Assignment 2:Text Classification

September 2020

1 Introduction

Stack Exchange is a very popular Q&A (Question-and-Answer) based website. We want to analyze some archived data of Stack Exchange using text classification. The link to stack exchange archive is

https://archive.org/details/stackexchange

The goal of text classification is to identify the topic for a piece of text (news article, web-blog, etc.). Text classification has obvious utility in the age of information overload, and it has become very popular for applied machine learning algorithms. In this project, you will implement k-nearest neighbor and Naive Bayes, apply these to text classification on Stack Exchange sample data, and compare the performances of these techniques.

2 Description of the Dataset

For your convenience, we have collected and accumulated a small replica of stack exchange dataset, on which you will perform our analysis. The description of the dataset is give below:

1. The size of the dataset is more 100MB. Download the zipped file containing data. The link of the dataset that we shall use is :

   https://www.dropbox.com/s/1jdct708qk8p6za/Data.zip?dl=0

2. In the training and test folder, there are respective xml files.

3. The topics.txt contains the name of the topics. For each topic, there should be a training xml file and test xml file in the respective folders.

4. For both training and test type of files, take every line which starts with "row" and keep only the "Body" portion of this row. **Consider only this portion as a document (or text) and the name of the file as the topic name.**

3 Text Preprocessing

The extracted text should be preprocessed before actual use. Following operations, at least, have to be performed as part of text preprocessing.

1. Conversion to lowercase
2. Punctuation removal
3. Stopword removal
4. Tokenization
5. Stemming
6. Lemmatization

4 k-Nearest Neighbor (k-NN)

1. Implement the k-NN algorithm for text classification. Your goal is to predict the topic for N (initially take small number of rows, e.g., 50 rows from each file and later increase the number during final submission) number of texts/rows/documents from each file in the Test folder. Try the following distance or similarity measures with their corresponding representations:

   - Hamming distance: each document is represented as a boolean vector, where each bit represents whether the corresponding word appears in the document.
   - Euclidean distance: each document is represented as a numeric vector, where each number represents how many times the corresponding word appears in the document (it could be zero).
   - Cosine similarity with TF-IDF weights (a popular metric in information retrieval): each document is represented by a numeric vector as in the case of euclidean distance. However, now each number is the TF-IDF weight for the corresponding word (as defined below). The similarity between two documents is the dot product of their corresponding vectors, divided by the product of their norms.

2. Let w be a word, d be a document, and $N(d,w)$ be the number of occurrences of w in d (i.e., the number in the vector as in the case of euclidean distance). TF stands for term frequency, and $TF(d,w) = N(d,w)/W(d)$, where $W(d)$ is the total number of words in d. IDF stands for inverted document frequency, and $IDF(d,w) = log(\frac{D+\alpha}{C(w)+\beta})$, where D is the total number of documents, $C(w)$ is the total number of documents that contains the word w, α and β are arbitrary values to avoid the IDF from zero value and division-by-zero respectively (during implementation, you can set $\alpha = \beta = 0$, and use a very small constant value when the word is

new in text document or available in all training documents). The base for the logarithm is not the determining factor, you can use e or 2. The TF-IDF weight for w in d is $TF(d,w) * IDF(d,w)$; this is the number you should put in the vector in Cosine similarity. TF-IDF is a clever heuristic to take into account of the "information content" that each word conveys, so that frequent words like "the" is discounted and document-specific ones are amplified. You can find more details about it online or in standard IR text.

3. You should try $k = 1$, $k = 3$ and $k = 5$ with each of the representations above. Notice that with a distance measure, the k-nearest neighborhoods are the ones with the smallest distance from the test point, whereas with a similarity measure, they are the ones with the highest similarity scores.

4. Output the result of running all the three values of k using all the three k-NN techniques into a single file.

5 Naive Bayes

Implement the Naive Bayes algorithm for text classification. Your goal is to predict the topic for N (initially take small number of rows, e.g., 50 rows from each file and later increase the number during final submission) number of texts/rows/documents from each file in the Test folder. Naive Bayes used to be the de facto method for text classification.

1. Consider all the words of a test text/document/question as independently, then calculate the probability of the text/document/question of being a topic and then pick up the topic which has the highest probability score.

2. Try different smoothing factors (at least 10 different values).

3. Output the result of running all the values of smoothing parameter into a single file

6 Comparison and Report Writing

In this part, you will compare between the performance of k-NN classifier and Nave Bayes classifier for text classification. Follow the steps below:

1. Among the three different measures of the K-NN, which one shows maximum accuracy? Why does it work better than other two? Write down your own justification.

2. Take the best classifier(among three) from k-NN. Compare the best k-NN with Bayesian classifier. Run 50 times both the K-NN and Bayesian learner. Compute mean and standard deviation of the results. Then,

compute t-statistic at significance levels of 0.005, 0.01 and 0.05, and compare which algorithm (k-NN or Bayesian) is better. Write the results in a report as well as the justification of the result in your own words.

3. Search in the Internet to learn t-statistics in details.

4. Never copy the report. Just answer the questions precisely. Make it as simple as possible. Too much description is not needed!

7 Special Instructions

1. You can use built-in libraries for file reading, text preprocessing and t-statistics analysis. But you CANNOT use any dedicated library for k-NN and Naive Bayes implementation. You have to implement these from scratch. However, you can use built-in utility libraries (e.g., numpy in python) during k-NN and Naive Bayes implementation.

2. Don't Copy anything! If you do copy from internet or from any other person or from any other source, you will be severely punished. More than that, we expect Fairness and honesty from you. Don't disappoint us!

3. The report should be in .docx/.pdf. Write precisely in your own language and keep it as simple as possible.

4. Your submission must have three files- two .py files for k-NN and NB implementations, and one .docx/.pdf file for report. Every file must have your student id as prefix. For example, "1505125_kNN.py","1505125_NB.py" and "1505125_Report.pdf". Note that you don't need to upload the data file.

5. The deadline of the submission is 11:55 PM, October 12, 2020 (Monday). Considering the technical problems, you should not submit the assignment at the last moment.

8 About Version

A version is included with the assignment pdf name. This is because if any information is changed, then the version will be upgraded and the changes will be summarized in this section.

8.1 Changes included in Version 1

1. This is the first version. Therefore, no changes.