neighbour classifier

- A k-nearest-neighbor algorithm, abbreviated knn, is a data categorization method that calculates how probable a data point is to belong to one of two groups based on which group the data points closest to it belong to.

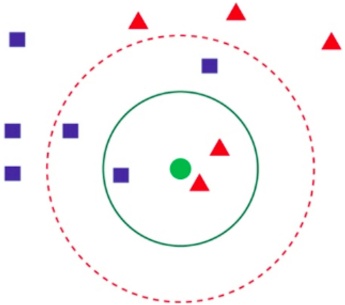

Figure: Classification using KNN approach

Accuracy and Error Rate

The number of correct predictions generated by the algorithm divided by the total data collected is the accuracy[2]. The accuracy is stated mathematically as

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

The error rate (ERR) is the total number of inaccurate predictions divided by the total data collection. The best error rate is 0.0, while the worst is 1.0. The error rate is stated mathematically as:

$$\text{Error rate} = 1 - \text{Accuracy}$$

Precision

The number of emails correctly recognised (True Positive) divided by the number of emails assigned to a certain category by the classifier (True Positive and False Positive). In mathematical form, the precision is:

$$\text{Precision} = \frac{TP}{TP + FP}$$

Recall

The recall is the proportion of accurately predicted positive emails to all emails in the actual class. The recall is stated mathematically as

$$\text{Recall} = \frac{TP}{TP + FN}$$

F1-score

The F1-score is the harmonic mean of accuracy and recall. When we add Precision and Recall together, we get the F1-score. F1-score has optimal and worst values of 1 and 0, respectively. As it includes both precision and recall, using one number for measurement is more efficient than using two. The F1-score is calculated as follows:

$$\text{F1-score} = \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

Confusion Matrix

	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)

Figure: Confusion matrix

Email classification matrix

- Accuracy score will imply that spam message goes to spam folder and ham message goes to inbox.
- Precision will imply that the message is spam and it goes to inbox.
- Recall score will imply that the message is ham and it goes to spam folder.

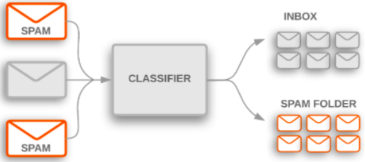

Figure: Email classification

Support vector classifier

Feature Selection Technique	Accuracy %	Precision %	Recall %	F1 Score
Counting Words Occurence (CWO)	97.2	97	96	0.97
Normalized Count Occurence (NCO)	98.6	98	99	0.98
TFIDF	99.4	99	99	0.99
Bi-grams	96.8	94	97	0.95

Figure: SVC classifier results

- The TFIDF approach achieves 99.4% accuracy on the test set.
- We will concentrate on recall rather than accuracy since we do not want non-spam emails to be categorised as spam. In terms of recall, TFIDF and Normalized count occurrence are both doing well.
- For all feature selection strategies, the F1 score is also balanced. It means that SVC is doing well across the board for all feature selection strategies.

Naive bayes classifier

Feature Selection Technique	Accuracy %	Precision %	Recall %	F1 Score
Counting Words Occurence (CWO)	98.3	98	98	0.98
Normalized Count Occurence (NCO)	91.09	94	85	0.88
TFIDF	91.49	95	86	0.89
Bi-grams	95.4	97	92	0.94

Figure: NB classifier results

- Accuracy by CWO technique is the highest(98.3%) and lowest by NCO (91.09%).
- NCO performed quite poorly as it contains lowest recall and F1 score: 85% and 0.88 respectively.

Logistic regression classifier

Feature Selection Technique	Accuracy %	Precision %	Recall %	F1 Score
Counting Words Occurence (CWO)	98.5	98	98	0.98
Normalized Count Occurence (NCO)	97.8	97	98	0.97
TFIDF	99.8	99	99	0.99
Bi-grams	96.9	96	96	0.96

Figure: LR classifier results

- TFIDF provides the best accuracy(99.8%) and recall(99%), although the Logistic regression classifier is also successful with other approaches.
- Bi-grams has the lowest overall scores, although it is still significantly better than some of the Naive Bayes approaches.

Decision tree classifier

Feature Selection Technique	Accuracy %	Precision %	Recall %	F1 Score
Counting Words Occurence (CWO)	92.49	90	93	0.91
Normalized Count Occurence (NCO)	92.49	90	93	0.91
TFIDF	92.49	90	93	0.91
Bi-grams	78.48	79	84	0.78

Figure: DT classifier results

- The Bi-grams approach is ineffective since it has just 78.48 percent accuracy on the test set and only 79 percent precision. It signifies that the decision tree classifier has identified a large number of spam messages as non-spam.
- All the remaining techniques are doing well and have the same numbers from accuracy to F1 score.

K nearest neighbour classifier

Feature Selection Technique	Accuracy %	Precision %	Recall %	F1 Score
Counting Words Occurence (CWO)	88.49	86	91	0.87
Normalized Count Occurence (NCO)	95.4	95	94	0.94
TFIDF	96.9	96	96	0.96
Bi-grams	60.46	71	72	0.60

Figure: KNN classifier results

- Bi-grams and CWO both had poor results, with Bi-grams being the poorest. In terms of accuracy and recall, TFIDF is the best feature selection technique for this algorithm.

Conclusion

- Email text classification is one of the applications of text classification. We use python (google colaboratory) as an experimental tool to provide email classification, feature selection, and performance evaluation.
- Our proposed scheme can be used in R, Tensor Flow, or a Matlab simulation platform. We used preprocessing in the beginning to choose the best features from the dataset.
- A supervised machine learning approach is used to extract text features from English language-based email texts. We examined and analysed various machine learning algorithms, including NB, SVM, DT, KNN and LR.
- Some selected metrics, such as accuracy, precision, recall, and F1 value, were used in the evaluation and comparison. Finally, we discussed the findings of our preferred machine learning techniques. Based on the simulations, it's evident that the SVM and LR outperformed the other machine learning approaches on the datasets

Future Work

- Apply Deep learning techniques on dataset and compare with existing algorithms.
- Improve hyperparameters of the algorithms and re run models.
- Run above algorithm on different email datasets and develop a general methodology to classify emails.

References

1. Xiaoyu Luo. Efficient english text classification using selected machine learning techniques. Alexandria Engineering Journal, 60(3):3401–3409, 2021.
2. Umid Suleymanov, Samir Rustamov, Murad Zulfugarov, Orkhan Orujov, Nadir Musayev, and Azar Alizade. Empirical study of online news classification using machine learning approaches. In 2018 IEEE 12th International Conference on Application of Information and Communication Technologies (AICT), pages 1–6.IEEE, 2018.

Thank you!