CMPUT 497: Assignment 5 Report Paper

Relation Extraction & Classification

Eden Zhou
University of Alberta
Edmonton, Canada
zhou9@ualberta.ca

Menghan Chen
University of Alberta
Edmonton, Canada
menghan4@ualberta.ca

OVERVIEW

Based on the supervised learning approach and the given datasets, a *Naive Bayes (NB)* classification model was constructed with a few relevant heuristics approaches for relation extraction and classification tasks in this project.

This report paper describes the ideas learned from the model prediction results (especially from errors and fairness estimations) and explains how the evaluation part has been processed with diverse analysis standards. An extension part about how further improvements to the model can be landed is also attached as the last section.

DATA

Given two datasets as training and test respectively with sizes of 2000 and 400. The training dataset was used to train the model and the test set processed presents the model accuracy extent.

METHODS

In this project, the *k-fold cross-validation* approach was applied to control data learning scopes and ensure multi-models trained parallel. We have the setting of k=3 which means all data in the given training set were divided into three parts uniformly after processing a shuffle. For each fold round, one NB model object will be created, with (k-1) parts in all k picked for training and the single part left will be used for the development test.

This algorithm will repeat the same process for k times and make sure all data is shuffled appropriately with different parts being selected as the development test. Finally, all three models learned and tested on diverse training and test sets were compared to figure out the one with the best performance on its development test. This model will then be used to work on the formal test set to make real predictions.

Settings and Decisions

There are three model learning modes (*ShuffleMode*, *FilterMode* and *UltraMode*) prepared and three possible data filters (*StopWordPruning*, *ParticularPruning* and *PunctuationPruning*) implemented as basic or enhancement settings for developing the NB classifier. In this part, we would like to analyze the first two modes and the first filter listed. Other settings left are considered as enhancement approaches will be discussed later in *Extension Part: Potential Improvements* of this paper.

As we introduced before, the cross-validation method may want to shuffle data before cutting the set into chunks. So we left the setting of *ShuffleMode* there, as the answer is true, the sequence rearranging works. This is not a mandatory process since the training data we got were already random in sequence. However, this setting may provide with some hidden improvement and helps to estimate the variance in data. Compared to the fixed value of 0.880 without rearranging, activation for shows a variance up to ±0.020.

As a part of data pre-processing, the *StopWordPruning* filter in *FilterMode* helps remove all stopwords in the given raw data. Firstly, NLTK Tokenizer was applied to divide sentences into tokens to contribute to further operations. Removing is processed based on the open-source of the NLTK Stopwords Corpus: if any words in the training dataset appeared in the corpus, then our filter will take them off and then return the documents pruned. The pruning may help to improve the effectiveness of classification in model training, but this improvement is not inevitable. For this task, we cancelled stopword pruning since it can not provide any positive effects in supporting the model performance: by running *StopWordPruning*, the model saw a decrease in accuracy from around 0.886 to around 0.8675.

PERFORMANCE ANALYSIS

Based on the analysis report raised from the given test set, evaluations about errors and model performance are valuable for further improvements.

Error and Accuracy

The details of the classification result are demonstrated in Figure 01, the overall confusion matrix. The table shows that the "characters" class has the highest numerical misclassification (8 in "director", 5 in "performer" and 10 in "publisher"), and the lowest accuracy (=0.91). In contrast, the class "publisher" has the highest accuracy (=0.955) and only 3 misclassifications (2 in "characters", 1 in "performer" and no "director"). For the other two classes, the instances in the “performer” class misclassified 7 in "characters", 3 in "director" and 2 in "publisher" with an accuracy of 0.9475. The instances in the “director” class misclassified 4 in "characters", 3 in "performer" and 3 in "publisher" with an accuracy of 0.9475 as well. This is an interesting result that our classifier is unlikely to classify "publisher" into other classes, but misclassifies about 15 instances from other classes into "publisher". The instance (row_id=843) is a typical example: tokens include the novel name "Champagne for One", the character name "Nero Wolfe", and the publisher "FremantleMedia Enterprises". It should be classified into the "characters" class but misclassified into "publisher" due to the extra entity. Also, the character's name is seen as the name of episodes in the tokens, which increases the misunderstanding. Having several relations due to extra entities caused the majority of the mistakes in the test set. However, there are still some mistakes that didn't meet the previous situation. While reading through them, we cannot clearly figure out their correct classes if we don't know the content of the work. This kind of mistake always appears when two relations have a similar entity, like in the "performer" and the "characters" relation, they may all be constructed by a person's name and a work. This kind of mistake could be found from the instance with row_id 2687.

Confusion Matrix

[Confusion Matrix]

Predicted / Actual	characters	director	performer	publisher	All
characters	80	8	5	10	103
director	4	84	3	3	94
performer	7	3	91	2	103
publisher	2	0	1	97	100
All	93	95	100	112	400

Figure 01: *Overall confusion matrix. The matrix illustrates potential gaps between the model prediction outputs and golden truth labels. Conditions for all four classes (classes=[‘director’, ‘performer’, ‘publisher’, ‘characters’]) listed and the sum data amount in the given test set present.*

For the “director” class, the precision was about 0.884, the recall was about 0.894 and the F-score was 0.889. For the second class of “performer”, the precision was about 0.91, the recall was 0.883 and F-score was 0.897. For the “publisher” class, the precision was about 0.866, the recall was 0.97 and F-score was 0.955. It performed the best out of all classes with the highest recall value and the almost best precision value

(=0.864 compared to =0.870 in "performer"). The last class is "characters" which has the precision of about 0.860, the recall of 0.777 and F-score of 0.816. This class came with the lowest recall value in all four.

```
* Class --> director
------------------------------------------------------------------
                              True director          True not
System director                84                       10
System not                     11                      295
------------------------------------------------------------------
TP=84   FN=10   FP=11   TN=295          --> Sum=400
precision:  0.8842105263157894
recall:  0.8936170212765957
accuracy:  0.9475
F1:  0.8888888888888888


* Class --> performer
------------------------------------------------------------------
                              True performer         True not
System performer               91                       12
System not                     9                       288
------------------------------------------------------------------
TP=91   FN=12   FP=9    TN=288          --> Sum=400
precision:  0.91
recall:  0.883495145631068
accuracy:  0.9475
F1:  0.896551724137931


* Class --> publisher
------------------------------------------------------------------
                              True publisher         True not
System publisher               97                       3
System not                     15                      285
------------------------------------------------------------------
TP=97   FN=3    FP=15   TN=285          --> Sum=400
precision:  0.8660714285714286
recall:  0.97
accuracy:  0.955
F1:  0.9150943396226415


* Class --> characters
------------------------------------------------------------------
                              True characters        True not
System characters              80                       23
System not                     13                      284
------------------------------------------------------------------
TP=80   FN=23   FP=13   TN=284          --> Sum=400
precision:  0.8602150537634409
recall:  0.7766990291262136
accuracy:  0.91
F1:  0.8163265306122449
```

Figure 02: *Class-based confusion matrices. Model Precision and Recall values based on the input test set that works for all four classification cases (classes=['director', 'performer', 'publisher', 'characters']). Relevant statistic data were shown in the confusion matrices of each class.*

```
[Overall Pooled Matrix]
------------------------------------------------------------------
                    True yes         True not
System yes      352.0                48.0
System not      48.0                 1152.0
------------------------------------------------------------------
(Overall) TP=352.0      FN=48.0 FP=48.0 TN=1152.0   --> Sum=1600.0

[Micro-average]
precision:  0.88
recall:  0.88
F1:  0.88

[Macro-average]
precision:  0.8801242521626648
recall:  0.8809527990084693
F1:  0.8792153708154266
```

Figure 03: *Overall precision and recall computation based on two different standards: The Micro-averaged and the Macro-averaged. The pooled matrix created for computing the micro-averaged.*

Filter Performance

For taking a close look at different filters' functionality, each filter is available to be provided individually in training the model.

With only providing *StopWordPruning* as the filter, the overall result shows a significant decline. Micro-average has only 0.8675 now compared to 0.88 with applying all filters. Macro-average declines of around 0.015 for each analysis data as well. This result shows that using *StopWordPruning* can bring improvement to the classifier.

If *ShuffleMode* is provided, the result is not stable due to randomization in the shuffle process. Sometimes it shows a slight improvement: the micro-average can increase to 0.885 from 0.88, and the macro-average can increase by 0.008 in precision, 0.004 in recall and 0.006 in F1. Sometimes shows a worse result than without applying *ShuffleMode*. For observing a steady result, we decided not to apply it in the program.

EXTENSION PART

Following ideas introduced in the early section, a few new modes and filters were established to pursue further improvements in model performance.

Notable Attempts in Filters

Two other heuristic methods are developed for data pre-processing, *ParticularPruning* and *PunctuationPruning*. Because the given data set delivers key indexes of heads and tails for each document, a more creative filter called *ParticularPruning* was built up to remove unimportant parts of sentences for clearer connection-identifying in influential words remained. In this filter, only the head and tail words with the connection words between can be kept with the original sequence, and all others will be removed. In practice, this measure brings an F1-value of about 0.7525 for both micro- and macro- standards.

PunctuationPruning is another shot at removing unnecessary information to improve learning

quality. It selectively drops punctuation marks in words and finally leads to an F1-value of approximately 0.8675 on both standards. We would like to analyze its error conditions since *PunctuationPruning* filter works better than other attempts.

The difference between applying *PunctuationPruning* and *ParticularPruning* is that *PunctuationPruning* removes less information than *ParticularPruning*. Therefore, the information removed in *ParticularPruning,* which is the tokens that are not between heads and tails, is useful in the relation extraction for the classification. Additionally, applying *PunctuationPruning* individually results in the same F1-score as applying the *StopWordPruning* that was analyzed previously in the Filter Performance. However, when applying them together will lead to an F1-score of approximately 0.8625, which declines more than applying them respectively. This result shows these two heuristic filters are eliminating pieces of information that are useful in Naïve Bayes classification.

Potential Improvement in Method

In addition to heuristic filters, another simple attempt in running mode was also constructed. *UltraMode* focus on approaching the upper limit of the distribution variance led by the k-fold cross-validation approach. It uses a long loop to cover the whole process of cross-validation (with shuffles in each fold round) and repeats the cross-validation learning processes for t times. The method will then pick the model with the best performance out as the formal one. Its error issues are similar to the originals, but it brings more effort in approaching the upper bound of the floating variance in shuffle-based training.

REFERENCES

[1] NLTK Tokenizor Documentation

[2] NLTK StopWords Corpus Documentation

[3] Speech and Language Processing. 2021

Chapter 18. Daniel Jurafsky & James H. Martin.