Topic 11: Text Categorization
Naive Bayes Learner/Classifier
COMP9417, Assignment 2

The task of text categorization naturally requires a probabilistic approach which is epitomized by Bayesian learning methods, more specifically (for text categorization) the Naive Bayes Classifier, which has been found to have performances comparable to that of neural networks and decision tree learning in some domains. The Classifier is suited to tasks where instances can be described as a conjunction of attribute values and where the target function can take on any value from some finite set[1], which is ideal for text categorization since a document (instance) can be represented by a conjunction of all words in it, and the target function would return the category to which belongs to some pre-defined list. The approach taken to solve this task was to probabilistically analyse each training document using the m-estimate of probability[2] and store it for classification of new documents, this training data would then be used to aid the learner in predicting the category for a given (unseen before) article. The performance measure of the learner would be the number of correctly classified documents out of the total number of documents given to classify.

The Naive Bayes Classifier was implemented using Java. While Java is not the best language regarding performance time, it is good for object orientation, which has been used to implement the task. Management of the training data is important to ensure accurate performance of the learner, therefore a class has specifically been created to manage the input of the training documents. This class separates the files into topics for easier reference and access throughout the program.

The training function of the Naive Bayes classifier separates a given document into a list of all the words/symbols separated by a whitespace(" "). Each word is checked for invalid symbols, e.g. an "@" symbol, and removed if not valid. Each word is also checked to determine if it is a meaningful word (by checking if it is contained in a vocabulary list and also not in a list of commonly used words such as "is" and "the") and removed if it is not meaningful for analysis. This is done on every document for a given training category. Once the list of words has been validated it is stored in an individual object representation of a category. The object then stores this list as a Hash Map, which is indexed by the actual word and the value is the frequency of the word, so while going through the list it will create new entries for unseen words or increment the frequency for seen words in the list. This is a convenient storage solution for easier referencing of the classifying function.

The testing function applies Bayes theorem[3] to predict the category for a given document. Each document for classifying is separated into a list of words, which goes through the same validation process as the words in the training files. Once this list is complete the algorithm cycles through the list of all possible topics and calculates the probabilities for each topic, while keeping track of the topic with the highest probability. The probability is calculated using the m-estimate of probability (mentioned above) for each word from the

---

[1] Tom M. Michael - Machine Learning - Chapter 6 - Page 177 - Naive Bayes Classifier

[2] Tom M. Michael - Machine Learning - Chapter 6 - Page179 - Estimating Probabilities

[3] Tom M. Michael - Machine Learning - Chapter 6 - Page177 - Naive Bayes Classifier - Figure 6.19

list of words from the current document. The parameters to the m-estimate of probability equation are provided from the training data (the Hash Map of frequencies and the overall topic probability).

The implementation of the Naive Bayes Classifier only classifies 250 out 7513 documents correctly (after multiple test runs) which is roughly the same as guessing the category for each document.