Programming Languages & Translators
(COMS W4115)
Department of Computer Science
Columbia University
Fall 2006

Automatic Text Categorization/Classification Language
(aTCl)

White Paper

Jawwad Sultan
JS2564@columbia.edu

September 26, 2006

Table of Contents

Automatic Text Categorization/Classification Language (aTCl)

1.1. Introduction:

Automated content-based document management tasks have gained a prominent status in the information systems field recently, largely due to the widespread and continuously increasing availability of documents in digital form, and the consequential need on the part of the users to access them in flexible ways. *Text categorization* (TC – also known as *text classification*, or *topic spotting*), the activity of labeling natural language texts with thematic categories from a predefined set, is one such task.

Text categorization may be defined as the task of determining an assignment of a value from {0, 1} to each entry $\mathbf{a_{ij}}$ of the decision matrix as given below:

	d_1	...	...	d_j	...	...	d_n
c_1	a_{11}	...	...	a_{1j}	...	...	a_{1n}
...	...	...	...	...	...	...	...
c_i	a_{i1}	...	...	a_{ij}	...	...	a_{in}
...	...	...	...	...	...	...	...
c_m	a_{m1}	...	...	a_{mj}	...	...	a_{mn}

where $C = \{c_1, \ldots, c_m\}$ is a set of pre-defined *categories*, and $D = \{d_1, \ldots, d_n\}$ is a set of documents to be classified. A value of 1 for $\mathbf{a_{ij}}$ indicates a decision to file $\mathbf{d}j$ under $\boldsymbol{c_i}$, while a value of 0 indicates a decision not to file **dj** under $\mathbf{c_i}$.

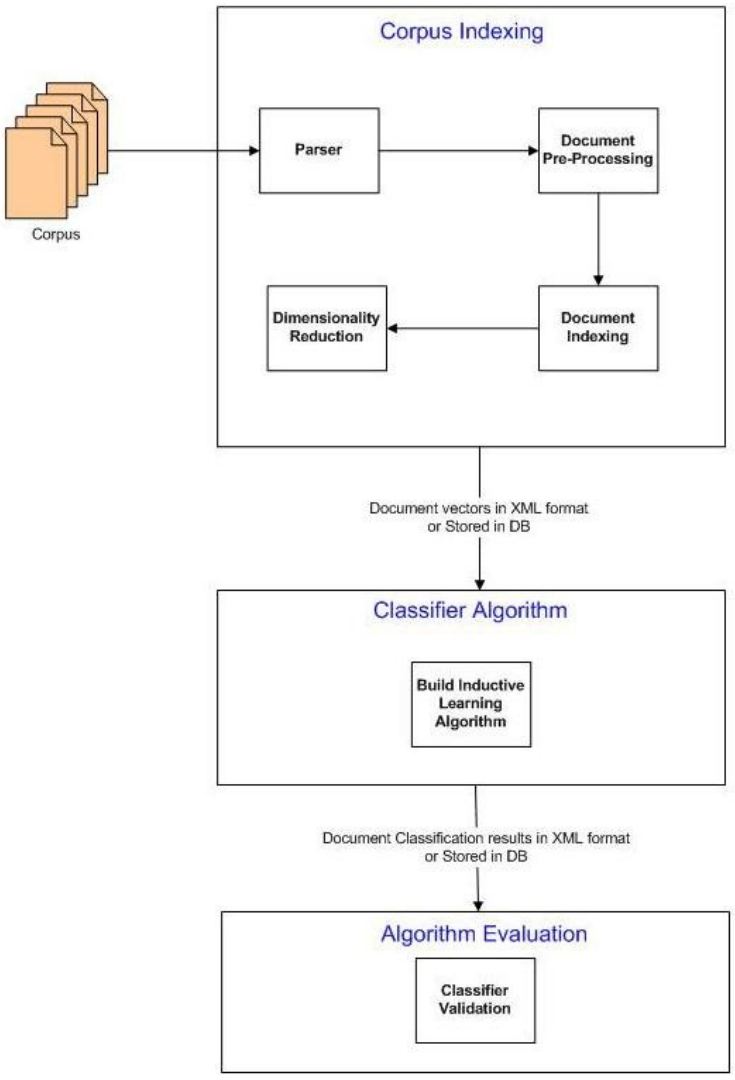
The attribution of documents to categories should, in general, be realized on the basis of the *semantics* of the documents, and not on the basis of *metadata* (e.g. publication date, document type, publication source, etc.). That is, the categorization of a document should be based solely on *endogenous* knowledge (i.e. knowledge that can be extracted from the document itself) rather than on *exogenous* knowledge (i.e. data that might be provided for this purpose by an external source).

aTCl would help extract useful knowledge from the set of documents called **Corpus** which would later be used by the Classifier Algorithm to label documents under $\mathbf{c_j}$. Machine Learning community could use aTCl to uniformly translate any available Corpus such as Reuters 21578 and the new RCV1 dataset and extract useful knowledge in the way that can later be used by the Classifier algorithm.

Future versions of aTCl might also include support for some of the most important Text classification algorithms and support for features like Boosting and K-fold cross validation. Intend is to allow users of this language to focus on the main part “Building Classifier Algorithm” instead of spending time mapping documents to different type of vectors and implementing other type of performance tuning features. Author of this language is also taking “Introduction to Computational Learning Theory” (COMS W4252) course and is planning to use the work done for this project as the basis for the project in COMS W4252.

1.2. Background:

Text Classification Life Cycle:



Parser:

Different datasets are usually presented in different formats. Reuters 21578 which is one of the most popular dataset is available in SGML format and comparatively new RCV1 has its own format for storing documents. Parser would abstract these details from low level process and provide a uniform interface for other modules.

Document Pre-Processing:

In this part several cleanup processes are applied to documents such as:

- Remove HTML (or other) tags
- Remove function words such as prepositions, conjunctions, articles

- ❖ Perform word stemming [clustering together types that share the same morphological root] example: {cluster, clustering, clustered ...}
- ❖ Noise removal.

Document indexing:

Texts cannot be directly interpreted by a classifier or by a classifier-building algorithm. Because of this, an *indexing* procedure that maps a text $\mathbf{d_j}$ into a compact representation of its content needs to be uniformly applied to training, validation, and test documents.
This module could involve features like:

- ❖ Selecting a set of terms T in the corpus (also known as features)
- ❖ Compute term weights either with binary weights indicating presence or absence of the feature in the document or weights that quantify feature occurrence in the document. Relative frequency using tfidf could also be used for term selection.
- ❖ Encoding documents as vector $\mathbf{d_j} = < \mathbf{w_{ij}} , ..., \mathbf{w_{|T|j}} >$
  $\mathbf{w_{kj}}$ represents how much feature k "contributes" to the semantics of text $\mathbf{d_j}$

Dimensionality Reduction (DR):

In TC the high dimensionality of the term space (i.e. the fact that the number of terms |T| that occur at least once in the corpus *Co* is high) may be problematic. Because of this, techniques for *dimensionality reduction* (DR) are often employed whose effect is to reduce the dimensionality of the vector space from **|T|** to **|T'|** << **|T|**. Set T' is called the reduced term set. Various different techniques include:

- ❖ DR by Tem selection or Term space reduction where **T'** is a subset of **T**.
- ❖ DR by Term extraction where Terms in **T'** obtained by combinations or transformations of the original ones using Term Clustering or Latent Semantic Indexing (LSI)
- ❖ DR by Term extraction

1.3. aTCl Introduction:

WHAT is **aTCl** language?
This high level language provides basic constructs to develop machine learning algorithms for text categorization and classification. This project would focus only on the first part of the life cycle called "Corpus Indexing". Author is planning to do "Classifier Algorithm" & "Algorithm Evaluation" in a COMS 4252 course.

WHY is it good to have language like **aTCl**?
Currently, there is no such language exists that provides capabilities to "slice and dice" corpus as per user requirements and provide much control programmatically. Some, toolkits do exists like WEKA which provides an API to perform such tasks.

Computer researchers doing research in Text classification could use this language for generating machine learning algorithms as they would now only have to spend time focusing on classifier algorithm instead of spending much time in writing code for corpus indexing.

Features of aTCl:

Simple Types:
Integers: Sequence of numbers with no decimal or exponent allowed. Used for arithmetic operations, loop control and list indexing.

Decimals: Decimal literals are a sequence of numbers followed by an optional period and a decimal part.

Strings: A string is a sequence of characters enclosed by double quotes "". A double quote inside the string is represented by two consecutive double quotes

Complex Types:
Term: Represent terms in the corpus and have attributes like, document frequency, term weight, term strength, term information gain.

Document: Collection of terms and other features like document id, document term ratio to number of original number of words, etc.

Corpus: An array of documents. Operation permitted on documents would also be permitted on Corpus which means to apply operations on every element of the corpus.

Control Statements:
Need conditional statements like "If" and loop such as "iterateAll"

Separators
Semi-colons are used to separate statements and curly brackets represent blocks of code.

Properties of aTCl:

Simple:
Main goal of the aTCl is to provide a very easy way for users to index documents and apply various other filters, dimensionality reduction techniques which if done without this language require advanced programming skills.
For e.g.: To remove stop words from the document, once could simply write an expression of the form: $\mathbf{d = d_i - d_j}$ where, d_j is some document vector having stop words as terms. To add terms from different document, one could simple write $\mathbf{d = d_i + d_j}$. The effect of this expression is to create a new document having terms from both the documents. Similarly, union and intersection operator could also be provided.

Robust:
Compile time checks to detect various syntax errors.

Portability:
Programs written in aTCl would be converted into Java program which can then be run on any platform having a supporting Java Virtual Machine implementation.

1.4. Project Planning and Development:

Iterative Development:
Project would be implemented using iterative methodology and is divided in following iterations.

- Complete working language with Parser module
- Addition of Document Pre-Processing module
- Addition of Document indexing module
- Addition of Dimensionality Reduction Module

First iteration would also encompass familiarity with Antlr and other frameworks used in the project. Each iteration will built upon the work done in previous iterations and adding some new features in the language. Following iterative approach would help build on the complete project in an efficient manner and learn from mistakes done in early iterations to achieve better overall results.

High Level Plan:
Following are the deadlines for each iteration and related submissions.

September 26, 2006	Language White Paper
October 14, 2006	Iteration 1
October 19, 2006	Language Reference Manual
November 7, 2006	Iteration 2
November 26, 2006	Iteration 3
December 12, 2006	Iteration 4

Development Environment:
This project would be developed on Windows XP 64 bit platform using JRockit 5.0 VM. Antlr would be used for generating Lexical Analyzer, Parser and Tree/Walker. The project would be tested using JUnit framework. Subversion with Tortoise SVN would be used for version control.

Testing:
Testing at various stages of the project would be necessary for the successful completion of the project with minimal amount of bugs.

Documentation:
All the project source files would be properly documented.

Project Risks:
Familiarity with Text Categorization process.

- To mitigate risk author has read survey papers related to TC.

No prior experience with Language construction and Antlr.

- To mitigate risk author is experimenting with Antlr using small examples

1.5. Appendix:

Project Log:

September 26, 2006	Language White Paper

References:

Machine learning in automated text categorization
Fabrizio Sebastiani
March 2002
ACM Computing Surveys (CSUR), **Volume 34 Issue 1**

A Comparative Study on Feature Selection in Text Categorization
Yiming Yang, Jan O. Pedersen
Proceedings of ICML-97, 14th International Conference on Machine Learning

Dataset:
Either Reuters 21578 or RCV1 would be used for corpus indexing.