**Text Classification**

**Setup, Compilation, Execution and Output Guidelines**

**Sharmodeep Sarkar and Sarita Joshi**

This is an implementation code designed in python 2.7 for a naïve Bayes text categorization system to predict whether movie reviews are positive or negative

For this code to be executed on any terminal, we need to have below pre-requisites met:

1. Python 2.7 installed

Go to https://www.python.org/downloads/ -> install as per operating system

Required Python Packages-

import sys, os

import re

import pickle

from decimal import Decimal

from math import log

This entire classification is done in two phases,

1. Training the system using the provide train data – implementing bag of words, naïve Bayes classifier using add-1 (Laplace) smoothing

Implement bag of words -> retrieve words from all documents in the train data by splitting the tokens by whitespaces, sequence of white spaces and/or newline

Smoothing -> Implements Laplace smoothing after ignoring terms that are fewer than 5 times in combined positive and negative training data

1. Using this learning from previous step (A Model file), generate classification score for unseen test documents and predict the files in test directory

Phase 1:

Requirement:

File name: nbtrain.py

No. of Input Arguments: Two

1. <training-directory> -> Input

This is the supplied directory path for all files to be read for training purpose by the system

Ideally, this directory structure to be placed in same file path as that of the nbtrain.py file (Our program is robust, can take any given file path)

1. <model-file>-> Output file

This is the generated output file containing all relevant parameters after the training data is provided to the system.

Use of Python pickle to dump the data as objects to text file

Steps to be done:

Place the given python file i. e. “nbtrain.py” along with the train directory that is to be supplied to this code ‘textcat/train’ and the file name for the model file to be generated i.e. ‘model.txt’ (Local file generated at same location)

Compilation and Run Python code:

Go to terminal

Type python. This will start the python shell terminal

Navigate to the required path and run .py file as below->

->FilePath/ nbtrain.py textcat/train model.txt

Expected Output:

File Output:

‘model.txt’ file generated storing all relevant parameters required for classifying the unseen test collection

It stores –

Class Prior for Positive and Negative Class

Positive term probability, Negative term probability, Positive word occurrences, Negative word occurrences, Vocabulary length

Console Output:

Prints the learned values from the train data i.e.

The class prior for positive and negative class

Total Positive word occurrence

Total Negative word occurrence

Total Vocabulary length

Phase 2:

Requirement:

File name: nbtest.py

No. of Input Arguments: Three

1. <model-file> -> Input

This is the generated output file from the previous phase and is input to the test phase containing all relevant parameters after the system is trained and ready for classification

Use of Python pickle to load the data stored as object

1. <test-directory> -> Input

This directory structures contains the list of unseen documents that are to be classified after calculating the positive and negative document score

1. <predictions-file> -> Output

This is the generated output file which stores the list of all files present in the test directory along with the calculated positive and negative score

It also stores the count of files classified along with the file names for both positive and negative class

Steps to be done:

Place the given python file i. e. “nbtest.py” along with the model file ‘model.txt’, test directory that is to be supplied to this code ‘textcat/test’ for classification and the file name for the prediction file to be generated i.e. ‘prediction.txt’ (Local file generated at same location)

Compilation and Run Python code:

Go to terminal

Type python. This will start the python shell terminal

Navigate to the required path and run .py file as below->

->FilePath/ nbtest.py model.txt textcat/test prediction.txt

Expected Output:

File Output:

This file generated three output text files-

* Prediction file “prediction.txt” containing list of all files under the test directory along with the calculated positive and negative score. It also contains list of files classified as positive/negative with the exact count of files
* ‘ListOfPosNeg.txt’ containing list of 20 Terms with highest(log) ratio of Positive to Negative Weight
* ‘ListOfNegPos.txt’ containing list of 20 Terms with highest(log) ratio of Negative to Positive Weight

Console Output:

Prints the count of files in the test subdirectory.

Displays the percentage and count of files classified under class positive and negative for the given unseen list of documents.

Involvement of dev data –

Once the system is trained using training data (Model file generated) we can use the development data to check and evaluate the accuracy of the prediction as dev data also have segregated positive and negative files

Below the prediction results obtained for the dev data based on phase 1 training described above-

See prediction files for detailed file list along with positive and negative scores for all dev files

Two separate files –one for positive files and one for negative files

predict\_dev\_pos.txt -> for all files in pos subdirectory

predict\_dev\_neg.txt -> for all files in neg subdirectory

**As per given conditions in HW6**

Percentage of positive reviews in the development data classified correctly = 73% (73 documents of total 100 documents)

Percentage of negative reviews in the development data classified correctly = 85% (85 documents of total 100 documents)

For Test data: (Total files =200)

Predicted Positive files -> 86 (43%)

Predicted Negative Files -> 114 (57%)

###################ADDITIONAL################################

**Additional Measures-** taken in regards to preprocessing other than given unigrams that occur five or more time and laplace smoothing factor

NOTE: Detailed Analysis excel provided – file AnalysisReport

**UNIGRAM**

* **Individual terms obtained by splitting the tokens by punctuations other than “-”. Use of Regex along with above mentioned factors**

re.sub(ur"[^\w\d'\s-]+",' ',input)

Dev Data -

Percentage of positive reviews in the development data classified correctly = 73% (73 documents of total 100 documents)

Percentage of negative reviews in the development data classified correctly = 86% (86 documents of total 100 documents)

Test Data- (Total Files - 200)

Percentage of positive reviews in the test data = 41% (82 documents of total 200 documents) and so percentage of negative reviews in the test data = 59% (rest 118 documents)

**BIGRAMS**

* **Use of Bigrams with ignoring no terms along with regex that removes punctuation except from “-” (BEST RESULT) – One in green**

Dev Data -

Percentage of positive reviews in the development data classified correctly = 82% (82 documents of total 100 documents)

Percentage of negative reviews in the development data classified correctly = 91% (91 documents of total 100 documents)

Test Data: (For bigrams with count of tokens >= 1, ignoring no item i.e. best case)

Positive file count: 92

Negative file count: 108

Below is the detailed analysis using Bigrams (For different token length)

We have applied bigrams and observed our progress based on the count of tokens considered.

Works best for token count =1

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Data Source** | **Preprocessing and Processing Measures** | **Count of obtained positive files** | **Count of obtained negative files** | **Total Pos Files** | **Total Neg Files** | **% of obtained correct positive reviews** | **% of obtained correct negative reviews** |  |
|  | **BIGRAMS** |  |  |  |  |  |  | count of tokens |
| DEV | Use of Bigram | 82 | 91 | 100 | 100 | 82% | 91% | Ignoring no terms |
|  | Laplace Smoothing |
|  | Split on the basis of whitespaces,sequence of spaces,newlines |
|  | Split using regex to keep underscore and remove other punctuation |
|  |  | 61 | 97 | 100 | 100 | 61% | 97% | tokens>=2 |
|  |  | 46 | 99 | 100 | 100 | 46% | 99% | tokens>=3 |
|  |  | 37 | 99 | 100 | 100 | 37% | 99% | tokens>=4 |
|  |  | 28 | 100 | 100 | 100 | 28% | 100% | tokens>=5 |
|  |  | 20 | 100 | 100 | 100 | 20% | 100% | tokens>=6 |
|  |  |  |  |  |  |  |  |  |

* **TRIGRAMS (For different token length**

Fails to give a better result for this collection

Test data: (total 200, with count of tokens >=1, ignoring no terms) 🡪(**Not at all matching with any of the other measures output**, for this data set)

Positive file predicted - 44

Negative file predicted - 156

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Data Source** | **Preprocessing and Processing Measures** | **Count of obtained positive files** | **Count of obtained negative files** | **Total Pos Files** | **Total Neg Files** | **% of obtained correct positive reviews** | **% of obtained correct negative reviews** |  |
| DEV | Use of Trigram | 38 | 98 | 100 | 100 | 38% | 98% | Ignoring no terms |
|  | Laplace Smoothing |
|  | Split on the basis of whitespaces,sequence of spaces,newlines |
|  | Split using regex to keep underscore and remove other punctuation |
|  |  | 32 | 97 | 100 | 100 | 32% | 97% | tokens>=2 |
|  |  | 5 | 100 | 100 | 100 | 5% | 100% | tokens>=3 |
|  |  | 2 | 100 | 100 | 100 | 2% | 100% | tokens>=4 |

* **UNIGRAM AND DIRICHLET SMOOTHING – (Below is the analysis for diff constant values)**

Almost similar results for different values of the changing parameter

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Data Source** | **Preprocessing and Processing Measures** | **Count of obtained positive files** | **Count of obtained negative files** | **Total Pos Files** | **Total Neg Files** | **% of obtained correct positive reviews** | **% of obtained correct negative reviews** |  |
|  | **UNIGRAM** |  |  |  |  |  |  |  |
|  | **Dirichlet Smoothing** |  |  |  |  |  |  |  |
| DEV | Removed words with frequency less than 5 | 72 | 79 | 100 | 100 | 72% | 79% |  |
|  | **Dirichlet Smoothing** |  |
|  | Split on the basis of whitespaces,sequence of spaces,newlines | parameter = 0.39 |
|  |  |  |
|  |  | 73 | 78 | 100 | 100 | 73% | 78% | parameter = 0.9 |
|  |  | 74 | 77 | 100 | 100 | 74% | 77% | parameter = 1000 |
|  |  | 77 | 77 | 100 | 100 | 77% | 77% | parameter = 1900 |

**Test data**: (total 200, with parameter value as 1900)

Positive file predicted - 99

Negative file predicted - 101

Also, implemented **Jelkin Merser smoothing** for lambda =0.1 and 0.7

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | JelKin Mercer Smoothing | 76 | 79 | 100 | 100 | 76% | 79% | lambda=0.1 |
|  |  | 76 | 81 | 100 | 100 | 76% | 81% | lambda=0.7 |

Test data: (total 200, for lambda = 0.7)

Positive file predicted - 90

Negative file predicted - 110

Other python files provided:-

nbtrain-dirichlet.py and nbtest-dirichlet.py for Dirichlet Smoothing

nbtrain-ngram.py and nbtest-ngram.py for various n-grams implementation

nbtrain-jm.py and nbtest-jm.py for JM Smoothing

Output files-> (predict-jm-smoothing,predict-ngrams,predict-trigrams,predict-dirichlet-smoothing)

ADDITIONAL PRE-REQUISITE for running above files for n-grams:

For running nbtrain-ngram.py and nbtest-ngram.py,

we need additional the following libraries ::

import nltk

from nltk import word\_tokenize

from nltk.util import ngrams

excepting the last 3 all the others come built with with Python 2.7 distribution.

For downloading nltk package please follow the following steps after going to the Python

power-shell::

>>pip nltk ----->>> this will install nltk package

>> nltk.download() ----->>> this will open up nltk download dialoguebox

                   ----->>> go to "Models" tab

           ----->>> select "punkt" from the "Identifier" column

                   ----->>> Click "Download" at the bottom on the pop-up

                   ----->>> this should complete all the installations required for

                            the above mentioned two files

REFERENCES:

http://www.tutorialspoint.com/

<https://www.youtube.com/>

[www.stackoverflow.com/](http://www.stackoverflow.com/)

College notes, reference material and ebooks for smoothing concept

<http://www.ntu.edu.sg/home/gaocong/papers/wpp095-yuan.pdf>

<http://lingpipe-blog.com/2009/10/02/bayesian-naive-bayes-aka-dirichlet-multinomial-classifiers/>

<https://inst.eecs.berkeley.edu/~cs188/sp12/slides/cs188%20lecture%2020%20--%20naive%20bayes%206PP.pdf>

<http://stackoverflow.com/questions/3473612/ways-to-improve-the-accuracy-of-a-naive-bayes-classifier>

<https://www.youtube.com/watch?v=TpjPzKODuXo>

<http://streamhacker.com/tag/bigrams/>

----------------------------------------------END---------------------------------------------------------