General text classifier
(Project specification)

Erik Lux

13th 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 categorise 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 a library application, 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 categorise incoming mail.

Project API

The library is designed in a way to be easily extended. It consists of 6 main packages:

First one, named plainText package contains classes usable for document conversion. The package provides a conversion function, which recognises the type of the document and convert it into a plain text using one of conversion tools available. A conversion tool is an appropriate class in the package.

The second package, called dataTypes, defines new data types for a document representation as well as for a category representation in a classifier.

Third one, named featureVector, provides functions for transforming an input vocabulary vector into a feature vector using one of the BIF(Best Individual Features) methods or their combination. SFS(Sequential feature subset) methods could be implemented here as well.

The forth one, called vectorClasssifier, consists of Naive Bayes classifier class pointing out a method for classifying a feature vector into a certain category.

The fifth, with a name Classifier is the centre of the API. It provides two main methods. One for training a document into a certain class and the second for applying a classification. This package functions work with provided package methods, mentioned earlier, to accomplish certain tasks of a classification process. The methods are called individually with certain parameters, choosing a way of input data handling. So, this basically means that each package could be extended by a functional class doing quite a different job but under a condition. The class has to provide a general method to be accessed properly by the centre of the API and preserve a return type.

Last package builds a graphical interface for the project application to access library in a convenient way. It defines the frames and the visual style of the library's interface in the most general way with an additional functionality for the email client which could be modified or extended by requirements of other applications instead.

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 categorising a document into the category.

Text conversion

Documents could come in a lot of different forms such as a plain text, html documents, xml documents, email documents,... Classification functions work only with plain text documents. Therefore, the library has an internal package for text conversion. The conversion is called through the main converting function which recognises the type of a document and convert its content into plain text. There are three converters implemented in the package. The html, the xml converters, and the email converter. However, the package could be easily extended by adding another converter class. The process of the conversion returns a new data type, containing metadata and a plain text document, extracted from the content of the text.

Metadata

The conversion tool extracts not only the plain text but basic information about the document as well. This operation remains quite important as it stores the metadata of the text for the informational or any other purposes. The data are represented as dictionary or a hashmap of a key-value pair. The main key Type shows the type of the document and other keys are actually derived from the type of the document. For example, an html document contains a key *Title*. Therefore, an email document would have a key, named *From*, for instance.

Data preprocessing

A string vector 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 proceeds in 5 steps. Firstly, an ordered vector of word occurences is constructed. Second one includes leaving out stopwords(modal verbs, conjunctions etc.).

Thirdly, 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 preprocessed data according to a document frequency criterion, term frequency criterion, mutual information, χ^2 statistic or information gain. There will be available one implementation of feature subset selection and that would be forward feature selection with mutual information. This feature vector is now available for next processing.

Feature selection

This tool for dimensionality reduction is quite popular. It evaluates each vector string according to the criterion, sorts the vector and takes the given number of string values for feature vector. Its popularity comes not only from the fast and easy implementation but merely from the small computational time. It is set to be used individually or by the criterion combination. Here is the implemented criterion record.

1. Term frequency - this criterion evaluates terms in the vector according to their frequency in the training data.
2. Document frequency - terms are evaluated according to the number of documents, they are part of. It is a very simple criterion and mainly easy to implement.
3. Mutual Information - The evaluation proceeds for each term by the formula:

$$MI_{max}(t) = \max_{i=1}^{m} I(t, c_i), \tag{1}$$

   where t is a term, m represents a number of categories and $I(t, c_i)$ can be computed as follows:

$$I(t, c) = \log \frac{P(t \wedge c)}{P(t) * P(c)}, \tag{2}$$

   where P(c) is the probability of the category c in the document and P(t) is a term probability.
4. Informational gain is considered to be a statistically approved criterion. The calculation for each string goes as the formula shows:

$$IG(t) = -\sum_{i=1}^{m} P(c_i) \log P(c_i) + \sum_{i=1}^{m} P(c_i|t) \log P(c_i|t)$$

$$+\sum_{i=1}^{m} P(c_i|\neg t)\log P(c_i|\neg t), \tag{3}$$

where t represents term t, P are the probabilities.

5. χ^2 statistic evaluates vector terms by the measure of the independence between the term t and the category c. The formula right under demonstrates the calculation.

$$\chi^2_{avg}(t) = \sum_{i=1}^{m} P(c_i)\chi^2(t, c_i), \tag{4}$$

where $\chi^2(t, c_i)$ is computed as follows:

$$\chi^2(t, c) = \frac{N * (AD - CB)^2}{(A + C) * (B + D) * (A + B) * (C + D),} \tag{5}$$

where N is the number of documents in the training data. A is the number of documents, where term t and class c cooccur. B represents data, where t occurs and c does not. C is the number, where c is represented but t is not. D is the number of documents without any occurrence.

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 serialised into a file. If the vector is a testing material, vocabulary vectors of tested classes are deserialised from the file. The vector(the document) is categorised into a class according to the probability information and the deserialised vectors. Right then, a new class vocabulary(extended by the testing vocabulary vector) is serialised 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 occurrences. The program finds a probability that a document d belongs to to a certain class c:

$$P(c|d) = P(c) \prod_{1 \leq k \leq n_d} (P(t_k|c). \tag{6}$$

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 categorised into the class:

$$Class = argmax_{c \in C} \bar{P}(c|d) = argmax_{c \in C} \bar{P}(c) \prod_{1 \leq k \leq n_d} (\bar{P}(t_k|c). \tag{7}$$

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:

$$log(xy) = log(x) + log(y), \tag{8}$$

using the logarithmic monotony to the:

$$Class = argmax_{c \in C}[\log \bar{P}(c) + \sum_{1 \leq k \leq n_d} \log \bar{P}(t_k|c)]. \tag{9}$$

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

$$\bar{P}(c) = \frac{N_c}{N}, \tag{10}$$

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

$$\bar{P}(t|c) = \frac{O_c t + 1}{\sum_{\bar{t} \in V} O_{c\bar{t}} + 1}, \tag{11}$$

where O_{ct} is the number of occurrences 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 probabilities could be zero, so it could damage all the calculation.

Application interface

By the open source project, there exists an additional classification API for document categorisation processing. The API defines its own shape independently of the project it extends.

There are two main areas to deal with. A classification menu defines current categories. It enables a user to see the content of each category in a separate frame. A new category can be created here in the menu. A dialog for a document training is also available here.
On the other hand, incoming text is automatically sorted in a category as soon as the category is trained and it is marked by a class label right in the inbox folder.

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, java.awt.

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.