Data classifier CX8
(Project specification)

Erik Lux

27^{th} March 2012

Classification

Classification or document classification is a problem in computer science. The goal is to assign a document to one or more categories. This may be achieved by dividing the problem into smaller subproblems:

1. search for significant information within input documents
2. information preprocessing
3. training
4. document processing
5. decision making.

Search for significant information

Information significance depends directly on a problem definition. The task is to construct a parser to retrieve the most relevant tokens out of input.
For example, requirements of plain text classification include text documents. The parser should analyze given data and extract only textual information.

Information preprocessing

Indexing

Information retrieval usually begins with indexing of documents. Every word takes an index or a position mark. The idea is to work with an input document as a vector of terms. This is named Vector space representation.
However, there arises a problem, the representation does not solve by itself,

it is high dimensional data.

Elimination

Some parts of English vocabulary are redundant. They do not carry semantic meaning. Therefore, they are not necessarily essential. These parts are known as stopwords (conjunctions, prepositions, ...). They are simply eliminated from input documents.

Modification

On the other hand, some parts of English vocabulary carry meaning, they even have the same base word, but they take different word forms. These are called stemming word (train, trained, training, ...). During preprocessing, hey are reduced to their bases.

Feature text representation

The motivation of this section is to reduce high dimensionality of vectors. It is performed by minimizing the number of terms in the vector.Terms of the modified vector are called features.
The indisputable advantage of feature vector representation is that it could be used by both, instance-based and model-based classifiers (explained later). However, the representation does not capture all important structural information and therefore, it is not convenient enough for representation of web documents.
Nowadays, there exists one other way to represent a document that statistically outperforms the others and also satisfies the categorization of web pages. It is a recently developed graph based document representation, using the k-nearest neighbor classification algorithm.
Although it presents much better performance on web pages, the problem is that, the eager, model based classifiers, cannot use the representation directly. It could not be used because these classifiers work only vector representation and do not accept any other. Therefore, new hybrid representation, combining the advantages of these two, start to appear ...
There is a wide variety of approaches dealing with the process of feature selection. Some of them are discussed in the next paragraphs.

Best Individual Features (BIF)

This technique uses an evaluation function that is applied to a single term. Scoring individual terms can be performed by some measures like term frequency, document frequency, mutual information, ...

Subset Feature Selection (SFS)

Contrary to BIF, this technique firstly selects the top scoring term and then adds one term at a time to fulfill the required number of terms in a feature vector.

Feature Transformation (FT)

The approach does not measure the weights of terms but compacts the vocabulary based on feature occurrences. The idea is to learn a discriminative transformation matrix in order to reduce initial vector space.

Training

From now on, the section *Classification* understands under the term document a preprocessed feature vector.
Before any of classifiers could be applied to document recognition, they need to gain certain knowledge of the problem.
Knowledge is provided by the documents, called training data. Classifiers store this knowledge into their data structures. The saved information becomes known under the term vocabulary.

Training versus Testing

Classifiers usually do not store all training documents. Documents are divided into two partitions instead. The first partition is provided to be stored but the second partition is intended to become testing data. Division ration depends on user preferences (typically around 50

Cross-validation

The technique is sometimes known as rotation estimation. It assesses how the results of a statistical analysis on training data generalizes to independent

testing data. It is used to estimate how accurately a classifier, working with the vocabulary, performs in practice. This is achieved by selecting different subsets of training documents for training data and testing data in multiple rounds.

Document processing

This section presents some of the approaches, except the Naive Bayes' classifier, that deal with pure categorization. It does not describe the way they work or evaluate input data.

Classification models

Classification model is a scheme for a vector of terms. It specifies its inner representation. Two models are described right below:

1. Multivariate model is an older model. It handles a vector as a binary vector that represents whether a current term is present or absent in vocabulary. Recently, it is not used very often because it lacks the ability to utilize term frequencies.
2. Thus a multinomial model is introduced as an alternative model. It stores the number of occurrences for each term in a vector. Two serious problems are encountered while working with the model:
   (a) Rough parameter estimation - training data contain a variety of documents. Some of them could be much longer than the others. Although their length is rarely related to their importance, these documents influence the calculation more than the others
   (b) Insufficient number of training data - this is a general problem. Some of categories are poorly equipped with data. They need more training.

Classifiers

Approaches for document classification can be divided into two groups. One could be named model-based classifiers and the other instance-based classifiers.

Model-based classifiers are built upon a mathematical model. Therefore, their calculation steps use the basic principle of the current model. They generalize the problem and apply the principles.These include classifiers such

as Naive Bayes, expectation maximisation, latent semantic indexing, artificial neural network, decision trees ...

On the other hand, instance-based learning or memory-based learning is a family of learning algorithms that, instead of performing explicit generalisation, compare new problem instances with instances seen in training data, which have been saved directly into the memory. Instance-based learning is a kind of lazy learning. It is known under the name of instance-based because it makes hypotheses directly from the training set of instances. As an example of instance-based classifiers is considered k-Nearest neighbor algorithm.

Decision making

The result of document processing is an assignment of some calculated values or probabilities to each category. Then, the category with the highest probability is chosen.

Similar projects

There are some well-known projects, using the Naive Bayes' classifier. They are fully discussed with all their features over here:

1. **Data mining software in Java named Weka, using Naive Bayes Classifier**

   Weka is a collection of machine learning algorithms for data mining tasks. The algorithms can either be applied directly to a dataset or called from your own Java code. Weka contains tools for data preprocessing, classification, regression, clustering, association rules, and visualization. It is also well-suited for developing new machine learning schemes. Moreover, Weka provides access to SQL databases with java connectivity. It is not directly matched with the input tables for categorization process but can be simply transformed into these tables. The area which is still not included in weka algorithms is sequence modeling.
   This cross-platform software provides user except Naive Bayes and other basic algorithms even the implementation of Expectation Maximalisation algorithm or K-means algorithm. It could be easily manipalated by a graphical interface. Simply, this software is a present of

text categorization.

Weka homepage

2. **Classifier4J**

   Classifier4J is a Java library designed to do text classification. It comes with an implementation of a Bayesian classifier, and now has some other features, including a text summary facility. It makes the use of vector data reprezentation. Its API could work as a spam filter or blog cl.

   Classifier4J homepage

3. **A Naive Bayesian Classifier in C# - NClassifier**

   NClassifier is an open source product. It is a .NET library that supports text classification and text summarization. It is a very extensible library consisting largely of interfaces. It includes, out of the box, an implementation of the Bayesian classification algorithm. The classifier is synched closely with Classifier4J project. For example, the database handling in java is replaced with ADO.Net solution. The future perspective of the project is undetermined. It could at least stay with Classifier4J or, at the most, cut into its own direction.

   NClassifier homepage

4. **The RDP Classifier - a nave Bayesian classifier**

   The Ribosomal Database Project (RDP) Classifier, a nave Bayesian classifier, can rapidly and accurately classify bacterial 16S rRNA sequences into the new higher-order taxonomy proposed in Bergey's Taxonomic Outline of the Prokaryotes (2nd ed., release 5.0, Springer-Verlag, New York, NY, 2004). It provides taxonomic assignments from domain to genus, with confidence estimates for each assignment.

   The RDP classifier homepage

5. **Mallet**

   Mallet is a java-based software for natural language text processing. It provides efficient routines for feature vector conversion. Mallet in-

cludes a wide variety of classification tools(Naive Bayes and decision trees techniques for instance). Furthermore, Mallet has got code for evaluating its classification algorithms and its efficiency.

Mallet homepage

6. **jBNC**

   It is a java toolkit, providing methods for training, testing and applying Bayesian Network Classifiers. This software was manly tested in artifiacial inteligence and machine learning tasks.

   jBNC homepage

7. **Orange**

   Orange is a component-based data mining and machine learning software suite, featuring friendly yet powerful and flexible visual programming front-end for explorative data analysis and visualization, and Python bindings and libraries for scripting. It includes comprehensive set of components for data preprocessing, feature scoring and filtering, modeling, model evaluation, and exploration techniques. It is implemented in C++ (speed) and Python (flexibility). Its graphical user interface builds upon cross-platform Qt framework. Orange is distributed free under the GPL. Orange includes a component based Naive Bayes Classifier.

   Orange Component Naive Bayes Classifier

Basic information

Data classifier CX8 is a java toolkit for classification of text documents of different file format. It implements several feature selection methods to be applied with the Naive Bayes classification algorithm.
CX8 stands for Classifier X of eight selection methods. CX8 classifies documents into user-defined categories. Category is a set of documents, containing some metadata information. Document examples, falling within the same category, share a common theme. Consequently, it is user competence to provide such examples.

General description

CX8 is based on two component model. The main project component is a library, applicable to a variety of different texts (e.g. newspaper articles). The library should recognize a given text and categorize it correctly due to its knowledge. The process includes information retrieval from texts, learning from collected samples and testing the knowledge gained in learning. The process results in user evaluation of library performance.
The component is built upon the Naive Bayes classifier, a simple and quite accurate probabilistic classifier applying Bayes' theorem [5].

Project application

The other component of the project is a library application. The application uses library services. It provides an extra graphical interface. This interface enables users to work with the library comfortably. The application could be used by any kind of existing software as its extension. In this case, an email client, called Mailpuccino, is considered to be the one. The client is extended in a way to categorize incoming mail.

Technical documentation

Project design

As mentioned above, CX8 is based on two component model. The application component needs to call the library component to perform classification tasks. These two components are dependent on each other. The communication between them is ensured by the public library interface. It provides the following methods: to create a new category, to train the category and to classify the input document into the category.

Text conversion

Documents could come in a lot of different forms such as plain text, html, xml, raw email,... Classification methods work only with plain text documents. Therefore, the library contains its own package for text conversion. The conversion tool recognizes the document type and converts its content into plain text. There are three converters implemented yet: the html converter, the xml converter and the raw mail converter.

The html converter is an extension of HTMLEditorKit class. HTMLEditorKit is a built-in Swing parser to parse HTML and extract the links.
The xml converter is a simple tool. It just removes six predefined xml entities: ",&,',<, >,/.
The raw mail converter is written in a way to extract certain parts of email. The parts are meta information (e.g. subject, sender, receiver, data,...) and email body. The tool parses information according to the standard email syntax.

Metadata

Conversion tools not only return a plain text but metadata of the document as well. This extraction is a necessity as document information is stored for technical and statistical purposes. Metadata are represented as a dictionary or a hashmap of key-value pairs. The main key *Type* represents the document type. Other keys depend on the actual document type. For example, a html document contains the key *Title*. An email document contains the key *From* and the key *To* for instance.

Data preprocessing

A string vector is expected as the input parameter of the classifier. The length of the vector is not known in advance. Extremely high dimensional data could flow through the program. However, not all dimensions are useful. To reduce the given data set, the program proceeds in 4 steps: .

1. create an ordered vector of word occurences
2. leave out stopwords(modal words, conjunctions, ets.)
3. stemm words in the vector(training -> train, train -> train, trained -> train)
4. find top features [1][3][7].

Feature selection

It is quite common to employ a method for dimensional reduction. The method is applied not only because of its easy implementation but merely because of its small computational time.
Such method evaluates the terms of each vector according to some criterion (criteria are applied individually or in groups). The terms are then sorted according to their weights. Finally, the given number of the best terms are selected to construct a feature vector.

The description of implemented criteria is present over here: [3].

1. Term frequency (TF) - the criterion evaluates terms in the vector according to their frequency in training data [1].

2. Document frequency (DF) - terms are evaluated according to the number of documents, they are part of [1].

3. Mutual Information (MI)- The criterion measures how much information the presence or the absence of term t contributes to making the correct classification decision on category c. Formally:

$$I(T,C) = \sum_{e_t \in\{1,0\}} \sum_{e_c \in \{0,1\}} P(T=e_t, C=e_c) \log \frac{P(T=e_t, C=e_c)}{P(T=e_t)P(C=e_c)}, \tag{1}$$

   where T represents a variable that takes value e_t=1 (the document contains term t) and e_t=0 (the document does not contain t). C is a variable that takes e_c=1 (the document is in category c) and e_c=0 (the document is not in category c). The equation is equivalent to the next:

$$\begin{aligned} I(t,c) = & \frac{N_{11}}{N} \log \frac{NN_{11}}{N_{1.}N_{.1}} + \frac{N_{01}}{N} \log \frac{NN_{01}}{N_{0.}N_{.1}} \\ & + \frac{N_{10}}{N} \log \frac{NN_{10}}{N_{1.}N_{.0}} + \frac{N_{00}}{N} \log \frac{NN_{00}}{N_{0.}N_{.0}}, \end{aligned} \tag{2}$$

   where $N_{subscript}$ are the counts of documents that have the value of e_t and e_c that are indicated by the two subscripts. For example N_{01} is the number of documents that contain term t and is independent of category c. $N = N_{00} + N_{01} + N_{10} + N_{11}$ is the total number of documents. An example of transformation from equation (1) to equation (2) is $P(T = 1, C = 0) = N_{10}/N$ [1][5].

4. Information gain (IG) is frequently employed as a criterion of term quality. Let T be the set of all terms and D be the set of all training documents, value v(x,a) defines the value of specific document d for term $t \in T$. H specifies the entropy. The information gain for document $d \in D$ is defined as follows (note the logarithm is still base 2 in the following samples):

$$IG(D,t) = H(D) - \sum_{w \in v(t)} \frac{|\{d \in D | v(d,t) = w\}|}{|D|}$$

$$H(\{d \in D | v(d,t) = w\}). \tag{3}$$

The entropy of the whole set of documents D is evaluated over here:

$$H(D) = \sum_{i=1}^{c} -p_i \log p_i, \tag{4}$$

where p_i is the proportion of D belonging to class c.

To make the value of Information gain for each term more accurate, IG(D,t) is divided by Intrinsic value of term t. It is actually the entropy of D with respect to the values of term t. Intrinsic value for term t is estimated:

$$IV(D,t) = - \sum_{w \in v(t)} \frac{|\{d \in D | v(d,t)\}|}{|D|}$$

$$\log \frac{|\{d \in D | v(d,t) = w\}|}{|D|}. \tag{5}$$

The ratio between Information gain and Intrinsic value is called Information Gain Ratio. It penalizes terms by incorporating a term, called Intrinsic value, that is sensitive to how broadly and uniformly the attribute splits the data:

$$IGR(D,t) = \frac{IG(D,t)}{IV(D,t)} [9][10]. \tag{6}$$

5. χ^2 statistics (X2) evaluates vector terms by the measure of the independence between term t and category c. The formula right under demonstrates the calculation:

$$\chi^2_{avg}(t) = \sum_{i=1}^{m} P(c_i) \chi^2(t, c_i), \tag{7}$$

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

$$\chi^2(t, c_i) = \frac{N * (AD - CB)^2}{(A + C) * (B + D) * (A + B) * (C + D),} \tag{8}$$

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 of documents, where c is represented but t is not. D is the number of documents without any occurrence of c, t [1][5][10].

6. Sequential feature selection (SFSMI) works with the pre-calculated value of MI. Let V be the initial input vector and w the feature term of MI features. After the MI-coefficients are calculated, the method finds the w' maximizing MI(w). Let U be $V \backslash \{w'\}$ and S be {w'}. The next selection step is a loop that runs until it reaches the desired number of features:
   (a) For w $\in U$: compute MI(w,S).
   (b) Find the word w" that maximizes MI(w,S), set $U = V \backslash$ {w"} and $S \cup \{w''\}$.

   The calculation of MI(w,S) or finding the direct relation between w and the whole set of words is quite difficult. Therefore, several approaches were presented so far to approximate the value of MI. One of the best solutions is suggested by maxMI algorithm. It substitutes the problematic assignment with the formula:

$$maxMI(w, S) = MI(w) - \max_{s \in S} I(w, S), \tag{9}$$

   which provides a handy way to deal with the calculation of MI for a term and a term set [1][8].

Expected time complexity

Trained vocabulary is a set of all training data. Classified vector is a vector of terms being classified. Let n be the length of trained vocabulary and l be the length of classified vector.
The stemming and stopword process takes O(l) time multiplied by a constant that represents the length of calculation process for one term. This means O(l) asymptotically.

Let us have a closer look at feature selection methods. Each method searches the whole system of trained vocabulary. It iterates over each term in every single document of categories of trained vocabulary. This means time of $|categories| * |documents| * |terms|$, approximately equal to O(n). Then, it iterates over classified vector, which takes O(l). All together it takes O(ln). Assuming that classified vector is probably much shorter than trained vocabulary, this takes linear time O(n). Let vocabulary search be the name of all these iterations.

1. Document frequency - Vocabulary search is performed once. This means linear time O(n).

2. Term frequency - Vocabulary search takes O(n). This takes O(n) as well.

3. Mutual information - To provide number of documents that contain term t and are in category c, trained vocabulary is searched during vocabulary search. The information if term t is present or absent is searched for in every iteration of vocabulary search. The same search is applied for the information if document d belongs with category c or does not. This takes O(l)$O(n^2)$. Assuming that l $\ll$ n, it is quadratic time $O(n^2)$.

4. Informational gain - Trained vocabulary is searched in every iteration of vocabulary search. This means quadratic time $O(n^2)$.

5. χ^2 statistic - The method applies the same principle of vocabulary search in vocabulary search as Information gain. This takes quadratic time $O(n^2)$.

6. Let k be the final number of features in a feature vector. It is guaranteed that the calculation complexity would be at least quadratic because of mutual information. Vocabulary search in vocabulary search is performed for each of k features. It could be quadratic in worst case scenario. This takes O(k)$O(n^2)$ time. Assuming that k $\ll$n, it is quadratic time $O(n^2)$ asymptotically [9].

Data storage

A newly created vocabulary vector could become a training material or a testing material. Suppose, it is the training material. The feature vector is either added to the existing category or a new category instance is initialized. The vector then becomes a part of the instance. In the end, the category is serialised into a file.
If the vector is the testing material, vocabulary feature vectors are deserialised from the file. The vector is classified into the category according to the probability, specified by deserialised items. It is added to one of the categories, which could be sebsequently serialised back into the file, if demanded.

Document classification

CX8 implements the multinomial Nave Bayes model [1] which treats a document as an ordered vector of word occurrences. It is based on the following algorithm:

1. compute a conditional probability that category c is a searched category given the probability that document d belongs to c.
2. determine the most probable category applying the Naive Bayes formula.

Conditional probabilities are estimated using:

$$P(c|d) = P(c) \prod_{1 \leq k \leq n_d} P(t_k|c). \tag{10}$$

P(c) is the probability of category c. t_k represents the k^{th} term in document d. P(t_k|c) is a conditional probability describing t_k as the part of document d given the probability that category c is a part of d as well. [5][6].

Calculation process

The Naive Bayes formula is applied to the newly calculated conditional probabilities:

$$Category = argmax_{c \in C} \bar{P}(c|d) = argmax_{c \in C} \bar{P}(c) \prod_{1 \leq k \leq n_d} \bar{P}(t_k|c). \tag{11}$$

The formula contains the probability of $\bar{P}$(c|d) instead of P(c|d) and $\bar{P}$(c) instead of P(c). To emphasize the fact that the probabilities are not real but estimated from training data.

The formula contains a high number of multiplications which can cause a problem, especially in the floating point underflow. Therefore, the logarithm function is applied to the formula using a rule:

$$log(xy) = log(x) + log(y). \tag{12}$$

Thanks to the logarithmic monotony, the formula can be rewritten as follows:

$$Category = argmax_{c \in C}[\log \bar{P}(c) + \sum_{1 \leq k \leq n_d} \log \bar{P}(t_k|c)]. \tag{13}$$

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

$$\bar{P}(c) = \frac{N_c}{N}, \tag{14}$$

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

$$\bar{P}(t|c) = \frac{O_c t + 1}{\sum_{\bar{t} \in V} O_{c\bar{t}} + 1}, \tag{15}$$

O_{ct} is the number of occurrences of term t in training documents belonging to class c. However, the probabilities could equal zero in the denominator $\sum_{\bar{t} \in V} O_{c\bar{t}}$, leading to an undefined expressions. Therefore, 1 is added to both the numerator and the denominator [5][6].

Project API

General

The classification library is implemented as a set of java packages working together. The packages are all exported as a jar file. To use the library, the built path of the project has to be set to the external jar file and import the main package *NaiveBayes*. The jar provides public methods to gain access to the library, described sequentially:

The method *train* is designed to expand classifier's knowledge. It accepts sample text data. There are implemented four variants of the method, differing in the input parameters. In general, the method accepts three parameters: the category name, the document type and the input text. First

two parameters apply to each of the variants. The document type flags the input format in order to preprocess the text. The third parameter specifies the form of the input. Method *trainFile* accepts the file name of the existing input. Method *trainString* accepts the input as the string. To provide multiple training data just by one method call, two overloaded variants of previous methods are available, accepting the third parameter as the vector of file names or the vector of strings respectively.

The method *classify* is constructed to choose the most suitable category for the input text. There are implemented two variants of the method: *classifyFile* and *classifyString*. In general, the methods accepts three parameters. The first one specifies if the input text is a string value or a string file name. The second parameter, the document type, has the same function as the parameter document type of the method *train*. The third one, the selection type, allows to choose one or more feature selection methods to select top features.

The trained material is stored into a file. The file's name is *catfile*. By default, there is only one file. The property methods *setcategoriesFileName*, *addcategoriesFileName* and *removecategoriesFileName* provide an interface to control the serialization mechanism, to store training data into more than one file and to enable even more training tasks, where the task could be described as a training system of files related to a certain topic.

The last function of the library available is *setnumFeat*. It changes the default number of selected features, accepting a new value as the integer parameter.

Main package

This package provides a public interface using the functionality of all packages. It is designed in a way to be easily extended. Therefore, each method represents a logical part in the document processing. This basically means that each package could be extended or changed independently of others but under two conditions. The package has to provide a function visible enough for the main package and this function has to perserve the required return type. Moreover, the library could be extended by a new logical block if required.

Data types package

The set of categories is internaly represented as a vector of category instances. Category is a new data type. Each category has member items such as a name, a length and a vector of documents. Additionally, a few setters and getters to add, rename or change the category content are included in Category.
In fact, Document is implemented as another data type. It contains a property vector, applicable as a metadata storage. A piece of metadata information is one property. It also contains a vocabulary vector, which is internaly represented as a hashmap of string words with the number of their occurences.

Plain text package

Document conversion is provided by Plain text. Each class represents a specific type of conversion, recognizing the document type and changing it to plain text . The html, email and xml conversion tools are available:

1. Html conversion tool removes all the html tags and its parameters. It parses plain text included between two of the tags. Unknown tags are not removed. The parser simply ignores wrong written html code(missing brackets, tags, etc.). This could lead to more serious problem as the parts of the tags remain untouched and harm the training or the classification process. The main reason not to implement a control mechanism over wrong code is a variety of mistakes, it could contain. Trying to handle unpredictable mistakes could be less efficient than leaving them out.
2. Xml parser just removes specific symbols of xml tagging(/, <, >, etc.) because it takes all tag names and attributes as a part of the document. However, some wrong written xml code remains untouched.
3. Email converter just parses the body section of the mail if not given explicitely and passes it to the next process.

The package is freely extensible by other conversion tools or any conversion methods. Moreover, a metadata parser is implemented as a part of the package, allowing to extract meta information from html or email documents if required. It searches for appropriate meta tags in documents. If they are found, the parser extracts information from the tags and creates a new metadata property.

Vector conversion package

The removal of redundant words and their stemming can be performed by the package. Two classes are available, stopwords and stemming words.

The input text is parsed according to the default stopword file deleting all the stopwords. The stopword file can be easily changed or updated by an overloaded variant of the method, removing the old stopwords from the package. The update method adds a new stopwords file, accepting its file name as the input parameter.

Stemming a word means reducing a derived word into its stem or base, which can differ a lot in many situation. A freely distributed Porter stemming algorithm, slightly adjusted, deals with the stemming in the package [10].

Serialize package

The current instance of the category set is serialised into the file specified by the private variable *catfile*. It is performed by the package using serialising and deserialising methods: *serjavaObject* and *deserjavaObject*.
The serialisation is built in a way to be used as little as possible because it takes a certain amount of time. By default, any changes are not serialised implicitly. Nevertheless, they might be if demanded.

Feature vector package

Construction of a feature vector tends to be the most peculiar part of all. Therefore, it is necessary to design this package with a certain amount of abstraction.

The implementation of these BIF(Best Individual Feature) methods is provided: term frequency, document frequency, mutual information, information gain and χ^2 statistics. In addition, the SFS(Subset Feature Selection) method, using the precalculated value of mutual information, is implemented as well. In order to ensure portability and easy extensibility, the best solution is to write each selection method into a different class of the package. Such class should implement some kind of a general scheme which would dictate types of parameters and a return type for each selection method. The scheme would also contain a structure to capture results of calculation from selection methods. All requirements on the scheme could be ensured by inplementation of an abstract class.

Therefore, class *selector* is constructed. It contains a member field, that represents a final feature vector, a virtual function (selecticting top features) and a hashmap (having one key feature and one value feature that represents an occurence counter of the key feature). It also provides a class that is used for comparison of two key features by their value. The class implements the Comparator interface.

In addition, the public interface of the package provides another class that specifies the type of feature method selection. Member methods of the class accept the selection type as the string parameter. The string is built from integer numbers, which define the type of method selection. Classes of selection methods are instanced according to the parameter. If only one method is selected, the instance calls its member function to find top features of a classified vector. A vector of these features is returned as a result.

Otherwise, a number of different methods could be applied. First of all, member methods calculate their feature vectors of user-defined length. Secondly, the mentioned methods merge these features into a single feature set of the same length . The difference is in a way, the methods merge these features.

Features are sorted according to the number of their occurence in the vector. Methods iteratete over the 1^{st} sorted feature, then 2^{nd} sorted feature and ..., until the whole vector is searched. There are two implemented approaches:
If the method access the feature, which does not belong to the final feature vector yet, it is immediately added to the vector either with the value of one occurence or the value of the actuall number of occurences in the input.
The first approach works with an iterated item as with a single feature while the second approach tries to strengthen the position of the topmost features. The second approach supports two variants of unsuccessful iteration (the feature already belongs to the final vector), either by moving the iterator to the next vector at the same position or by moving the iterator to the next feature of the vector.

The paragraph deals with the selection method classes. Their names corre-

spond to the appropriate selection methods. They contain derived variables and methods from the class *selector*. Furthermore, they implemet several string-double hashmaps, which store certain information about input data and support the final selection of top features. The key of the hashmap is a concatenated string value of the category name, the mark $ (a separator) and the appropriate vocabulary term. The current probabilities or any other meta values, useful for the following calculation, are stored in the hashmaps as double numbers.
The following calculation of top features is based on details, mentioned in the section *Feature Selection*. Selected features are then put into another map. Afterwards, the map is sorted by the implemented comparator and the topmost features are returned as a final feature vector.

Vector classifier

The namespace provides two methods, using the current categories instance and the vocabulary feature vector as their parameters to calculate conditional probabilities and to find the right category. The *Calculation process* section describes the implementation of the algorithm. The return type is a string value corresponding to the existing category name.

Mailpuccino

Mailpuccino is a free, fully featured email client written as a java application. It supports POP3 and SMTP, has an address book and features to view, save and send attachments. Its source code is distributed under the terms of the GNU General Public License.

Extension

The application of the project extends the functionality of Mailpuccino. It provides a graphical interface to classify mail into user-defined categories.
Physically, it adds a right-side panel that displays an option menu and a list of categories.The menu provides options to add or remove categories. It provides settings option to set classification parameters: the type of input (the file or the string), the selection method, the form of classification (train or classify) and the number of features.
The menu provides one more option. A new document could be added. The classification process of the document depends on the selected parameters in the menu. The process adds the document into the training data or classify it into one of the categories.

Moreover, the extension marks all mail. Marking could be performed automatically if demanded. An email document is marked with "U" (unclassified) if the document has not undergone the classification process yet. The document is marked "T+Category" if the document belongs to the training data of Category. The document is marked with the label "C+ Category" if it was classified into Category. Marks are visible among the other mail identifiers in mail folders such as Inbox or Trash.

Exceptions

Generally speaking, there are two exception types possible. Internal exceptions, occuring right in the packages, do not affect the program a lot. They mean that the calculation cannot be procceded exactly in the way the user expects it to. If the new stopword file cannot be found, the program uses the default one. There is no need in propagating the exception up the methods' hierarchy.

On the other hand, if the exception handles the serialization process, it could seriously affect the calculation. The approach propagates the exception with the well defined description which quickly terminates the program right after.

Libraries

CX8 uses mainly some of the dynamically loadable libraries called in runtime from the Java Class library. These are java.io, java.utils, java.lang and java.math, java.awt, java.swing, java.swing.text, java.swing.text.html, java.swing.text.html.parser.

Programming language and platform

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

References

[1] S. Kotsiantis, E. Athanasopoulou, P. Pintelas, Logitboost of Multinomial Bayesian Classifier for Text Classification, International Review on Computers and Software (IRECOS), Volume: 1(3), Pages: 243 – 250, November 2006.

[2] Dmitry Pavlov, Ramnath Balasubramanyan, Byron Dom, Shyam Kapur, Jignashu Parikh, KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data miningsification and Clustering with Mixture of Multinomials, Pages: 829 – 834, ACM, ISBN:1-58113-888-1, 2004.

[3] Sang-Bum Kim, Kyoung-Soo Han, Hae-Chang Rim, and Sung Hyon Myaeng, Some Effective Techniques for Naive Bayes Text Classification, Ieee Tranasactions on knowledge and data engineering, Volume: 18 Issue: 11, pp. 9185444, ISSN: 1041-4347, November 2006.

[4] Michael W. Berry and Malu Castellanos, Survey of Text Mining: Clustering, Classification, and Retrieval, Second Edition, Editors. Springer, January 2008.

[5] Christopher D. Manning, Prabhakar Raghavan and Hinrich Schtze, Introduction to Information Retrieval, Cambridge University Press, ISBN: 0521865719, 2008.

[6] Ethem Alpaydn, Introduction to Machine Learning Second Edition, Massachusetts Institute of Technology, Cambridge, Massachusetts, London, England, The MIT Press, ISBN-10: 0-262-01243-X, 2010.

[7] Peter Jackson and Isabelle Moulinier, Natural language processing for online applications Text retrieval, Extracton and Categorization, John Benjamins Publishing Company, ISBN-10: 1588112500, July 2002.

[8] Jana Novovicov, Anton Maly, and Pavel Pudil, Feature Selection using Improved Mutual Information for Text Classification, Institute of Information Theory and Automation, Department of Pattern Recognition, Academy of Sciences of the Czech Republic; The University of Economics, Faculty of Management; Czech Technical University, Faculty of Electrical Engineering; Prague, Czech Republic; Springer, 2004.

[9] Yiming Yang and Jan O. Pedersen, A Comparative Study on Feature Selection in Text Categorization, School of Computer Science, Volume: 20, Issue: 15, Publisher: Citeseer, Pages: 412 – 420, ISBN:1-55860-486-3, 1997.

[10] Willett, P., The Porter stemming algorithm: then and now. Program: electronic library and information systems, Volume: 40 (3), Pages: 219 – 223, 2006.

[11] Tom M. Mitchell, Machine Learninng, The McGraw-Hill Companies,Inc.,

ISBN: 0-07-042807-7, 1997.