CENG463 - Assignment 1 Report

Ali Dogan

December 18th 2020

1 Implementation

1.1 Representation of data: preprocessing, cleaning, tokenization

I have started the data preprocessing in the text files. I have cleaned the unintended white spaces that cause imperfect data for extraction, links that cause special characters, and partially the unspaced characters between end of the sentences. I have written *data_prep.y* file to clean the data and then create train-dev-test sets. Then I continued my process by tokenizing and stemming. I have implemented a function in *utils.py* file named $preprocess(data)$ which takes the tokenized data which is done by another function I wrote named $word\_tokenize\_all(..)$ in the same file and then implements stemming and stopword cleaning. I chose to use stopword cleaning because there was a slight improvement in the accuracy.

1.2 Classifiers

I have used Naive Bayes and MaxEnt classifiers. I tried Bernoulli Naive Bayes classifier as well but I was getting the same accuracies as Naive Bayes with a very little difference, and the most informative features part were always the same. So I chose to implement MaxEnt to experience different models. I have also tried SVM but I was getting results similar to maxEnt and It was taking to much time with high dimensional feature set so I did not pursue on SVM. For the Naive Bayes implementation, the best results are achieved when the number of word features were high as 1500-3000 whereas in Maxent model, that much high dimensionality caused overfitting very quickly. I achieved the best results in Maxent with small word features such as 50-100-150 and I got around 0.6 accuracy in the test and dev set. 5 to 10 Number of iterations for MaxEnt model was enough based on my experiences.

1.3 Feature Extractions

I have implemented several feature extraction functions in feature_functions.py file to use them in Classifier class. I got the best results from the first implementation which is basic word feature implementation. It is creating a feature set which contains the first N most used word, and given a book entity, it shows whether that common words exist in that entity or not. I also implemented another feature set which contains only 10 features. it is in the following form:

$$feature\_extraction\_2 = \{len(title), len(description), word\_count(horror), word\_count(mystery)....\}$$

where

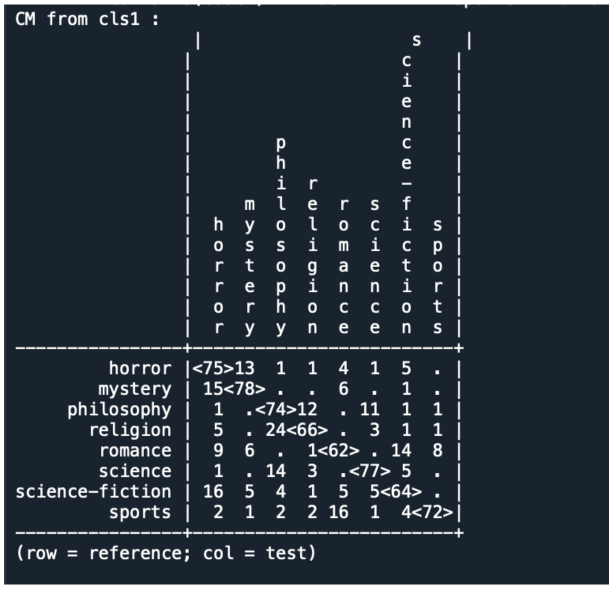
$$word\_count(genre) = \sum_{w \in book} \{number\ of\ occurances\ of\ w\ in\ the\ genre\}$$

I thought that this feature is very promising, however it did not give good results such that it was nearly insignificant of having the word counts. I thought the problem was in the normalization and I developed the 3rd feature extraction function but that did not give good results as I hoped. The accuracy was around 0.25.
So I sticked with the first feature extraction model I implemented.

2 General Questions

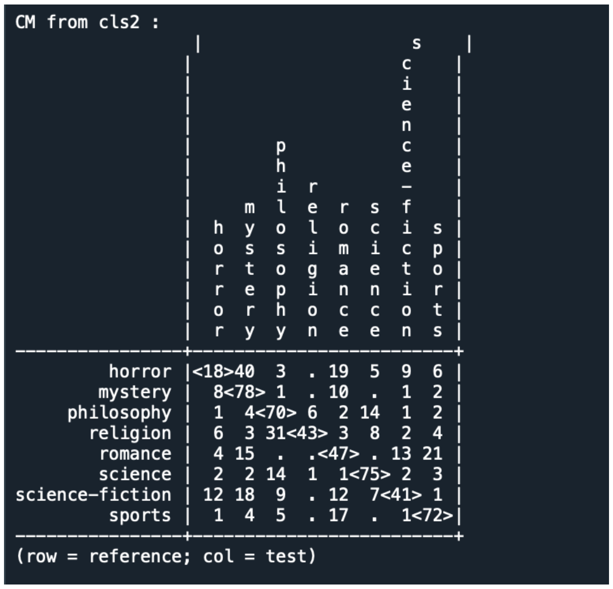
2.1 Confusion matrices

I have written a function in Classifier to construct confusion matrices. The two confusion matrices are appended here one for each model. For the naive model, feature extraction has 2000 most common words feature which gave 0.71 accuracy on test and dev sets.



```
CM from cls1 :
                |                     s    |
                |                     c    |
                |                     i    |
                |                     e    |
                |                     n    |
                |        p            c    |
                |        h            e    |
                |        i  r         -    |
                |     m  l  e  r  s  f    |
                |  h  y  o  l  o  c  i  s |
                |  o  s  s  i  m  i  c  p |
                |  r  t  o  g  a  e  t  o |
                |  r  e  p  i  n  n  i  r |
                |  o  r  h  o  c  c  o  t |
                |  r  y  y  n  e  e  n  s |
----------------+-------------------------+
         horror |<75>13  1  1  4  1  5  . |
        mystery | 15<78> .  .  6  .  1  . |
     philosophy |  1  .<74>12  . 11  1  1 |
       religion |  5  . 24<66> .  3  1  1 |
        romance |  9  6  .  1<62> . 14  8 |
        science |  1  . 14  3  .<77> 5  . |
science-fiction | 16  5  4  1  5  5<64> . |
         sports |  2  1  2  2 16  1  4<72>|
----------------+-------------------------+
(row = reference; col = test)
```

Figure 1: Naive Bayes Model Confusion Matrix on test data



```
CM from cls2 :
                |                     s    |
                |                     c    |
                |                     i    |
                |                     e    |
                |                     n    |
                |        p            c    |
                |        h            e    |
                |        i  r         -    |
                |     m  l  e  r  s  f    |
                |  h  y  o  l  o  c  i  s |
                |  o  s  s  i  m  i  c  p |
                |  r  t  o  g  a  e  t  o |
                |  r  e  p  i  n  n  i  r |
                |  o  r  h  o  c  c  o  t |
                |  r  y  y  n  e  e  n  s |
----------------+-------------------------+
         horror |<18>40  3  . 19  5  9  6 |
        mystery |  8<78> 1  . 10  .  1  2 |
     philosophy |  1  4<70> 6  2 14  1  2 |
       religion |  6  3 31<43> 3  8  2  4 |
        romance |  4 15  .  .<47> . 13 21 |
        science |  2  2 14  1  1<75> 2  3 |
science-fiction | 12 18  9  . 12  7<41> 1 |
         sports |  1  4  5  . 17  .  1<72>|
----------------+-------------------------+
(row = reference; col = test)
```

Figure 2: MaxEnt Model Confusion Matrix on test data

2.2 Precision, recall, F-score, accuracy scores

I have implemented a function in classifier class to call the precision and recall scores :

2.2.1 Naive Bayes Model

In the order : (philosophy, romance, science-fiction, horror, science, religion, mystery, sports)
Precision : 0.74, 0.62, 0.64, 0.75, 0.77, 0.66, 0.78, 0.72
Recall : 0.62, 0.66, 0.67, 0.60, 0.78, 0.76, 0.75, 0.87
F-score : 0.68, 0.64, 0.66, 0.67, 0.78, 0.71, 0.77, 0.79
Accuracy : 0.71

2.2.2 MaxEnt Model

In the order : (philosophy, romance, science-fiction, horror, science, religion, mystery, sports)
Precision : 0.7, 0.47, 0.41, 0.18, 0.75, 0.43, 0.78, 0.72
Recall : 0.52, 0.42, 0.58, 0.34, 0.68, 0.86, 0.47, 0.64
F-score : 0.6, 0.45, 0.48, 0.24, 0.72, 0.57, 0.59, 0.68
Accuracy : 0.55

3 Analysis of errors with confusion matrices

At the Naive Bayes Confusion Matrix, we can point to couple of conclusions. First, there is a significant number of mislabelling between history and mystery in both ways. Beside the natural similarity between these two genres, there are actually same book data stored in both history and mystery files. I have seen the same books while I was manually arranging the dataset. I believe there are a significant number of them even though I did not do a deeper analysis.

The other point we can look is between philosophy and religion. These two topics are also similar in natural sense. A significant portion of the topic religion can be seen as a subtopic of philosophy. So, there is a possibility that there are same books labelled as both. And due to word features, it is possible to encounter same word frequencies.

4 Conclusion and Comments

In conclusion I have achieved 0.7 accuracy on test data with naive bayes model by using the most common 2000 words as features. With MaxEnt model, I have achieved around 0.55 accuracy by using the most common 200 words as features. I think there is room for improvement in both models.

For Naive Bayes approach, the crucial improvement, I believe, can be achieved by more data cleaning and tokenizing. Maybe after that, the synonyms can be used to merge two synonym word groups or adding other helpful features might be helpfull. Or another approach can be adding some features to solve specific problems such as history-mystery or philosophy-religion mislabeling. Also, having the same book entity in more than one dataset is also a problem since we are only predicting one label for each sample.

For MaxEnt model, the biggest problem is the dimensionality of the features. that is why using a bigger feature set resulted in bias variance, on the other hand using fewer word fetures or other feature extraction method such as the word count approach resulted in overfitting. I think selecting the featureset carefully is much more important in maxent than it is in naive bayes approach. This model may work better using a mapping of each word into categories like a dimensionality reduction instead of using directly the word features.