TMT: Object-Oriented Text Classification Library

Artur Šilić, Frane Šarić, Bojana Dalbelo Bašić, Jan Šnajder
Faculty of Electrical Engineering and Computing, University of Zagreb
Unska 3, 10000 Zagreb, Croatia
{*Artur.Silic, Frane.Saric, Bojana.Dalbelo, Jan.Snajder*}*@fer.hr*

Abstract. *This paper describes an object-oriented library for text classification called TMT (Text Mining Tools). We put forward the advantages of object-oriented design and describe the design requirements as well as the functionalities of the library. We give an example of an integration of the TMT into an end-user application.*

Keywords. Text classification, document indexing, object-oriented design, programming library.

1. Introduction

The purpose of the TMT (Text Mining Tools) library is to enable the use of modern text-mining techniques for natural languages on cross-platform environments that can be applied equally well to research and development of end-user text-mining applications.

The TMT library is developed in the C++ programming language. This is because we wanted to use a programming language that produces fast executable code, compiles on multiple platforms, is easy to integrate into existing programs, and makes possible the use of many existing math or text mining oriented libraries.

The TMT library is relatively complex so the object-oriented paradigm comes as a natural choice, allowing the decomposition of the modules into a hierarchical class system with inheritance and encapsulation. These concepts have proven to be important and speed up the development of new modules.

Software design decisions are made as the TMT library is being developed and the most important feedback comes from the researchers who implement new modules into the library. On the other hand, the following development guidelines are persistent [16]:

- computational efficiency – design and implementation should be efficient in memory usage and CPU time,
- code reusability – abstract methods should enable reusability of the already implemented methods,
- modularity – the modules should be able to change without affecting other modules,
- rapid research cycle – users should be able to easily integrate and test new methods within the library,
- rapid application development – the structures and the methods of the library should enable intuitive and fast integration into end-user applications.

The paper is structured as follows. In the next section we discuss the related work. Section 3 describes the functionalities of the library, whereas Section 4 describes its usage. Section 5 concludes the paper.

2. Related Work

Any general data mining library can be used for solving text classification problems as those are close to many data mining problems. One widely used data mining library is Weka [17]. It is designed for solving the traditional machine learning problems requiring a moderate number of features with a moderate number of values and therefore it is not well suited for large data sets. In order to use Weka for text classification purposes, one would have to apply some feature selection method. Orange [2] is another good data mining library. It is more suitable for processing larger data sets and has a wide range of advanced visualization options, but it lacks text-oriented modules.

Transforming a set of documents into a format more appropriate for such general purpose libraries can be done using a tool like Bow [7]. Text documents are transformed into a vector feature space model, along with some basic transformations like pruning word counts, TFIDF, etc. Unfortunately a lot of tools, including Bow, lack stemming, lemmatization, or more advanced language-specific preprocessing steps.

In contrast to general purpose data mining libraries, text mining libraries usually include all the necessary text preprocessing steps. ADCS [16] is a very good example of a text classification framework written in Perl. Unfortunately, it lacks basic text preprocessing capabilities such as lemmatization. Mallet [8] is another text mining library written in Java that has several classifiers, but is missing some text processing features and more recent classifiers such as SVM. There are several other text mining libraries such as Text-Garden [10] or JavaNLP [5], but at the time of writing they are closed source or have proprietary file formats. Another example of a text categorization toolkit is TECAT, which is written in ANSI C and has similar design requirements as the TMT library (speed, modularity, experimentation, and portability) [9].

The TMT library strives to overcome most of the mentioned shortcomings.

3. Description of the library

The modular scheme of the TMT library is shown on Fig. 1. The shown modules and links between them provide the functionalities described in the following subsections.

3.1. Tokenization

Tokenization converts the raw text into a series of tokens (e.g., words). This process finds beginnings and endings of the words and optionally removes formatting such as capitalization. It is worth noting that the whole system operates in UTF-8 format so it can be easily adapted to support other languages.

Stopword removal is supported by including a file with a list of stopwords.

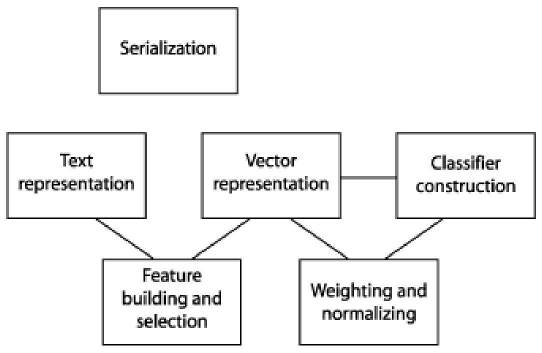


Figure 1: Association of the modules in TMT library

3.2. Lemmatization and stemming

Lemmatization converts a word from its original text form into a normalized linguistic form called the *lemma*. In this way various word forms are conflated into a single normalized form resulting in a more compact vector space model. In the TMT library, lemmatization is performed using a lemmatization dictionary. Lemmatization of different languages (or sublanguages) can be achieved by using different lemmatization dictionaries. The lemmatization dictionaries are implemented as minimal Finite State Automata (FSA). For example, FSA containing Croatian dictionary of cca. 500,000 (word, lemma) pairs consumes only 300KB of memory. Lookup takes $O(m)$ time, where m is the length of the word [1].

Stemming achieves an even greater degree of conflation in that it computes the root of a word instead of its lemma. This is typically done by applying simple stripping rules. Currently, the TMT library uses the Porter's algorithm [12] for English language stemming.

3.3. Feature building, selection, and weighting

Feature building in the TMT library is designed to be very flexible; the text can be vectorized as a *bag-of-words*, as letter or word based n-grams, or as a combination of these. Special care needs to be taken to ensure that during both training and testing phases the documents are mapped to the same vector space model. In the TMT library the text preprocessing, feature selection, and weighting are altogether described with a single object –

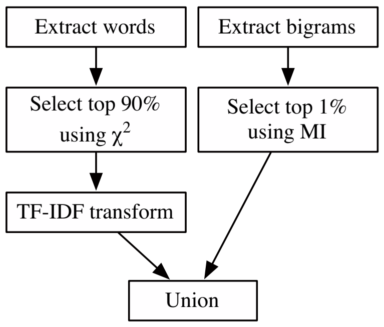


Figure 2: An example of a feature building object

the `FeatureBuilder` object. Once this object is generated, it can be made available to training and testing programs using serialization.

The TMT library makes it possible to define the feature building object by a single command line parameter. For example, the feature building object that selects 90% words using the χ^2 measure and 1% of word bigrams using the mutual-information measure is shown in Fig. 2. This particular object can be build by specifying the following option: `feature=tfidf(select[chi2,sum,0.9](words)), select[mi,sum,0.01] (bigrams)`.

3.4. Classifier training

The TMT library provides abstract classes to define interfaces for binary and multi-class classifiers. Since the TMT library uses supervised learning techniques to train the classifiers, the input documents have to be preclassified. After the training phase the obtained classifier object can be serialized. This provides an effective solution for the integration of an arbitrary classifier into other programs.

The classification methods are a good example of reusability in the TMT library. The class diagram of the classification methods is shown in Fig. 3. So far, the TMT library supports k-NN, Bayes, SVM, Winnow, and Rocchio classification methods. Many classifiers inherit the abstract class `BinaryClassifier`, which allows their use in the boosting algorithm, in a committee of classifiers, as a part of multi-class classifier, and other classifier aggregation methods. This demonstrates the concepts of code reusability and rapid research cycle.

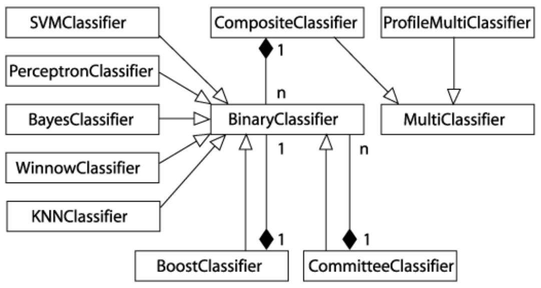


Figure 3: Diagram of classes used for different classifying methods

3.5. Evaluation of classifiers

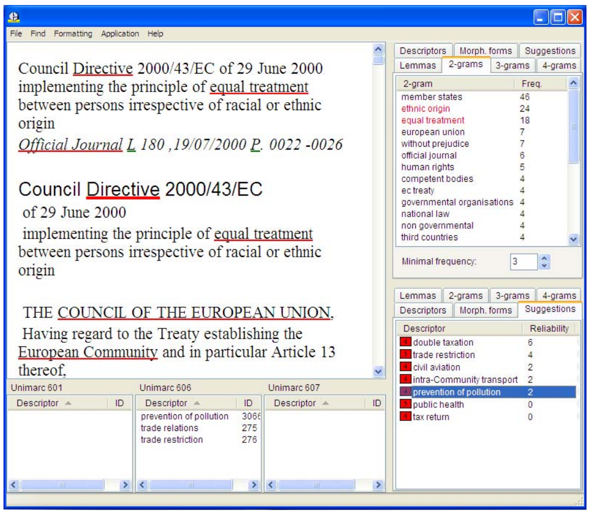
The TMT library provides classes that can be used to evaluate the classification effectiveness of a classifier by any standard measure [13] as well as the computation time and memory usage. The classifiers can be evaluated using predefined testing and training document sets, using the k-fold cross validation or using the holdout validation.

3.6. Building classifier committees

Classifier committees are made out of different binary classifiers. For example, the SVM and the k-NN classifier can yield different results, so the TMT library makes it possible to combine these results using an arbitrary function.

3.7. Hierarchical classification

If the number of classes is large and a hierarchical structure exists among them, then a divide-and-conquer approach to classification is appropriate. Specific classes are grouped into abstract ones yielding a tree structure. In that way a classifier for each tree node can be built. Each document is not tested against specific categories unless it has passed the classification at the upper abstract levels. Benefits from using hierarchical classification include performance gains with respect to hierarchical relationships and reduced training and testing times [14]. In the TMT library the basic hierarchical classification methods are provided.



Council Directive 2000/43/EC of 29 June 2000
implementing the principle of equal treatment
between persons irrespective of racial or ethnic
origin
Official Journal L 180 ,19/07/2000 P. 0022 -0026

Council Directive 2000/43/EC
of 29 June 2000
implementing the principle of equal treatment
between persons irrespective of racial or ethnic
origin

THE COUNCIL OF THE EUROPEAN UNION,
Having regard to the Treaty establishing the
European Community and in particular Article 13
thereof,

2-gram	Freq.
member states	46
ethnic origin	24
equal treatment	18
european union	7
without prejudice	7
official journal	6
human rights	5
competent bodies	4
ec treaty	4
governmental organisations	4
national law	4
non governmental	4
third countries	4

Minimal frequency: 3

Descriptor	Reliability
4 double taxation	6
3 trade restriction	4
4 civil aviation	2
4 intra-Community transport	2
4 prevention of pollution	2
5 public health	0
4 tax return	0

Unimarc 601

Descriptor	ID

Unimarc 606

Descriptor	ID
prevention of pollution	3066
trade relations	275
trade restriction	276

Unimarc 607

Descriptor	ID

Figure 4: A screenshot of CADIS

3.8. Optimization of classifier parameters

When working on a new classifier, the researchers tweak the training parameters "by hand" or by making a script that sweeps across various parameters values. However, if the training time is low enough it is reasonable to use optimization algorithms over the training parameters. In the TMT library these algorithms include the simplex method, genetic algorithms, simulated annealing, etc.

3.9. Serialization of intermediate results

The objects intended to be reused can be serialized and later deserialized. An example for this is a sparse matrix or a trained classifier. Serialization in the TMT library supports storing and loading various data structures such as arrays, linked lists, graphs, and polymorphic objects. Data can be stored in a binary or a text file.

4. Using the TMT library

The main idea behind this library is that it should be easily integrated into end-user applications as well to serve as a flexible set of tools for scientific research. To achieve this aim, during the development of the TMT library we were also immediately using it for both research and text-mining applications.

4.1. Typical users

We can distinguish among three different groups of TMT library users:

Researchers. Use existing methods in simple programs or by executing compiled command line programs. In the latter case, they are advised to look at the list of available executables and try them on sample data files. Each executable accepts unix-style command line options. The TMT library provides no GUI (Graphical User Interface) at the moment.

Researchers/developers. Research new text classifying methods by implementing them in our library. It is best for these users to learn by example by studying the source of the provided command line programs. Moreover, there is an automatically generated documentation for each class and function [3]. In principle, the user need not be aware of the underlying implementation unless doing time consuming experiments.

Developers. Make use of TMT library methods in end-user applications. They should pay attention to similar details as the previous group, but need not go into the details of the implementation.

4.2. Example of use in an end-user application

As an example of an end-user application based on TMT library we consider the Computer-Aided Document Indexing System (CADIS) [6] shown in Fig. 4. This system facilitates document indexing using controlled vocabulary keywords from the EUROVOC thesaurus [4]. It provides useful visual information such as word, lemma, and n-gram occurrences that can help human indexers choose appropriate keywords without reading the whole document. The system also offers indexers a list of keyword suggestions obtained using machine learning techniques. CADIS is currently in use by the Croatian Information Documentation Referral Agency (HIDRA).

Most of the functionality in CADIS that is not related to the user interface is pro-

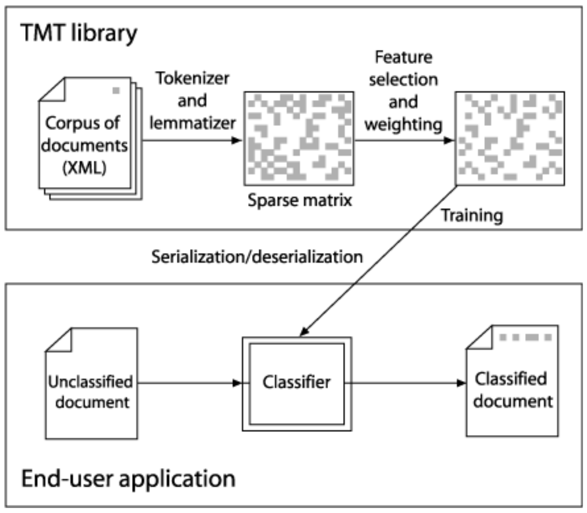


Figure 5: Updating an end-user application with a new text classifier

vided by the TMT library. That includes text and XML processing, loading controlled vocabulary, finding keyword and n-gram occurrences in the document, dealing with Croatian morphology, loading classifiers, and suggesting keywords. CADIS is usually run on desktop computers, so the memory and CPU usage had to be kept at low levels.

To enable the user to choose among different classifiers and feature building objects, CADIS uses serialization features of the TMT library to load classifier and feature building data. In this way it is also possible to process documents in both English and Croatian language since each language requires a different classifier. Moreover, serialization enables CADIS to be updated with new classifiers without reinstalling the system (Fig. 5).

5. Conclusion

We have developed the TMT library that provides a range of text classification techniques for use in research as well as in end-user applications. The TMT library is well balanced between computational efficiency and the ease of use among different user groups. Interfaces of the classes are designed to make the integration into end-user applications straightforward. On the other hand, command-line executables enable versatile research experiments.

The TMT library will be released under an open source licence soon.

Future work will endeavor more advanced text preprocessing steps such as *part-of-speech* tagging and collocation extraction, as well as other text-mining techniques apart from text classification.

Acknowledgment

This work has been jointly supported by the Ministry of Science, Education and Sports, Republic of Croatia and the Government of Flanders under the grant No. 036-1300646-1986 and KRO/009/06 (CADIAL).

References

[1] Daciuk J, Mihov S, Watson B, Watson R. Incremental Construction of Minimal Acyclic Finite State Automata, Computational Linguistics, 26(1), p. 3–16, March 2000.

[2] Demsar J, Zupan B, Leban G. Orange: From Experimental Machine Learning to Interactive Data Mining, White Paper, Faculty of Computer and Information Science, University of Ljubljana, 2005. http://www.ailab.si/orange/ [12/12/2006]

[3] Doxygen, Source code documentation generator tool. http://www.stack.nl/ dimitri/doxygen/ [01/17/2007]

[4] EUROVOC thesaurus. European Union publications office. http://europa.eu.int/celex/eurovoc/ [01/17/2007]

[5] JavaNLP Project, The Stanford Natural Language Processing Group. http://www-nlp.stanford.edu/javanlp/ [02/22/2007]

[6] Kolar M, Vukmirović I, Dalbelo Bašić B, Šnajder J. Computer-aided Document Indexing System. Journal of Computing and Information Technology, 13(4): 299–305, 2005.

[7] McCallum AK. Bow: A Toolkit for Statistical Language Modeling, Text Retrieval, Classification and Clustering, 1996.
http://www.cs.cmu.edu/ mccallum/bow/ [01/11/2007]

[8] McCallum AK. MALLET: A Machine Learning for Language Toolkit, 2002.
http://mallet.cs.umass.edu/
[02/01/2007]

[9] Montejo-Ráez A. Automatic Text Categorization of documents in the High Energy Physics domain, December 2005.

[10] Text-Garden – Text-Mining Software Tools.
http://www.textmining.net/
[01/17/2007]

[11] LIBSVM – A Library for Support Vector Machines.
http://www.csie.ntu.edu.tw/ cjlin/libsvm/
[03/05/2007]

[12] Porter MF. An Algorithm for Suffix Stripping. In: Jones S, Willet K, Willet P. Readings in Information Retrieval, 1997.

[13] Sebastiani F. Machine Learning in Automated Text Categorisation: A Survey. Technical Report IEI-B4-31-1999, Istituto di Elaborazione dell'Informazione, C.N.R., Pisa, IT, 1999.

[14] Sun A, Lim EP. Hierarchical Text Classification and Evaluation. Proceedings of the 2001 IEEE International Conference on Data Mining (ICDM 2001), p. 521–528, California, USA, November 2001.

[15] Šnajder J. Rule-based Automatic Acquisition of Large-coverage Morphological Lexicons for Information Retrieval. Tech. Report, MZOŠ. 2003-082, ZEMRIS, FER, University of Zagreb, 2005.

[16] Williams K, Calvo AR. A Framework for Text Categorization. University of Sydney, 7th Australasian Document Computing Symposium, Syndey, Australia, 2002.

[17] Witten IH, Eibe F. Data Mining: Practical Machine Learning Tools and Techniques, 2nd Edition, Morgan Kaufmann, San Francisco, 2005.