Weicong Feng

Professor Alla Rozovskaya

CS74040 NLP

10/24/2021

Homework 2

Report

1. [Movie review classification using Naïve Bayes - 10 points]

Assume that you have trained a Naïve Bayes classifier for the task of sentiment classification (Please refer to Chapter 4 in the J&M book). The classifier uses only bag-of-word features. Assume the following parameters for each word being part of a positive or negative movie review, and the prior probabilities are 0.4 for the positive class and 0.6 for the negative class.

Table

Description automatically generated

Question: What class will Naive Bayes assign to the sentence “I always like foreign films”? **Show your work.**

Ans:

P( “I always like foreign films” | pos ) = P(pos)

= 0.4 \* 0.09 \* 0.07 \* 0.29 \* 0.04 \* 0.08 = 0.00000233856

P( “I always like foreign films” | neg ) = P(neg)

= 0.6 \* 0.16 \* 0.06 \* 0.06 \* 0.15 \* 0.11 = 0.0000057024

According to the result, the Naïve Bayes classifier will assign negative to “I always like foreign films”.

2. The system consists of two classes, process and NB, which are stored in two files, preprocess1.py and NB.py, respectively. The process class stored in preprocess2.py is a specific version of process for b) and c) questions.

1) Interface

The process class is responsible for lowercase, tokenization and cleaning data, while the NB class takes charge of counting the words, likelihood of documents, prior possibilities, predicting the class or validate the test data when the test data includes labels.

2) Data exchange

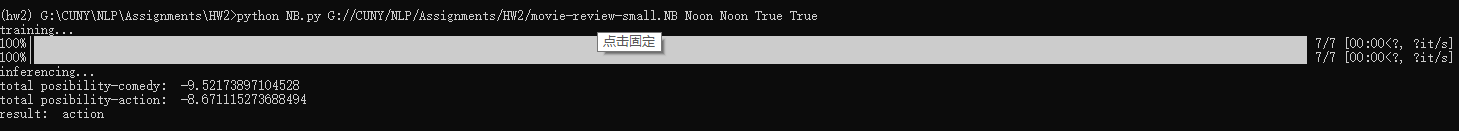
The data is dump and load with whole object by pickle. After loading the data, including training data and test data, the data is stored as properties of the object. The cleaned tokens are all stored as properties of the object. These results are saved altogether with the object to a file called movie\_review\_big.NB. This file will be loaded as a object and passed to NB object when NB is instantiated.

3) Data cleaning

All the training review documents are combined to one document, while the test documents are kept separate after loading data. All of them are lowercased, split by space. Then they are removed punctuation, digits, html tags, stop words. On the other hand, I combined the word types encountered in training review documents and vocabulary file which comes with assignment package.

4) Result

a) The result of b) and c) is action.



b) In the test data set, system precision is 0.8228.

Text

Description automatically generated with medium confidence

5) Analysis

At the first validation, the system only returned a precision of 0.5616. The issue is the possibilities become too tiny to zero in long documents, so that these documents were assigned to negative. The correction of using log space increased the precision to 0.81316. After adding removing stop word and expanding vocabulary, the precision achieved 0.8228. It shows that both of transformation of target space and cleaning data can improve system performance.

By observing the misclassification, it can be found that 35% of the errors occurred in assigning pos to neg, and 65% of errors were due to assigning neg to pos. The reason is not clear yet. One possibility is more tokens which doesn’t encounter in test reviews in positive class than that in negative class. Because these unseen tokens were ignored in prediction process, it means 1.0 timed previous multiplied possibilities chain. On the contrary, a token in vocabulary means it a tiny possibility, such as 2.51e-7 timed previous possibilities chain.

Another thing can contribute to increase precision is lemmatization and embedding. Due to the limitation of no using professional package, stemming need to create a huge dictionary and be computation intensive. Embedding is impossible to do it without huge training data set.

On the other hand, the BOW loss the context. For example, “he is happy” has opposite meaning of “I don’t think he is happy”.

6) Use

The preprocess1.py and preprocess2.py work for different cases.

1. In the case like question 2b, one line per example, the label at the beginning of each line, and the test data without label, run the preprocess2.py on the command line:

**python preprocess2.py training\_file.py test\_file.py parameter\_file.NB**

where the training\_file.py, test\_file.py, parameter\_file.NB should include the path, for example:

*python preprocess2.py G://CUNY/NLP/Assignments/HW2/train2.txt G://CUNY/NLP/Assignments/HW2/test2.txt G://CUNY/NLP/Assignments/HW2/movie-review-small.NB*

Once the execution of preprocess2.py is done, execute the NB.py:

**python NB.py preprocess\_file\_name parameter\_file\_name output\_file\_name train pred**

where, preprocess\_file\_name: the preprocessed file, the output file of preprocess1/2.py,

including path and file name

parameter\_file\_name: the parameter file, the parameter file of NB model, including path

path and file name

output\_file\_name: the output file, the validation results, including path and file name

train: True – train the mode; False – load the model parameters

pred: True – predict the test data; False – validate the test data

For example, the preprocessed file called movie-review-small.NB, train and inference the test data:

*python NB.py G://CUNY/NLP/Assignments/HW2/movie-review-small.NB Noon Noon True True*

1. In the case of validation, like question d), where the test files are stored in folders that is named as corresponding label, run the preprocess1.py on the command line:

**python preprocess1.py training\_dir test\_dir parameter\_file\_name**

where, training\_dir: training data folder

test\_dir: test data folder

parameter\_file\_name: the parameter file, the parameter file of process object, including

path path and file name

For example:

*python preprocess1.py G://CUNY/NLP/Assignments/HW2/train G://CUNY/NLP/Assignments/HW2/test G://CUNY/NLP/Assignments/HW2/movie\_review\_big.NB*

Once the execution of preprocess1.py is done, execute the NB.py:

**python NB.py preprocess\_file\_name parameter\_file\_name output\_file\_name train pred**

where, the parameters are same as above.

For example, the preprocessed file called G://CUNY/NLP/Assignments/HW2/movie\_review\_big.NB, and the saved parameters file called G://CUNY/NLP/Assignments/HW2/movie-review-BOW.NB, we don’t want to retrain the model, only reload it, and validate:

*python NB.py G://CUNY/NLP/Assignments/HW2/movie\_review\_big.NB G://CUNY/NLP/Assignments/HW2/movie-review-BOW.NB G://CUNY/NLP/Assignments/HW2/big-result.txt False False*

Conversely, if you want to train a new model from preprocessed data and validate, you should run:

*python NB.py G://CUNY/NLP/Assignments/HW2/movie\_review\_big.NB G://CUNY/NLP/Assignments/HW2/movie-review-BOW.NB G://CUNY/NLP/Assignments/HW2/big-result.txt True False*