General text classifier (Project specification)

Erik Lux

13^{th} June 2011

Basic information

The main goal of this work is to implement a general algorithm classifying an input text into categories, constructed by a user. The category is created according to the examples of documents containing an appropriate material describing its content. It is in the competence of the user to provide such examples.

General description

The project is based on creating a library, applicable to the variety of different texts (e.g. including newspaper articles). The library should recognize given text and categorize it correctly due to its knowledge. The most essential part of the library includes way of information retrieval, considering learning from examples and testing its capability. It will be built upon the Nave Bayes classifier, a simple and quite accurate probabilistic classifier applying Bayes' theorem.

Project application

Moreover, the resultant project forms an application of the library, designed to classify certain texts. The application, implemented as a plug-in project, could be usable by an open-source email client. It is developed in a way to categorize incoming mail.

Project implementation

Project design

The Project is divided into two parts. The first part is a general library, where the main classification program is implemented.And the second - application part, which calls the library by certain methods. The methods include creating a new category, training the category, testing the category and categorizing a document into the category.

Data preprocessing

A text document is expected as a classification program input. Therefore, an extremely high dimensionality of text data could flow through the program. However, not all of it is useful. To reduce a given data set, the program proceedes in 5 steps. First one includes leaving out stopwords(modal verbs, conjunctions etc.). Secondly, the program takes care of stemming words(training, train, trained, etc.) and uses just the most common format. Right then, a BIF (best individual features) method makes a decision for every single word, if it will be included in a preproccesed data according to a document frequency criterion, which means the positive word frequency in previous text documents.Finally, an ordered vector of word occurencies is constructed. This vector is now available for next processing.

Data storage

A newly created vocabulary vector of a certain document could become a training material or testing material for certain class. Suppose, it is a training material. A new instance class of such vectors is built with certain probability parameters for classification purpose and it is serialized into a file. If the vector is a testing material, vocabulary vectors of tested classes are deserialized from the file. The vector(the document) is categorized into a class according to the probability information and the deserialized vectors. Right then a new class vocabulary(extended by the testing vocabulary vector) is serialized with the other unchanged class vocabulary vectors into the file.

Document classification

The multinomial Nave Bayes model is implemented. It treats a documents as an ordered vector of word occurences. The program finds a probability

that a document d belongs to to a certain class c:

$$P(c|d) = P(c) \prod_{1 \leq k \leq n_d} (P(t_k|c). \tag{1}$$

t_k represents k^{th} term in a document vocabulary.

Calculation process

The class Class with the highest probability is chosen. And the document d is categorized into the class:

$$Class = argmax_{c \in C} \bar{P}(c|d) = argmax_{c \in C} \bar{P}(c) \prod_{1 \leq k \leq n_d} (\bar{P}(t_k|c). \tag{2}$$

However, it is used $\bar{P}$ instead of P because the values are not real, they are estimated from the training data. Meanwhile, there is a lot of multiplying in the formula which can cause a problem, especially in the floating point underflow. Therefore, the formula is changed according to:

$$log(xy) = log(x) + log(y), \tag{3}$$

using the logarithmic monotony to the:

$$Class = argmax_{c \in C} [\log \bar{P}(c) + \sum_{1 \leq k \leq n_d} \log \bar{P}(t_k|c)]. \tag{4}$$

The parameters $\bar{P}(c)$ and $\bar{P}(t_k|c)$ are estimated:

$$\bar{P}(c) = \frac{N_c}{N}, \tag{5}$$

where N_c is the number of documents in the class c and N is a number of all documents.

$$\bar{P}(t|c) = \frac{O_c t + 1}{\sum_{\bar{t} \in V} O_{c\bar{t}} + 1}, \tag{6}$$

where O_{ct} is the number of occurences of term t in the training documents from class c. It is added 1 to the numerator and denominator because of the possibility that the value of the probabilies could be zero, so it could damage all the calculation.

Libraries

The program will use mainly some of the dynamically loadable libraries, that can be called in runtime, from the Java Class library. These are java.io, java.utils, java.lang and java.math.

Programming language and platform

The code will be written in Java due to its functionality on several platforms including (Unix, Windows etc). As a developer tool, the Eclipse IDE will be used. Implementation of the algorithm will be written under the Linux Platform as a java project.