Topic Modeling and Text Classification Using Python

Abstract

The focus of this project is the development of machine learning models capable of text classification for sentiment analysis and topic modeling. The data used for this project is lines of dialog from the Lord of the Rings films. Using this data, the group created both logistic regression and latent dirichlet allocation models that were able to classify the sentiment of a line of dialog and provide summary topics.

1. Introduction

Text based data has become incredibly popular amongst data scientists, business analysts, and more in the past several years. For example, the importance of being able to identify a negative or positive review based solely on the text in the review itself could be a major asset to a business, or consider a researcher that needs the ability to sum up the main points of scientific articles and journals but cannot physically read each of these documents. For this project, we explored solutions to both of these hypothetical problems by using machine learning methods applied to a text based data set containing lines of dialog from the Lord of the Rings movies.

2. Early Analysis

2.1. Dataset

The data sets we used for this project were “lotr_scripts.csv” and “Articles.csv”. The Lord of the Rings data set has 2391 rows and 4 columns. The 4 variables in this data set are:

X: An index variable
Char: The character that is speaking
Dialog: The line of dialog spoken by the character
Movie: The movie that the line of dialog belongs to

The “Articles.csv” data set consists of news articles from 2015 to the present. These articles fall into one of two categories; business and sports. The data set contains 2692 rows and 4 columns. The 4 columns in this data set are:

Articles: Text the news articles contain and the place it was published.

Heading: Text containing the heading of the news article.

Data: The date the article was published

NewsType: Type of Art (Business or Sports)

2.2. Research Questions

The group considered two research questions:

1. Can we develop a machine learning model capable of text classification for sentiment analysis?
2. Can we develop a machine learning model capable of topic modeling

2.3. Proposed Methodology

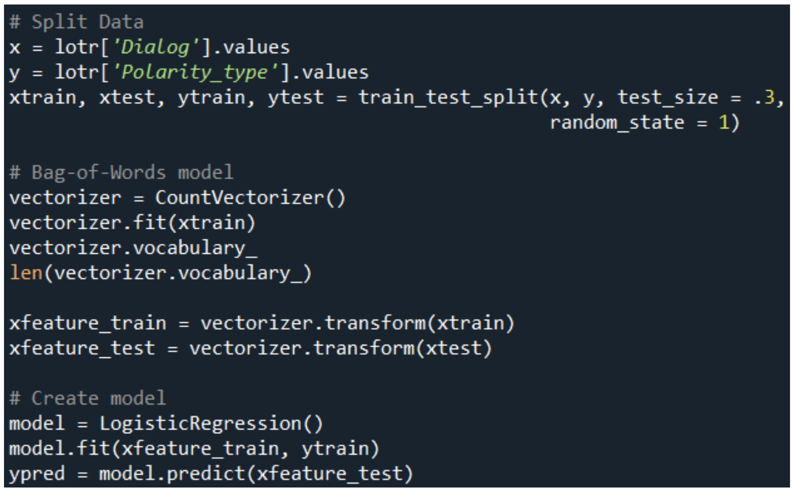
For text classification for sentiment analysis, the group will utilize the TextBlob and sklearn packages in Python. We will create a new column that contains the polarity (the measure of sentiment, a measured between -1 and 1) of a line of dialog and ultimately fit a logistic regression model to predict whether a line of dialog is negative, positive, or neutral based on a -1, 0, 1 classification respectively. For topic modeling, we will utilize similar functions from the sklearn package used previously, but we will primarily utilize latent dirichlet allocation to identify 10 different topics.

3. Applied Statistical Analysis

3.1. Text Classification

To start the text classification process, the group preprocessed and tidied the data. In addition, a new column for polarity was added along with a new dummy variable. The group specified that all observations with a polarity less than 0 would be denoted by -1 (negative statement) and all observations with polarity greater than 0 would be denoted by 1 (positive statement), all other observations were labeled 0 (neutral statement).

The group defined the target variable as the new dummy variable Polarity_type, and feature variable as Dialog. CountVectorizer() was used to learn the vocabulary from the training data, and then to convert the dialog into a sparse matrix. In this matrix each row represents a line of dialog, and each column is a unique word in the vocabulary. The cell values indicate word counts, resulting in a sparse matrix with a large number of 0 values. After transforming the text data into a matrix of token counts using the vocabulary learned during the fitting process, a logistic regression model was applied and the model achieved around an 80 percent accuracy rating. However, the model tended to predict 0 most often.

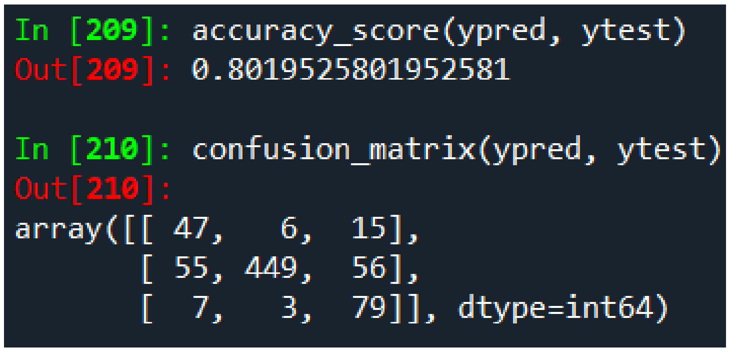


```
# Split Data
x = lotr['Dialog'].values
y = lotr['Polarity_type'].values
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size = .3,
                                                random_state = 1)

# Bag-of-Words model
vectorizer = CountVectorizer()
vectorizer.fit(xtrain)
vectorizer.vocabulary_
len(vectorizer.vocabulary_)

xfeature_train = vectorizer.transform(xtrain)
xfeature_test = vectorizer.transform(xtest)

# Create model
model = LogisticRegression()
model.fit(xfeature_train, ytrain)
ypred = model.predict(xfeature_test)
```



```
In [209]: accuracy_score(ypred, ytest)
Out[209]: 0.8019525801952581

In [210]: confusion_matrix(ypred, ytest)
Out[210]:
array([[ 47,   6,  15],
       [ 55, 449,  56],
       [  7,   3,  79]], dtype=int64)
```

3.2. Topic Modeling

The group used the CountVectorizer() function again to fit a document term matrix. A document term matrix is a way to represent textual data in a structured numerical format. In our matrix, rows correspond to lines of dialog, and columns correspond to individual terms or words present in the dialog. The parameters used in the count vectorizer are as follows:

- max_df =.95: Terms in more than 95 percent of the documents are ignored
- min_df=5: Terms in less than 5 documents are ignored
- stop_words = “English”: Ignores words such as “or,” “it” and “the”

Finally we set the parameters of latent dirichlet allocation, the model that will be used to identify topics. Here, the group only had to specify one parameter:

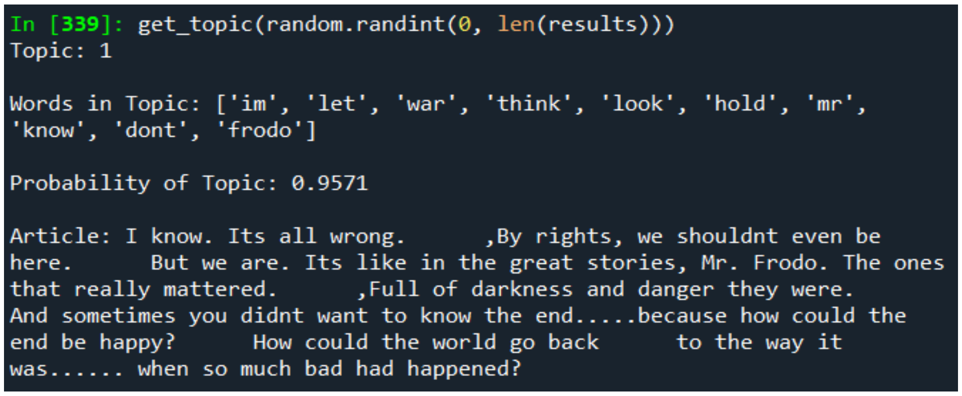
- n_components = 10: We want to identify 10 topics

Next the group fit the document term matrix to the model and wrote a function to display each topic identified by the model, these are shown below. Each topic is based on the 10 words with the highest probability of belonging to a specific topic.

Topic 1	end, need, want, sam, shire, let, know, mr, dont, frodo
Topic 2	away, sauron, great, ring, master, lord, mordor, men, gondor, come
Topic 3	said, hope, victory, wish, lets, right, battle, sorry, just, im
Topic 4	carry, ive, hobbits, old, friends, youve, come, long, youre, ring
Topic 5	le, theres, lost, tree, old, city, aragorn, way, dead, gandalf
Topic 6	remember, quick, hurry, know, saruman, run, took, like, tell, kill
Topic 7	leave, whats, rohan, does, theoden, good, man, did, look, king
Topic 8	ill, day, coming, bilbo, baggins, death, pippin, time, hes, yes
Topic 9	youll, doing, home, thats, road, war, shall, ride, sam, merry
Topic 10	say, master, hobbit, gollum, think, nice, hold, precious, oh, smeagol

Finally, we applied the trained LDA model to transform the document term matrix into a topic distribution for each line of dialog. This is still a matrix, but now rows represent lines of dialog, and the values in each row represent the probability of the line belonging to each topic.

After writing a function to display the results neatly, we obtained some randomly selected outputs which are shown below.

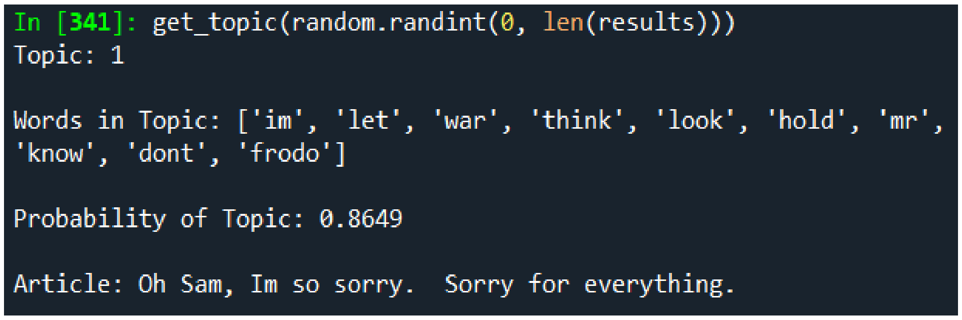


```
In [339]: get_topic(random.randint(0, len(results)))
Topic: 1

Words in Topic: ['im', 'let', 'war', 'think', 'look', 'hold', 'mr',
'know', 'dont', 'frodo']

Probability of Topic: 0.9571

Article: I know. Its all wrong.      ,By rights, we shouldnt even be
here.      But we are. Its like in the great stories, Mr. Frodo. The ones
that really mattered.      ,Full of darkness and danger they were.
And sometimes you didnt want to know the end.....because how could the
end be happy?      How could the world go back      to the way it
was...... when so much bad had happened?
```



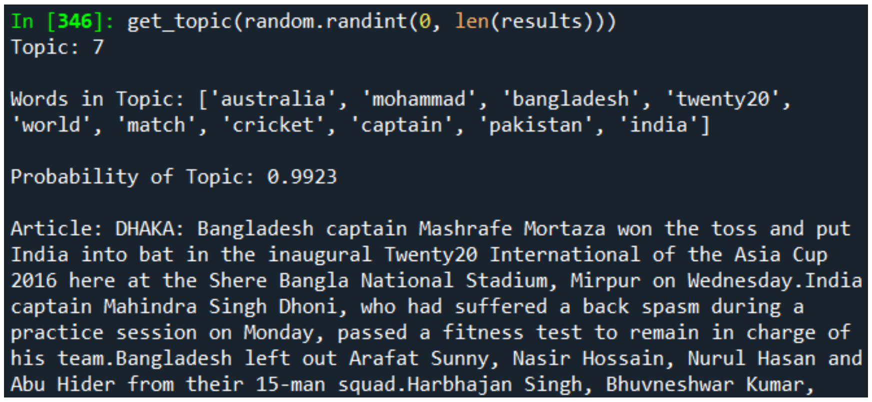
```
In [341]: get_topic(random.randint(0, len(results)))
Topic: 1

Words in Topic: ['im', 'let', 'war', 'think', 'look', 'hold', 'mr',
'know', 'dont', 'frodo']

Probability of Topic: 0.8649

Article: Oh Sam, Im so sorry.  Sorry for everything.
```

The same methodology was used on a data set containing global news articles in an effort to compare the two. The code used was the exact same for this second round of topic modeling. The use of more structured and polarized data helped further validate the model used for the Lord of the Rings data set.



```
In [346]: get_topic(random.randint(0, len(results)))
Topic: 7

Words in Topic: ['australia', 'mohammad', 'bangladesh', 'twenty20',
'world', 'match', 'cricket', 'captain', 'pakistan', 'india']

Probability of Topic: 0.9923

Article: DHAKA: Bangladesh captain Mashrafe Mortaza won the toss and put
India into bat in the inaugural Twenty20 International of the Asia Cup
2016 here at the Shere Bangla National Stadium, Mirpur on Wednesday.India
captain Mahindra Singh Dhoni, who had suffered a back spasm during a
practice session on Monday, passed a fitness test to remain in charge of
his team.Bangladesh left out Arafat Sunny, Nasir Hossain, Nurul Hasan and
Abu Hider from their 15-man squad.Harbhajan Singh, Bhuvneshwar Kumar,
```

4. Discussion

In conclusion our methods work proficiently with what we had in mind originally when we first thought of the concepts we wanted to cover. The logistic regression model used in text classification for sentiment analysis predicted polarity at 80% accuracy. The only problem we faced with this was that the distribution of the classes was skewed, 0 was far more common than -1 or 1. As we moved into topic modeling, we noticed that our original data set that contained movie lines from the *Lord of the Rings*, we noticed that it didn't give us enough for the end goal we had in mind. In practice topic modeling did work, however the topics were a bit vague, likely due to the lines of dialog themselves being relatively short. That is what led us to using the articles data set because it had more definitive topics in the text data to reach our goal of effectively using topic modeling.

References

- https://www.kaggle.com/datasets/paultimothymooney/lord-of-the-rings-data?select=lotr_scripts.csv
- https://www.kaggle.com/datasets/asad1m9a9h6mood/news-articles