Matthew Duong
z3415386
COMP9417
https://github.com/thegalah/9417_assignment.git

# COMP9417 - Naive Bayes text categorization

A version of the Naive Bayes classifier which implements the classifier algorithm in chapter 6 of Mitchell's book. The algorithm is written in python 2.

**Algorithm description/overview**

In the learning stage the algorithm gathers word frequencies for each of the document categories. It uses these frequencies divided by the total number of words in the category to generate a dictionary/hash table of word usage probabilities to be stored in memory.

The classification of the documents is simply the multiplication of probabilities using the combination of words discovered in the documents.

The algorithm essentially uses the bag of words model.

**Generalisations/assumptions**

Word order was ignored for simplicity.

**Explanation of the code**

vocab.py

The script scans the provided training set directory. Each folder in the directory is then scanned for documents which are then processed to remove all non-alphanumeric characters and all whitespace is converted to a single space. The remaining text is then split along whitespace and each piece is treated as a word and added to the vocabulary set which is then sorted alphabetically and written to the output *vocabulary.txt*.

learn.py

*Learning*

The script first begins the process of learning from the training set. Each directory name in the training set is treated as a category. All the documents under a category is then concatenated into a single larger document. A dictionary object is created (since it has constant lookup time) with the following format to contain the number of word occurrences or frequencies for each word found in the vocabulary:

```
occurences=
{
        "category 1":{
                "word1":1,
                "word2":12,
                ...
        },
        "category 2":{...},
        ...
}
```

The object is then scanned and a separate object is created in the same format as above which contains the probability that the word from the vocabulary is listed in the document set/larger document. The log of this probability is stored to deal with the issue of arithmetic underflow with the really small probabilities. The fraction can never be evaluated for this reason too. To overcome this problem simple log rules are used:
$log(a/b) = log(a) - log(b)$
Another issue (log of zero) arises since log probabilities are used. A simple quick fix is adding 1 to both the denominator and numerator:
$log(a+1) - log(b+1)$

*Classification of a document*
The classification of the document is determined by multiplying all the probabilities (or adding the log probabilities together) for all of the words in the current document being analysed. For each of the categories a log probability score is calculated using the probability dictionary object from the learning stage of the algorithm. The category with the highest log probability is selected and the document is classified under that category.

**Readme**
There are two parts to the code base:

- vocab.py - generates vocabulary from the training set.
    - Training set path must be set (relative to the vocab.py script)
- learn.py - implements the algorithm described above
    - Training set path must be set (relative to the learn.py script)
    - Validation set path must be set (relative to the learn.py script)

The scripts must be invoked using python2.
Neither of the scripts will run if there are hidden or empty directories in either the training set path

**Accuracy**
The algorithm described above achieved a success rate of 3817 correct out of the 7532 documents in the set(close to 50.7%). The probability of guessing the document category correctly is 5% (1/20).

**Performance**
Time on a single 2.5GHz thread:

```
time python learn.py
real    0m27.237s
user    0m25.880s
sys     0m1.262s
```