Film Classification

Abstract

NLP is a field that emphasises practice and application, where innovations can be new algorithms, tasks, applications, data, discoveries, etc., and its impact depends on how it promotes the development of the field. In the field of artificial intelligence, natural language processing belongs to a relatively small branch, but it serves a very important role. Chatbots with intelligent replies designed through a large amount of data analysis, navigation assistants with different voices and even local dialects in China, knowing the attitude of Twitter users towards a certain news by analysing their positive and negative emotions, and knowing the attitude of Twitter users towards a certain news through a large number of readings of high-frequency vocabulary of spam text messages , automatically filtering spam content.

The usefulness of NLP goes far beyond this, with the emergence of ChatGPT, it is also possible to be a 'screenwriter' yourself, by inputting a film genre and keywords to ChatGPT, for example, input the film genre, comedy film, and then input the keywords, Japan, school, ghosts, etc., and ChatGPT will automatically generate a film based on that content and by combining it with a web A large amount of film content, automatically generate a film. In order to figure out the principle and study the related NLP modules, for this purpose I worked on a film classification application, this dataset contains different types of films and film lines, the original dataset was a time-limited competition in Kaggle with a total number of 22580 entries, in order to run the code in a more portable way I removed the number of datasets in the train to 15471 entries. I use, and make predictions for film lines. Using numpy, os, I trained the dataset using LSTM, CNN and MultinomialNB to train the dataset model for the main training, predicted the accuracy of CNN, and predicted the text of the movie lines using MultinomialN, and at the end, I used IF - IDF, LDA, sentence_transform package to read the code and produce new model data.

Data Cleaning and Preprocessing

Firstly, the dataset is divided into three categories: 'train', 'test', and 'Val', and in the subsequent training, since these three datasets are too large, the following datasets are added: 'val1' with only a thousand or so data. 'val1' has only a thousand or so pieces of

data, after Create the environment, set numpy, pandas , os, matplotlib, seaborn as sns %matplotlib inline as the basis of the run of the environment code module import. Defined as df, using the panda module to read the training set/movie_train.csv, read to learn that there are a total of three columns, respectively 'id, 'text', 'genre', the 15470 film data. Open train.csv to confirm that the reading is correct.

```
df.columns
```

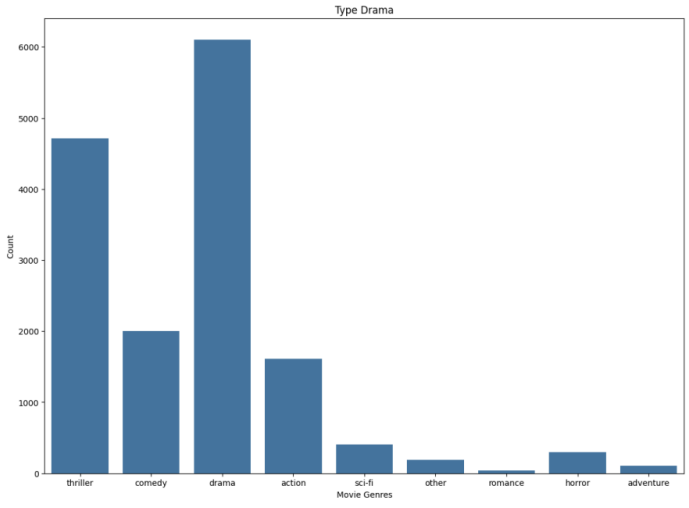
```
Index(['id', 'text', 'genre'], dtype='object')
```

```
df.shape
```

```
(15470, 3)
```

15464	19336	I would've told you, and you would have stc	comedy
15465	19340	STEINSMA Mr. Knox! The knife! Mickey glan	drama
15466	19342	k hands. It's full combat. The Korean Dragor	thriller
15467	19343	ce time. ROCKY Thanks, Mr. Gazzo. Rocky e	drama
15468	19345	evere bandages on the spear caused holes c	action
15469	19348	en at DR. SCOTT. BRAD JANET Dr. Scott! FR/	comedy
15470	19349	bs, who crashes into his computer table, ser	thriller
15471	19350	it all. ARLO blows BRYNNER grunts continu	thriller
15472			

'id' represents the serial number of the film, 'text' represents a part of the lines of this film, 'genre' represents the type of the film, in the visualised database. Calculate the number of 'genre', set the chart size figsize=(14,10), y-axis is the number, x-axis is the name of the genre, use %matplotlib to show plt.show(). Yields a total of nine types of films, 'action', 'adventure', 'comedy', 'drama', 'horror', 'other', 'romance', 'sci-fi', 'thriller', with drama having the highest number of 6100, and the second highest number of film types being thriller with about 4750 data and the least number of films is 'romance' with only about six per cent of the 15740 films.



Next the nine film genres are defined as VALUES as 'other': 0, 'action': 1, 'adventure': 2, 'comedy':3, 'drama':4, etc., and 'genre' is changed from ' thiller', 'comdy' to '8', '3' defined respectively. Prepare for the next cleaning data, data preprocessing.

```
           id                                               text     genre
0           0  eady dead, maybe even wishing he was. INT. 2ND...  thriller
1           2  t, summa cum laude and all. And I'm about to l...    comedy
2           3   up Come, I have a surprise.... She takes him ...     drama
3           4  ded by the two detectives. INT. JEFF'S APARTME...  thriller
4           5  nd dismounts, just as the other children reach...     drama
...       ...                                                ...       ...
15465   19343  ce time. ROCKY Thanks, Mr. Gazzo. Rocky enters...     drama
15466   19345  evere bandages on the spear caused holes of hi...    action
15467   19348  en at DR. SCOTT. BRAD JANET Dr. Scott! FRANK G...    comedy
15468   19349  bs, who crashes into his computer table, sendi...  thriller
15469   19350   it all. ARLO blows BRYNNER grunts continues u...  thriller
```

	id	text	genre
0	0	eady dead, maybe even wishing he was. INT. 2ND...	8
1	2	t, summa cum laude and all. And I'm about to l...	3
2	3	up Come, I have a surprise.... She takes him ...	4
3	4	ded by the two detectives. INT. JEFF'S APARTME...	8
4	5	nd dismounts, just as the other children reach...	4
5	6	breadth of the bluff. Gabe pulls out his ancie...	8
6	7	uilding. A MAN in pajamas runs out into the ra...	8
7	9	ELLES AND RITA HAYWORTH Just disgustingly rich...	4
8	10	Memphis goes back into the garage, Budgy cack...	8
9	11	e reels as the world spins. Sweat pours off hi...	1

Installation of the NLTK library. The NLTK library provides a rich set of natural language processing functions and tools, offering a variety of text preprocessing tools, including text cleaning, text normalisation, word splitting, etc. These tools can help users quickly transform raw text data into a data format that can be used for further analysis. Using the nltk library's 'stopwords', stopwords are common words that are ignored in text processing, such as "the", "a", "an", etc. to clean the text, remove special characters from dialogues/scripts, convert the whole dialogues/scripts to lower case, tag dialogues/scripts by words, remove stop words, stem words, join words in the stem, and create a corpus.

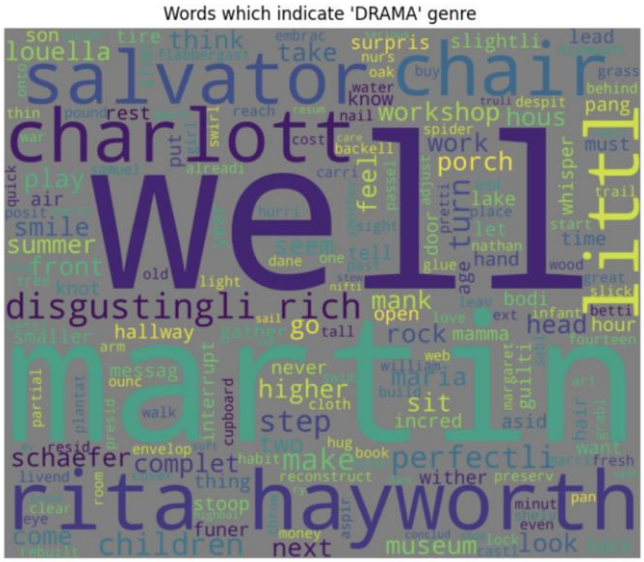
Install the wordcould library to generate word cloud images from text, the wordcloud

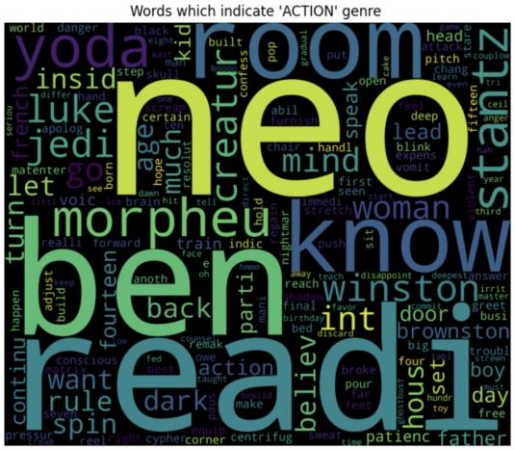
library treats the word cloud as a WordCloud object and needs to be installed via the pip command:

```
In [24]: pip install wordcloud
```

Select 'drama' 'action' 'comedy' to make a wordcloud image, create a WordCloud object, use the .generate() method to load the text. Calculate the word frequency, output the wordcloud file and display the image using pyplot.

```
wordcloud1 = WordCloud(background_color='gray', width=3000, height=2500).generate(drama)
plt.figure(figsize=(8,8))
```





LSTM

Text data processing

Tokenizer is a class for vectorizing text, or converting text into a sequence (i.e. a list consisting of the subscripts of words in a dictionary, counting from 1).list of sequences, each sequence in the list corresponds to a piece of input text. padding sequences pad_sequences converts sequences of length nb_smaples to (nb_samples,nb_timesteps)2Dnumpy attay. if maxlen is supplied, nb_timesteps=maxlen。

Simple RNN

Import the keras model, first construct an RNN class, add the first RNN and Dropout to prevent overfitting. If return_sequences is True, it means there is another RNN upstream and return, (batch_size,time_step,units) shape matrix, otherwise return (batch_size,units) matrix. Setting Embeding, embedding is an idea to compress a high dimensional sparse tensor into a low dimensional dense tensor for easy representation and computation.

```
model.add(Embedding(max_words,128,input_length=max_len))
model.add(LSTM(64))
```

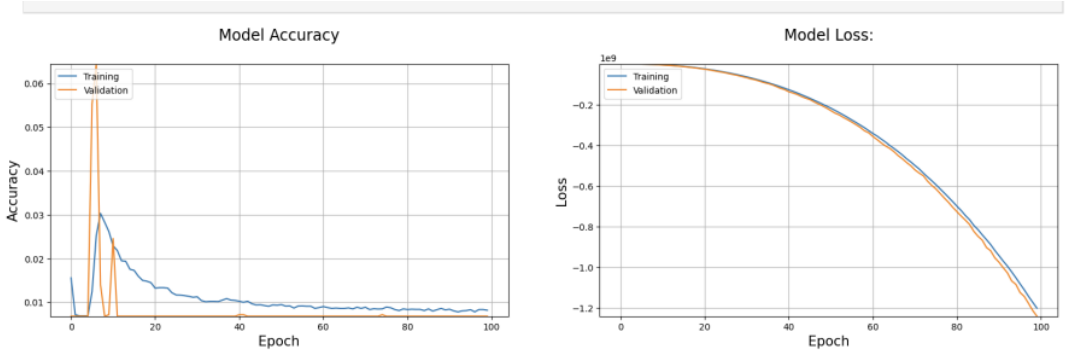
```
def plot_performance(history=None,figure_directory=None,ylim_pad=[0,0]):
    xlabel="Epoch"
    legends=["Training","Validation"]

    plt.figure(figsize=(20,5))

    y1=history.history["accuracy"]
    y2=history.history["val_accuracy"]

    min_y=min(min(y1),min(y2))-ylim_pad[0]
    max_y=max(max(y1),max(y2))+ylim_pad[0]

    plt.subplot(121)

    plt.plot(y1)
    plt.plot(y2)

    plt.title("Model Accuracy\n",fontsize=17)
    plt.xlabel(xlabel,fontsize=15)
    plt.ylabel("Accuracy",fontsize=15)
    plt.ylim(min_y,max_y)
    plt.legend(legends,loc="upper left")
    plt.grid()

    y1=history.history["loss"]
    y2=history.history["val_loss"]

    min_y=min(min(y1),min(y2))-ylim_pad[1]
    max_y=max(max(y1),max(y2))+ylim_pad[1]

    plt.subplot(122)

    plt.plot(y1)
    plt.plot(y2)

    plt.title("Model Loss:\n",fontsize=17)
    plt.xlabel(xlabel,fontsize=15)
    plt.ylabel("Loss",fontsize=15)
    plt.ylim(min_y,max_y)
    plt.legend(legends,loc="upper left")
    plt.grid()
    plt.show()
```

```
rnn_model = RNN()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 1000, 128)         20247040

 lstm (LSTM)                 (None, 64)                49408

 dropout (Dropout)           (None, 64)                0

 batch_normalization (Batch  (None, 64)                256
 Normalization)

 dense (Dense)               (None, 256)               16640

 dropout_1 (Dropout)         (None, 256)               0

 batch_normalization_1 (Bat  (None, 256)               1024
 chNormalization)

 dense_1 (Dense)             (None, 64)                16448

 dropout_2 (Dropout)         (None, 64)                0

 dense_2 (Dense)             (None, 1)                 65

=================================================================
Total params: 20330881 (77.56 MB)
Trainable params: 20330241 (77.55 MB)
Non-trainable params: 640 (2.50 KB)
_________________________________________________________________
```

Total params: 20330881 (77.56 MB) Trainable params: 20330241 (77.55 MB) Non-trainable params: 640 (2.50 KB)

The accuracy of the Validation dataset is highest at epoch 6, and the accuracy of the Training datase is highest at around epoch 10, and then both gradually decrease together.The Model Loss curve is similar for both datasets, with a gradual decrease.



Sklearn-Naive Bayes

This is a probabilistic model based on Bayes' theorem, which assumes that all features are conditionally independent from each other. This assumption makes the Naive Bayes model computationally and learning efficient, and also makes it perform well in natural language processing tasks such as text classification and sentiment analysis.

Fitting Naive Bayes to the training set and predicting the results of the test set, the Accuracy score is: 89.01% using sklearn.metrics module.

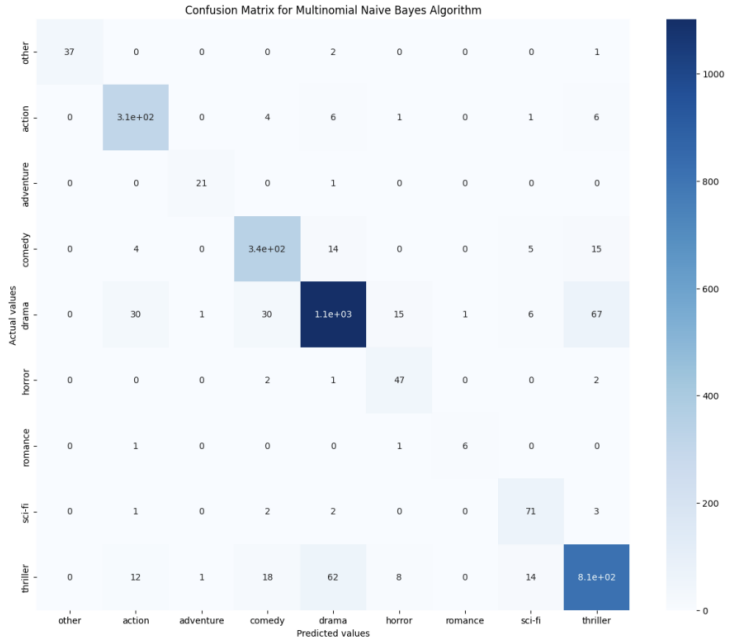
```
print("---- Score ----")
print("Accuracy score is: {}%".format(round(score1*100,2)))
```

```
---- Score ----
Accuracy score is: 89.01%
```

The confusion matrix then aggregates the records in the dataset in a matrix form according to two criteria: the true category and the category judgment predicted by the classification model. The rows of the matrix represent the true values and the columns of the matrix represent the predicted values. Accuracy is calculated using a test set and Accuracy score is: 89.01%. Import the confusion matrix and evaluate it. Get the following parameters, which are derived from the plt export graph.

```
nb_cm
```

```
array([[  37,    0,    0,    0,    2,    0,    0,    0,    1],
       [   0,  312,    0,    4,    6,    1,    0,    1,    6],
       [   0,    0,   21,    0,    1,    0,    0,    0,    0],
       [   0,    4,    0,  344,   14,    0,    0,    5,   15],
       [   0,   30,    1,   30, 1102,   15,    1,    6,   67],
       [   0,    0,    0,    2,    1,   47,    0,    0,    2],
       [   0,    1,    0,    0,    0,    1,    6,    0,    0],
       [   0,    1,    0,    2,    2,    0,    0,   71,    3],
       [   0,   12,    1,   18,   62,    8,    0,   14,  814]])
```



Adjusting the hyperparameters of Naive Bayes classifier, the adjusted Accuracy score for alpha=0.1 is: 91.18%

```
    alpha_val = i
print('---------------------------------')
print('The best accuracy is {}% with alpha value
```

```
Accuracy score for alpha=0.1 is: 91.18%
Accuracy score for alpha=0.2 is: 90.66%
Accuracy score for alpha=0.3 is: 90.37%
Accuracy score for alpha=0.4 is: 90.01%
Accuracy score for alpha=0.5 is: 89.75%
```

Predictions

Word set the basic requirements, imported the random module and validated the test set with the model. Randomly generate textual content and film types.

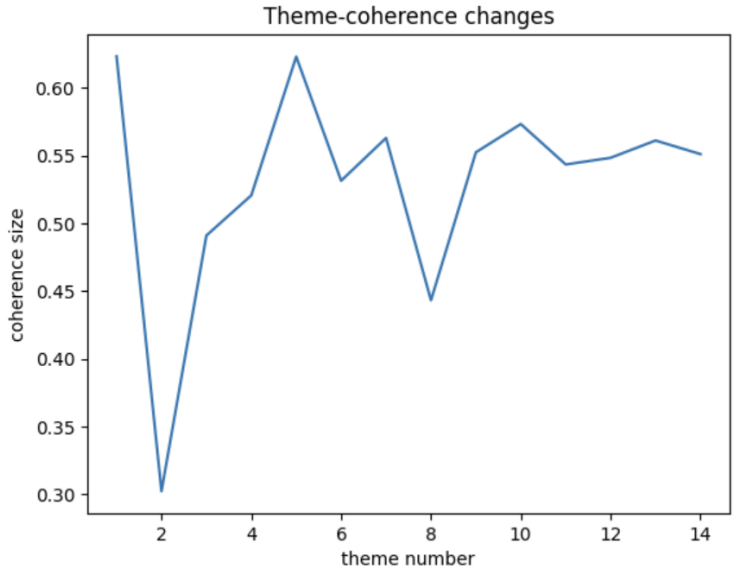
In [366…

```
# Predicting values
row = randint(0,test.shape[0]-1)
sample_script = test.text[row]

print('Script: {}'.format(sample_script))
value = genre_prediction(sample_script)
print('Prediction: {}'.format(list(genre_mapper.keys())[value]))
```

```
Script: t one. But what do I know? I like Laurel and Hardy movies. DANA Really? I
never really cared for those. Why does the fat one always have to be so mean to t
he skinny one? INT. ENID'S ROOM EVENING It's 9 30 PM. Enid is drawing in her sket
chbook. She looks impatiently at the phone. Time passes it's 11 PM. She can't sta
nd it anymore. INT. SEYMOUR'S APARTMENT CONTINUOUS Seymour picks up the phone. Da
na is in the background getting some ice in the kitchen. SEYMOUR Uh... hello? ENI
D Hi, it's me... SEYMOUR Oh, hi... ENID So, what happened? SEYMOUR almost whisper
ing Actually, it's kind of still happening... she's over here right now... I thin
k everything's going pretty well... ENID What? You're kidding me... SEYMOUR Yeah,
so I better go it's not really the best time to talk... ENID What, are you going
to like have sex with her on your first date? SEYMOUR Jesus, Enid... I'll talk to
you later... bye! He hangs up. Enid is stunned... Now what? She calls Rebecca. IN
T. OOMIE'S LIVING ROOM CONTINUOUS
Prediction: drama
```

LDA

LDA (Latent Dirichlet Allocation) is a document topic generation model, also known as a three-layer Bayesian probabilistic model, which contains three layers of words, topics, and documents. The so-called generative model, that is, we believe that each word in an article is obtained through a process of "selecting a certain topic with a certain probability, and selecting a certain word from this topic with a certain probability". Documents to topics follow a polynomial distribution, and topics to words follow a polynomial distribution.

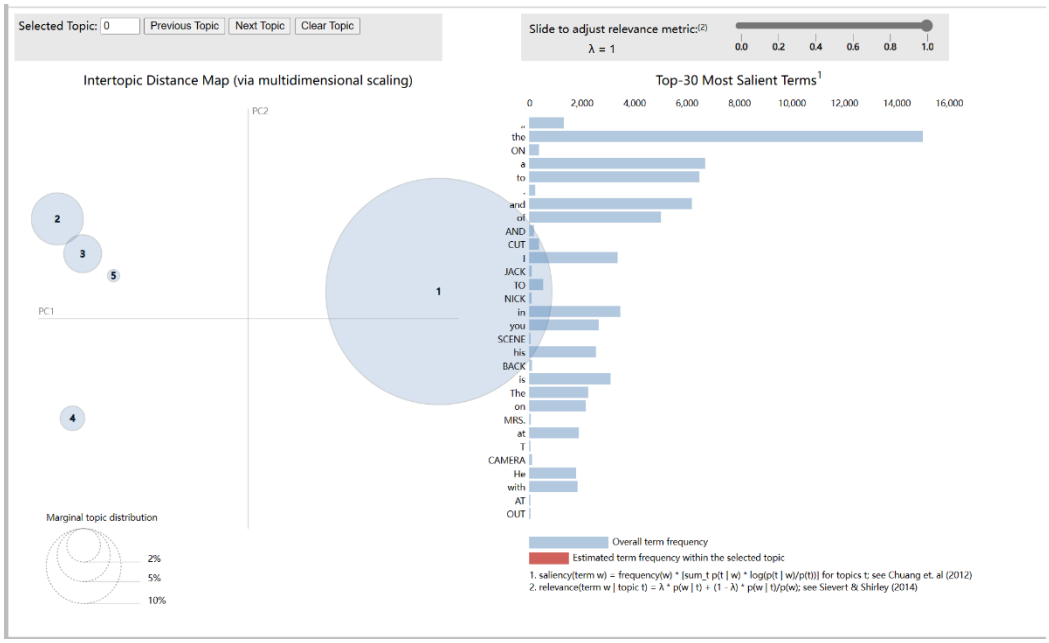


The 'stop word' package is called on the data to remove irrelevant high-frequency but not very semantic words, and the Gensim package is called for text mining. Semantic topics can be extracted from documents efficiently and automatically. Call Gensim in LDA to do text mining, build a document-term matrix. obfuscation model, calculate

relevance, and come up with the following picture:

```
[(0, 0.70153), (1, 0.29846901)]
Topic 0
[(',,', 0.21404079),
 ('he', 0.00032511278),
 ('you', 0.00031107728),
 ('i', 0.0002833127),
 ('is', 0.0002752074),
 ('his', 0.00023484562),
 ('on', 0.00023229515),
 ('with', 0.00021084811),
 ('her', 0.00020975889),
 ('at', 0.0002064162)]
Topic 1
[(',,', 0.02332261),
 ('i', 0.0003429208),
 ('you', 0.0003190586),
 ('his', 0.00029097806),
 ('is', 0.00027563752),
 ('he', 0.0002481566),
 ('on', 0.0002353534),
 ('it', 0.00022653308),
 ('as', 0.0002137504),
 ('her', 0.00019687579)]
```

Character ratios from LDA training



Visualisation image topic.html

Comparison between the bag-of-words model and TF-IDF

The main idea of TF-IDF: If a word or phrase appears in an article with a high Term Frequency (TF), and has a high word frequency and rarely appears in other articles, it is considered that this word or phrase has a good ability to distinguish between categories and is suitable for classification. The bag-of-words approach does not take into account the order of words, which simplifies the complexity of the problem and provides an opportunity for model improvement. Each document represents a probability distribution consisting of some topics, and each topic represents a

probability distribution consisting of many words.

```
vectorizer = CountVectorizer(min_df=1)
X = vectorizer.fit_transform(corpus)
feature_names = vectorizer.get_feature_names_ou
# print(feature_names,len(feature_names))
X_feature = X.toarray()
print(X_feature,len(X_feature))
print(data.shape)
```

```
[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]] 15470
(15470, 4)
```

bag-of-words model

```
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]] 15470 55778
```

TF-IDF Model

Sentence Transformers

Sentence Transformers is a Python library that can be used for sentence, text and image embeddings. Text embeddings can be computed for more than 100 languages and they can be easily used for common tasks such as semantic text similarity, semantic search, and synonym mining. Inference using Pretrained Model to obtain embedding vectors ... Select three sentences with similar subject, predicate based on three partial film lines, two dramas and one comedy as predicted by plain Bayes. And obtain the distance of similarity as 0.76, 0.55,0.51.

```
from sentence_transformers import SentenceTransformer, util
model = SentenceTransformer('distilbert-base-nli-mean-tokens')

sentences = [
    'She cant stand it anymore.',
    'I dont see it as leaving. ',
    'You ll get used to it.'
    ]
sentence_embeddings = model.encode(sentences)

for sentence, embedding in zip(sentences, sentence_embeddings):
    print("Sentence:", sentence)
    print("Embedding:", embedding)
    print("")
```

```
Sentence: She cant stand it anymore.
Embedding: [-4.96767879e-01 -4.66517568e-01  7.31650949e-01  1.36780515e-01

Sentence: I dont see it as leaving.
Embedding: [-8.22446048e-01  1.22109234e-01  6.15323186e-01 -2.15633675e-01


Sentence: You ll get used to it.
Embedding: [-2.74016678e-01 -5.09460628e-01  8.23282540e-01 -2.38889053e-01
```

```
from scipy.spatial import distance
print(1 - distance.cosine(sentence_embeddings[0], sentence_embeddings[1]))
print(1 - distance.cosine(sentence_embeddings[0], sentence_embeddings[2]))
print(1 - distance.cosine(sentence_embeddings[1], sentence_embeddings[2]))
```

```
0.7611411809921265
0.5598791241645813
0.5903242826461792
```

Conclusion

With the development of Artificial Intelligence, Natural Language Processing will be everywhere in the future, RNN and LSTM as well as Naive Bayes, in my opinion, the most accurate one is the Simple Bayes, which learns the lines of different movies by using obfuscated computation, and it is more accurate in predicting the lines of the movies, and it can get more than fifty-five percent similarity by using SentenceTransformers to analyse the similarity of the underlying semantic text. The semantic text similarity of the base of the analysis can also be obtained more than fifty-five per cent similarity.

LDA is suitable for text cleaning, call different packages in LDA, such as genism, history, etc., to analyse what is the commonality of high-frequency words, and if possible, import into Naive Bayes, use sklearn and other modules for deep learning, and then use high-frequency vocabulary to predict new movie text.