Java Library for Support of Text Mining and Retrieval

Peter Bednar, Peter Butka, Jan Paralic

Dept. of Cybernetics and Artificial Intelligence, Technical University of Kosice,
Letna 9, 042 00 Kosice, Slovakia
{Peter.Bednar, Peter.Butka, Jan.Paralic}@tuke.sk

Abstract. This paper describes an original software system developed in Java for support of information retrieval and text mining. The system is being developed as open source with the intention to provide an easy extensible, modular framework for pre-processing, indexing and further exploration of large text collections. In paper, overall architecture of the system is provided, followed by three typical use case scenarios showing some possible applications of our system.

1 Introduction

Our research and education goals in the area of text mining and information retrieval with the emphasis of advanced knowledge technologies for the semantic web resulted in identification of the following requirements, which ideal software system for our purposes should poses.

1. Be able to efficiently preprocess potentially large collections of text documents with flexible set of available preprocessing techniques.
2. Particular preprocessing techniques should be well adopted for various types and formats of text (e.g. plain text, HTML or XML).
3. Text collections in different languages were envisaged, e.g. English and Slovak, as very different sorts of languages require significantly different approaches in preprocessing phase.
4. Support for indexing and retrieval in these text collections (and experiments with various extended retrieval techniques).
5. Well-designed interface to knowledge structures such as ontologies, controlled vocabularies or WordNet.

The decision to design and implement a new tool, Java library for support of text mining and retrieval, was based on the detailed analysis of existing free software tools that could be used to support the abovementioned functionality requirements.

We found four different groups of tools:

- Text indexing and retrieval tools (such as e.g. Lucene [1]),
- Tools for text processing (e.g. GATE [2], JavaNLP [3]),
- Tools and APIs for support of the process of knowledge discovery in databases (Weka [4], KDD Package [5], JDM API [6]),
- Frameworks for work with ontologies (e.g. KAON [7]).

Features of all groups are summarized in the following table.

Table 1. Some existing tools related to JBOWL.

Tools	Features					
	text analysis	NLP methods	vector representation	mining models	interface to ontologies	full-text search
Lucene	yes	no	no	no	no	yes
KDD Package	no	no	not optimized	yes	no	no
WEKA	no	no	not optimized	yes	no	no
JDM API	no	no	not optimized	yes	no	no
GATE	yes	yes	base support	text extraction task only	yes (text annotation)	no
JNLP	yes	yes	base support	no	no	no
KAON	yes	no	base support	no	yes	no

Each of the group covers very well one or two from the requirements stated above, but none of the tools is suitable for support of the other requirements and therefore are not very well suited for text mining and semantic retrieval.

Proposed Java library for support of text mining and retrieval provides an easy extensible and easy to learn modular framework for preprocessing and indexing of large text collections, as well as for creation and evaluation of supervised and unsupervised text-mining models.

The rest of the paper is organized as follows. Next, section 2 presents the overall architecture of the designed system, briefly describing supported text-mining tasks. Sections 3, 4 and 5 provide three different scenarios, how the system has been exploited for different purposes. Finally, Section 6 provides a brief summary of the paper as well as some sketches of our future work.

2 Architecture

JBOWL has the same architecture like standard Java Data Mining API (JSR 73 specification [6]). This architecture has three base components that may be implemented as one executable or in a distributed environment.

- **application programming interface (API)** - The API is set of user-visible classes and interfaces that allows access to services provided by the *text min-*

ing engine (TME). An application developer using JBOWL requires knowledge only of the API library, not of the other supporting components.

- **text mining engine (TME)** - A TME provides the infrastructure that offers a set of text mining services to its API clients. TME can be implemented as a local library or as a server of a client-server architecture.
- **mining object repository (MOR)** - The TME uses a mining object repository which serves to persist text mining objects.

Figure 1 depicts possible architectures for a JBOWL implementation. In (a), each component resides in a separate physical location or separate executable. This implementation has three-tier architecture with the data stored in a separate repository (i.e. in relational/object database). In (b), the system is monolithic: API, TME and MOR reside in, or are managed by a single executable. Current JBOWL implementation follows architecture (b).

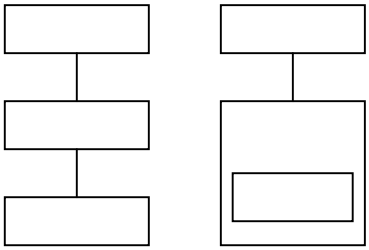

Figure 1. JBOWL architecture (a) - three tired, (b) - monolithic

Supported Mining Tasks

TME manages execution of common text mining tasks, e.g. document analyzing, building a model, testing a model, applying a model on new data, computing statistics, and importing and exporting existing mining objects from and to MOR. Reference JBOWL implementation supports following tasks:

Document analyzing

Text analyzing on some language level is required by all following text-mining tasks. For analyzing task JBOWL provides tokenizer classes, which divide text into tokens and various token filters, including filters for token normalization, stemming, collocations and stop words filtering. Some other token filters can provide more sophisticated processing like POS tagging or word sense disambiguation.

Building a model

JBOWL currently enables users to build classification model, clustering model and attribute selection model. To build models, users define settings that describe the type

of the model, selected algorithm, and input data (i.e. training data set and validation data set used for evaluation of model in learning phase). After a model is build by the TME, it can be persisted in the MOR.

Settings provided by the users can be divided into two levels: general settings related to model *function* (i.e. classification, clustering or attribute selection) and settings related to selected *algorithm* (i.e. Naive Bayes classifier, SOM neural networks, or information gain attribute selection).

Testing a model

Model testing provides estimation of accuracy for predictive (classification) models. Test task accepts a model and data for testing. As a result of model testing, JBOWL can produce accuracy measures such as micro and macro averaged recall and precision.

Applying a model

Applying a model to the new document produces one or more predictions or assignments. For supervised text mining, model applying produces predictions of category assignments possibly along with their corresponding probabilities. In unsupervised text mining (i.e. clustering) apply assigns a document to a cluster. JBOWL enables batch scoring as well as single document scoring, intended for real-time response.

Computing statistics on text data

This task computes term and category occurrence and co-occurrence statistics required for building of various text-mining models.

3 Text Categorisation

JBOWL provides set of common java classes and interfaces that enable integration of various classification methods. The design of JBOWL distinguishes classification *algorithms* (i.e. SVM, linear perceptron, etc.) and *classification models* (i.e. linear classifier, rule based classifier).

The same type of the model (the same classifier) can be build with many algorithms or with many different algorithm implementations. In this way, JBOWL can be used as a standard environment for testing and benchmarking of algorithms or implementations from various suppliers.

General classification models enable to implement common modules for visualization and model interpretation. For example, model for decision trees and rules independent of particular algorithm allows user to implement common module that will describe models with rules in limited natural language.

Note that users have many possibilities how to implement selected algorithm. For example, our implementation of SVM algorithm is simply wrapper around the SVMlib library. In this way, other packages like Weka can be integrated. Algorithms can even be implemented in different programming languages (i.e. C, C++) and integrated into JBOWL with Java Native Interface.

Following subsections describe implemented classification algorithms and some experiments that we have done in JBOWL environment.

Instance based learning - kNN

The kNN algorithm [8] is simple: given a new document, the system finds the k nearest neighbors among the training documents, and uses the categories of the neighbors to weight the category candidates. The similarity score of each neighbor document to the new document is used as the weight of the categories of the given neighbor. By sorting and thresholding the scores of candidate categories, binary category assignments are obtained.

Since JBOWL provide support for document vector model, implementation of kNN algorithm was very simple. In document analyzing task user can select various components for term frequency, inverse document frequency and normalization factor and in this way, it is possible to create document vector model with various weighting schemes (i.e. binary, tf, tf-idf weighting).

As settings for kNN algorithm user can select distance or similarity function (for convenience, the cosine value of two document vectors is used as a similarity function) and tresholding strategy used to binary category assignments.

One practical disadvantage of kNN classifier is that it requires large memory size to store all training documents. Since to find *k* nearest neighbors classifier has to compute similarity score to all training documents, classification of new unlabeled document can be time consuming. Using attribute selection and instance reduction methods can reduce these disadvantages.

For attribute selection, we have evaluated combination of three JBOWL attribute selection models, including ranking selection based on document frequency, information gain and mutual information. For instance reduction we have adopted algorithm called Decremental Reduction Optimalization Procedure 4 (DROP). More information about experiments was published in [9].

SVM

The support vector machine (linear) classifier [10] attempts to find, among all the surfaces β_1, β_2 ... in *n*-dimensional document space that separate the positive from the negative training examples, the β_m that separates the positives from the negatives by the widest margin.

Since learning of SVM requires solving of quadratic optimization problem JBOWL implementation is based on SVMlib library [11] that is optimized for large sparse matrices. We have tested our implementation on standard Reuters-2157 dataset and dataset of news articles in Slovak language (comparable in number of categories and documents to Reuters).

Decision trees and decision rules

A decision tree classifier [12] is a tree in which internal nodes are labeled by attributes (words), branches departing from them are labeled by tests the weight that attribute has in the test document, and leafs are labeled by categories. Decision tree categorizes a test document by recursively testing for weights that the attribute labeling the internal nodes have in document vector, until a leaf reached.

Closely related to decision trees are rule-based algorithms, where each category is represented as a list of rules in disjunctive normal form (DNF) [12].

A possible method for learning a decision tree for given category consists in a "divide and conquer" strategy when (1) algorithm checks if all examples have the same label and (2) if not it selects attribute test which split documents into subsets with the same category label placing each subset in separate subtree. This process is recursively repeated on subtrees until each generated leaf contains documents assigned to the same category.

Such a "fully grown" tree may be prone to overfitting (some branches may be too specific to the training documents). Most decision trees methods thus includes a method for pruning, which removes overly specific branches.

Similar top-down strategy can be used for induction of decision rules. Rules induction methods attempt to find a compact "covering" rule set that completely partitions the documents into their correct categories. The covering set is found by heuristically searching for a single "best" rule that covers documents from one category. "Best" rule is added to the rule set and documents covered by it are removed from the training set. The process is repeated until no documents remain to be covered.

JBOWL implementation of decision trees learning algorithm is decomposed to attribute test evaluation function, search strategy and pruning method. In this way user can build algorithm that uses various combinations of methods for tree growing and pruning. For example user can implement algorithm, which will use information gain criterion for test evaluation as in C4.5 algorithm and cost-complexity pruning as in CART.

Algorithm for rule induction is decomposed in the similar way, so user can implement base algorithms such as FOIL, or complex algorithms like RIPPER which consists of growing and pruning phase of individual rule and global optimization of the final generated rule set.

Both implementations support learning on weighted training set, so they can be used directly as a base algorithms in boosting method.

4 Clustering of Texts

The goal of text clustering is to find some unseen categories (or clusters) in the set of analysed documents. Unlike text categorization, text clustering is the case of unsupervised learning task. JBOWL support for text clustering is based on self-organizing map architecture that is suitable for high dimensional data produced in text mining.

The self-organizing map method [13] is a non-linear projection, which maps a high-dimensional data collection to an organized two-dimensional output. SOM executes a cluster analysis with given feature vectors of the input document collection and result is a map grid. The initial map consists of neurons, which have associated vector of randomly initialised weights. The training process of the map is sequential where algorithm: (1) finds closest neuron (winner) to the presented document vector and (2) updates winner's weights to achieve a topological landscape of the document collection.

One disadvantage of SOM maps is their static architecture. Related to SOM is GHSOM architecture (Growing Hierarchical SOM [14]) that is designed to dispose of this problem. GHSOM grows the initial map (some rows or columns of neurons are added) and creates hierarchy of SOMs with possibly many layers.

Implementation of GHSOM integrated to JBOWL consists of clustering algorithm and visualization and evaluation method. We have implemented modified GHSOM algorithm [15], which avoids some problems with growing of the map and initialisation of the new layers. Compared to original method, modified version adds only single neuron in the growing phase that avoids appearance of un-initialised neurons when the whole row or column is added.

Currently output of the visualization and evaluation method is set of HTML pages generated for each layer. For each neuron, list of characteristic content bearing terms sorted according to their variability of occurrences in covered documents are extracted together with information about number of documents assigned to cluster [18].

The quality of the document cluster analysis depends on the text analysis task used for document pre-processing. This is noticeable mainly for inflective languages like Slovak or Czech with complex morphology and syntax. To improve quality of clustering analysis of documents in Slovak language, JBOWL provides NLP methods for morphological and syntactical analysis [19]. Implemented morphological filter assigns tokens to grammatical categories (i.e. POS, case, number etc.) required for syntactical analysis and transform various word forms to base form (lemmatisation). Syntactical analysis can be used to recognize nominal phrases denoting one concept or proper name (i.e. "artificial intelligence", "European Union"). Note that all token filters are general and can be used for other text mining tasks.

5 Webocrat Scenario

The Webocracy[1] project addressed the problem of providing new types of communication flows and services from public institutions to citizens, and improving the access of citizens to public administration services and information. The new types of services increase the efficiency, transparency and accountability of public administration institutions and their policies toward citizens.

[1] IST-1999-20364 Webocracy: “Web Technologies Supporting Direct Participation in Democratic Processes”

Within this project a *WEBOCRAT* system[2] has been designed and developed [17]. *WEBOCRAT* system is a Web-based system comprising Web publishing, computer-mediated discussion, virtual communities, discussion forums, organizational memories, text data mining, and knowledge modeling. The *WEBOCRAT* system supports communication and discussion, publication of documents on the Internet, browsing and navigation, opinion polling on questions of public interest, intelligent retrieval, analytical tool, alerting services, and convenient access to information based on individual needs.

WEBOCRAT intelligent retrieval mechanism is built on top of JBOWL functionality. Document analysis, indexing, vector representation and API for text mining have been used as a basis. Three different text-mining tasks have been experimented for their possible exploitation within or for the *WEBOCRAT* system [16]. *Clustering* and *association rules* mining are used to support development and maintenance of the ontology. But the most important is use of *text categorisation* support for semi-automatic annotation of newly added text resources as well as for automatic routing of users' informal submissions to particular department.

Moreover, *full-text retrieval* mechanism has also been added and tightly integrated with the concept-based retrieval. That means, if the user starts e.g. with full-text search, in addition to the results of the full-text query he/she will get also a list of relevant concepts and clicking to them, switch to concept-based retrieval is performed.

Support for design and maintenance of the ontology

Clustering does not fit the functionality of the *WEBOCRAT* system, because documents in *WEBOCRAT* system are primarily organized by their links to knowledge model so that primarily knowledge model is used for document retrieval and topic-oriented browsing. On the other hand, it could be useful to use techniques like GHSOM, because of its hierarchical structure that is tailored to the actual text data collection, *as a supporting tool within the initial phase, when the knowledge model of a local authority is being constructed*. This is true in such a case when local authority has a representative set of text documents in electronic form available for this purpose. It is assumed that these documents will be later on published using the *WEBOCRAT* system and linked to the knowledge model for intelligent retrieval purposes.

But users must be aware of the fact, that GHSOM does not produce any ontology. It is just a hierarchical structure, where documents are organized in such a way that documents about similar topics should be topologically close to each other, and documents with different topics should be topologically far away from each other. Particular node in this hierarchical structure is labeled by (stemmed) words – terms, which occur most often in cluster of documents presented by this node. This list of terms can provide some idea about concept(s), which can be (possibly) represented in the designed knowledge model.

Finally, particular documents represent leave nodes of this hierarchical structure. It is in our opinion necessary to look carefully through the whole structure, including

[2] http://www.webocrat.sk/

particular documents in order to make reasonable conclusions about particular concepts proposed for the knowledge model and relations among them.

Association rules can be exploited e.g. for automatic improvements of the knowledge model in the following way. When we use as input attributes for association rules mining algorithm only concepts to which documents are linked that means that we are looking for frequently occurring linking patterns. These patterns can be confronted with the actual ontology. When e.g. the algorithm finds association between concepts X and Y and in the ontology no relation between concepts X and Y is presented, one can expect a missing relation between them.

Support for semi-automatic annotation of documents to concepts from ontology

In the *WEBOCRAT* system, ontology is used as a knowledge model of the domain, which is composed from concepts occurring in this domain and relationships between these concepts. Information is stored in the system in the form of mainly text documents, which are annotated by set of concepts relevant to the document content.

One strategy for document retrieval is based on concepts. User selects interesting concepts and asks for information related to them. The decision about document relevance to the user query is based on a similarity between set of query concepts and a set of concepts, which are annotated to the document. This task of document retrieval can be viewed as a classification task when the decision is made, whether the document is relevant for the user or not. With appropriate ontology which models domain well, use of this knowledge model can yield better results than e.g. retrieval based on vector representation of documents.

Retrieval accuracy depends on the quality of documents annotation. *Data mining methods can be very useful to guide user at annotating new document*. Annotation of the new document is the classification task ***(text categorization task)*** when we need to make decision which concept (concept represents category) is relevant to the content of the document.

The system must propose relevant concepts for new document in real time, so important requirement to used algorithm is execution time efficiency. User can add or delete some associations between new document and concepts, and these changes can be immediately integrated into classifier. This requires ability of incremental learning. Relevance weighting of the concepts to the new document is better than simple binary decision. Concepts are ordered by weight of the relevance to the new document and user can search for additional relevant concept according to this ordering.

6 Conclusions and Future Work

In this paper a new research and educational toolkit has been described. Proposed Java library for support of text mining and retrieval provides an easy extensible and easy to learn modular framework for preprocessing and indexing of large text collections, as well as for creation and evaluation of supervised and unsupervised text-mining models.

Different target user groups are expected to have interest to use this toolkit.

- Text mining researcher, who wish to develop and test new text mining methods;
- Application developers – Java programmers who wish to use text mining API for building of WEB or GUI applications;
- Component developers – that intend to implement framework extensions or integrate existing software with (part of) functionality of our framework;
- Students – Java knowledgeable students who have a basic understanding of the problems that text mining can solve.

We plan to provide our system for the research community for free and we expect the research community to use it for various experiments and extend it with new features (coordinating these efforts with our group). Currently, we are working on the following new modules – word sense disambiguation module; improved NLP methods for preprocessing of texts in Slovak language; vector representation of RDF sources with fuzzy retrieval and interface to OWL ontologies.

Acknowledgements

The work presented in the paper was supported by the Slovak Grant Agency of Ministry of Education and Academy of Science of the Slovak Republic within the 1/1060/04 project ”Document classification and annotation for the Semantic web”.

References

[1] Jakarta Lucene project, URL: http://jakarta.apache.org/lucene/docs/index.html, 2004
[2] Cunningham, H., Maynard, D., Bontcheva, K., Tablan, D.: GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02). Philadelphia, July 2002.
[3] Java NLP project: URL: http://www-nlp.stanford.edu/javanlp/, 2004
[4] Witten, I. H., Frank, E.: Data Mining: Practical machine learning tools with Java implementations, Morgan Kaufmann, San Francisco, 2000
[5] Bednar, P., Paralic, J.: KDD Package. In Proc. of the Znalosti 2003 Conference (Vojtech Svatek Ed.), Ostrava, February 2003, Czech Republic, ISBN 80-248-0229-5, pp. 113 – 122
[6] JSR 73: Data Mining API: URL: http://www.jcp.org/en/jsr/detail?id=73, 2004
[7] Staab, S. Studer, R.: An extensible ontology software environment, In *Handbook on Ontologies*, chapter III, pp. 311-333. Springer, 2004.
[8] Yang, Y. Chute, C. G.: An example-based mapping method for text categorization and retrieval. ACM Transaction on information systems, pp.12(3): 252-277, 1994
[9] Bednar, P., Futej, T.: Reduction techniques for instance based learning. BASYS, 2004
[10] Joachims, T.: Text categorization with support vector machine: Learning with many relevant features. European Conference on Machine Learning (ECML), pp. 137-142, Berlin, Springer. 1998.
[11] Chang, C-C., Lin, C-J.: LIBSVM: a Library for Support Vector Machines. 2004

[12] Apté, C., Damerau, F., Weiss, S.: Towards language independent automated learning of text categorization models. *Research and Development in Information Retrieval*, pp. 23-30, 1994
[13] Kohonen, T.: Self-organizing maps. Springer-Verlag, Berlin, 1995
[14] Dittenbach, M., Merkl, D., Rauber, A.: Using growing hierarchical self-organizing maps for document classification. In Proc. of European Symposium on Artificial Neural Networks – ESANN 2000, Bruges, April 2000, Belgium, ISBN 2-930307-00-5, pp. 7-12
[15] Butka, P.: Clustering of textual documents (*in Slovak*). Master thesis. Department of Cybernetics and Artificial Intelligence, Technical University, Kosice, 2003
[16] Paralič, J. – Bednár, P.: Text Mining for Documents Annotation and Ontology Support. A book chapter in: "Intelligent Systems at the Service of Mankind" (W. Elmenreich, T. Machado, I.J. Rudas eds.), Ubooks, Germany, 2003, pp. 237-248
[17] Paralič, J. – Sabol, T. – Mach, M.: Knowledge Enhanced e-Government Portal. Proc. of the 4th IFIP International Working Conference on Knowledge Management in Electronic Government (KMGov 2003), Rhodes, Greece, May 2003, LNAI 2645, pp. 163 – 174
[18] Rauber, A.: On the labeling of self-organizing maps. In Proc. of International Joint Conference on Neural Networks – IJCNN 1999, Washington DC, July 1999, USA
[19] Blicha, M.: Spracovanie textových dokumentov v slovenskom jazyku s využitím nástrojov lingvistickej analýzy pre účely ich zhlukovania a kategorizácie. Master thesis. Department of Cybernetics and Artificial Intelligence, Technical University, Kosice, 2003