CSCI-3202

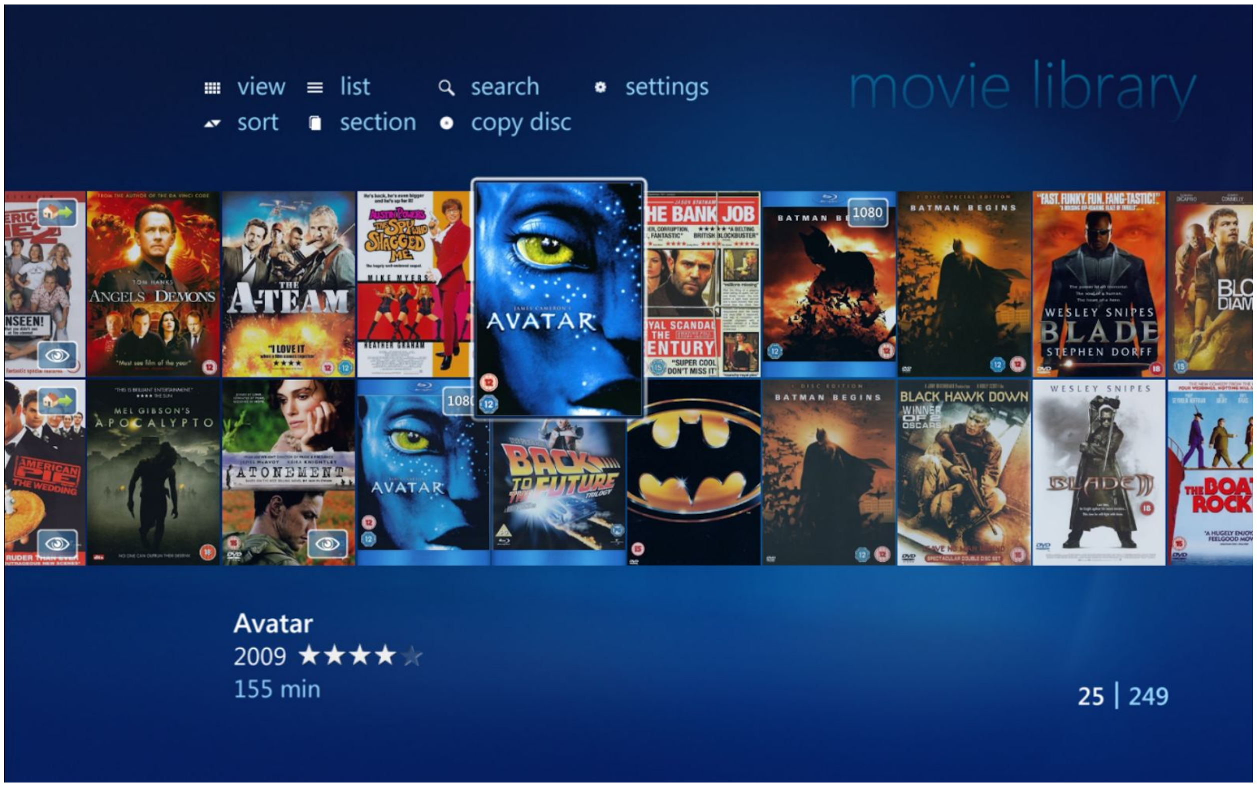
Final-report

Zhengwu Yuan

5/1/2019

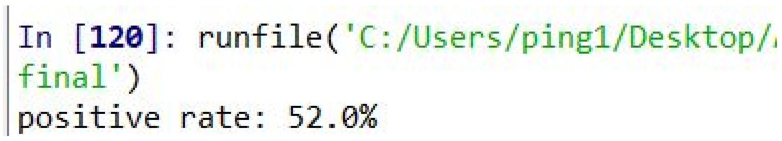
Final report

You might have a feeling that it's hard to find a good movie nowadays that make you feel like it's worth your time watching. That's totally normal since we have too many movies in 2019 and many more will affect the average score of a movie that makes it is not reliable for you to judge a movie. Reddit user profound_whatever has been a reader at five different companies and has done coverage of 300 specs, only recommending eight and putting 89 through for "consideration." profound_whatever decided to make this infographic to break everything down in terms of what went wrong with all of those scripts and what they were like. Of course a screenplay by someone famous may get an extra boost, but for the most part, readers are stuck going through an endless mass of words to find those gems that eventually make it.[1] When you try to find a good movie to see, but you want more judge factor rather than the average score of the movie. Such that for some people do not give points seriously and you want to determine if that's a good movie by the real reviews and comments that people leave. I would definitely do it in this way but it will take a lot of time to see if the majority of the reviews are positive or negative. So I want to build a probability model that can give the user the percentage of positive and negative reviews in a second to help to decide if the movie it’s worth to see.



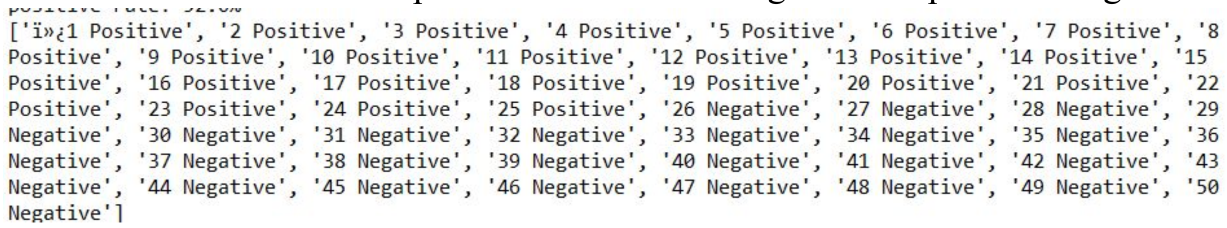
My proposed method for solving the problem is to use Naive Bayes to solve this problem. But why am I using Naive Bayes to implement the classifier for the movie review? Becuase I can compute the probability of each sentence is positive or negative in a fast and accurate way. Naive Bayes is a family of probabilistic algorithms that take advantage of probability theory and Bayes’ Theorem to predict the tag of a text (like a piece of news or a customer review). They are probabilistic, which means that they calculate the probability of each tag for a given text, and then output the tag with the highest one. The way they get these probabilities is by using Bayes’ Theorem, which describes the probability of a feature, based on prior knowledge of conditions that might be related to that feature.we can assume that each word in a sentence is independent with each other. So that we are not looking at the entire sentence, but rather at each individual words.[2] First, convert all the text to lower case, remove the special character, and spamming words which I define it as tokenize to improve the accuracy of the predicted outcomes. And create a dictionary to store negative and dataset. Then create three dictionaries stores the total words, and counting positive words and negative words. Using naive Bayes to calculate the probability and using Laplace smooth to get the probability for each word by add 1 smoothing to increase the zero probability values to a small number. And then use naive Bayes to classify and get whether it is positive or negative by the larger probability to predict wheater it’s positive or negative and count the positive rate in the case for any other movies. And last using an accuracy metric to check how good is the model that I build. For the data, I find it on kaggle that is the IMDB dataset that contains the movie reviews.[3] I have got the positive reviews-training dataset and the negative reviews-training dataset and creating a test set by picking 25 of the positive reviews and 25 of the negative reviews from the test set in kaggle as it’s going to be easier to check the accuracy of our predicted value. Also, a correct solution set that used to check the accuracy by using sklearn.metrics-accuracy_score.

The results of my problem that has a 90% of accuracy using an accuracy metric which return an accuracy classification score. In multilabel classification, this function computes subset accuracy: the set of labels predicted for a sample must exactly match the corresponding set of labels in y_true. [4] For instance, I will say that my implementation is successful.Since that the confidence of my probabilistic classifier has a high accuracy rate that’s more than 70% and exceed the baseline of knowing what a “good” accuracy is.[5] I put 25 of positive movie reviews and 25 of negative movie reviews and it’s easy to tell that the positive rate is 50%. And as after my implementation, it gave a 52% positive rate for predicted movie reviews and that’s close to what it should be.



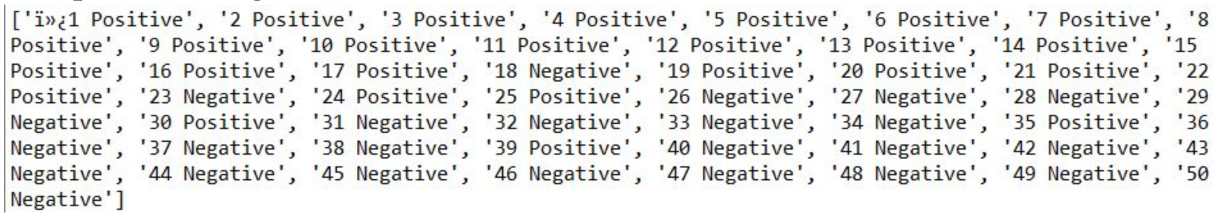
```
In [120]: runfile('C:/Users/ping1/Desktop/
final')
positive rate: 52.0%
```

Let’s take a look at the comparison of the correct tags and the predicted tags:



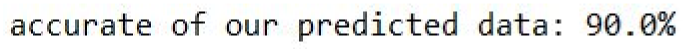
```
['ï»¿1 Positive', '2 Positive', '3 Positive', '4 Positive', '5 Positive', '6 Positive', '7 Positive', '8
Positive', '9 Positive', '10 Positive', '11 Positive', '12 Positive', '13 Positive', '14 Positive', '15
Positive', '16 Positive', '17 Positive', '18 Positive', '19 Positive', '20 Positive', '21 Positive', '22
Positive', '23 Positive', '24 Positive', '25 Positive', '26 Negative', '27 Negative', '28 Negative', '29
Negative', '30 Negative', '31 Negative', '32 Negative', '33 Negative', '34 Negative', '35 Negative', '36
Negative', '37 Negative', '38 Negative', '39 Negative', '40 Negative', '41 Negative', '42 Negative', '43
Negative', '44 Negative', '45 Negative', '46 Negative', '47 Negative', '48 Negative', '49 Negative', '50
Negative']
```

And predicted tags:



```
['ï»¿1 Positive', '2 Positive', '3 Positive', '4 Positive', '5 Positive', '6 Positive', '7 Positive', '8
Positive', '9 Positive', '10 Positive', '11 Positive', '12 Positive', '13 Positive', '14 Positive', '15
Positive', '16 Positive', '17 Positive', '18 Negative', '19 Positive', '20 Positive', '21 Positive', '22
Positive', '23 Negative', '24 Positive', '25 Positive', '26 Negative', '27 Negative', '28 Negative', '29
Negative', '30 Positive', '31 Negative', '32 Negative', '33 Negative', '34 Negative', '35 Positive', '36
Negative', '37 Negative', '38 Negative', '39 Positive', '40 Negative', '41 Negative', '42 Negative', '43
Negative', '44 Negative', '45 Negative', '46 Negative', '47 Negative', '48 Negative', '49 Negative', '50
Negative']
```

We can see that the majority of the predicted tags matches the correct tags which showing that my implementation was on the right track. By importing module from sklearn.metrics, the predicted tags have a 90% accuracy compare to the correct tags



```
accurate of our predicted data: 90.0%
```

In conclusion, I would say this implementation of Naive Bayes of the project is successful and I have learned a lot from it especially in the understanding of Naive Bayes and how to use it for text classification. Not only for movie reviews but much other recognition of dataset since the Navie Bayes model are working extremely well in this set of problems that you might face in the future. But to get a decent accuracy rate, we will need to do a lot of work to prepare for the training data such as remove stopwords, punctuation, lower all the sentence, remove the blank space and line and so on. Becuase I didn't remove the stopwords and I guess that's part of the reason that the accuracy is only 90% and wasn't that perfect. Also while implementing a Naive Bayes model, you should always use Laplace Smoothing in the case for a word or observation isn't in the data to not have a probability of zero to have more precise and accurate results from Naive Bayes model.

References

[1]Lily Hay Newman and Lily Hay Newman. 2013. Here's Why It's So Hard to Find Good Movies. (November 2013). Retrieved May 1, 2019 from https://gizmodo.com/heres-why-its-so-hard-to-find-good-movies-1470678245

[2]Anon. 2018. A practical explanation of a Naive Bayes classifier. (November 2018). Retrieved May 1, 2019 from https://monkeylearn.com/blog/practical-explanation-naive-bayes-classifier/

[3] Arunava. 2018. IMDB Movie Reviews Dataset. (July 2018). Retrieved May 1, 2019 from https://www.kaggle.com/iarunava/imdb-movie-reviews-dataset/version/1

[4]Anon. sklearn.metrics.accuracy_score¶. Retrieved May 1, 2019 from https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html#sklearn.metrics.accuracy_score

[5]Stephen Hutt. IntroToAI_19 Naive Bayes. Retrived May 1,2019 from IntroToAI_19 Naive Bayes.pdf