Programming Languages & Translators

(COMS W4115)
Department of Computer Science
Columbia University
Fall 2006

Automatic Text Categorization/Classification Language

(aTCl)

Final Report

Jawwad Sultan
JS2564@columbia.edu
December 19, 2006

Distribution

Copy Number	Name, Title	Location
1	Prof. Stephen A. Edwards sedwards@cs.columbia.edu	Course Professor for COMS W4115

Document Control

Change Record

Date	Author	Version	Comments
09/26/2006	Jawwad Sultan	0.1	Language White Paper
10/19/2006	Jawwad Sultan	0.2	aTCl sample program added in the tutorial section. Language Reference Manual added. Reuters 21578 corpus details added in the appendix section.
12/19/2006	Jawwad Sultan	1.0	aTCl final report published. Sections like aTCl architecture, tutorial, test plan added to the report.

Table of Contents

1 Introduction

1.1 Purpose

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 build “Classifier Algorithm” & do “Algorithm Evaluation” in a COMS 4252 course.

WHY is it good to have language like **aTCl**?
Currently, 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.

1.2 Scope

This document starts with basic information about Text Categroization and the need of aTCl languiage. It provides user friendly aTCl tutorial, Langauge Reference Manual and core architecture of the language. Project planning, Test suite and source code listing is also being added.

1.3 Definitions, Acronyms and Abbreviations

aTCl – Automated Text Categroization/Classification Language
ANTLR – Another Tool for Language Recognition

1.4 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:
Reuters 21578 and 20Newsgroup dataset were used in the project.

2 Text Categorization

2.1 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 $\mathbf{c}_i$, while a value of 0 indicates a decision not to file $\mathbf{d_j}$ 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.

2.1.1 Machine Learning approach:

With the rapid growth of online information available and proliferation of WWW since early 90's the ML approach to TC has gained popularity and has eventually become the dominant one and has almost automated the classifier construction process. Domains experts job now has changed to create a Training set of documents which an inductive learner processes and automatically build a classifier for a category c_i by observing the characteristics of a set of documents manually classified as c_i or $\check{c}_i$ by a domain expert; from these characteristics, the inductive process gleans the characteristics that a new unseen document should have in order to be classified under c_i.

In ML approach, initial set of documents (Corpus) is available in the form $\mathbf{\Omega = \{ d1, \ldots , d_{|\Omega|} \} \subset D}$ of documents pre-classified under $\mathbf{C = \{c_1, \ldots, c_{|c|}\}}$ such that the target function $\mathbf{\Phi'}$ is known. A document $\mathbf{d_j}$ is a *positive example* of a category $\mathbf{c_i}$ if $\mathbf{\Phi' (d_j, c_i) = T}$, a negative example if $\mathbf{\Phi' (d_j, c_i) = F}$.

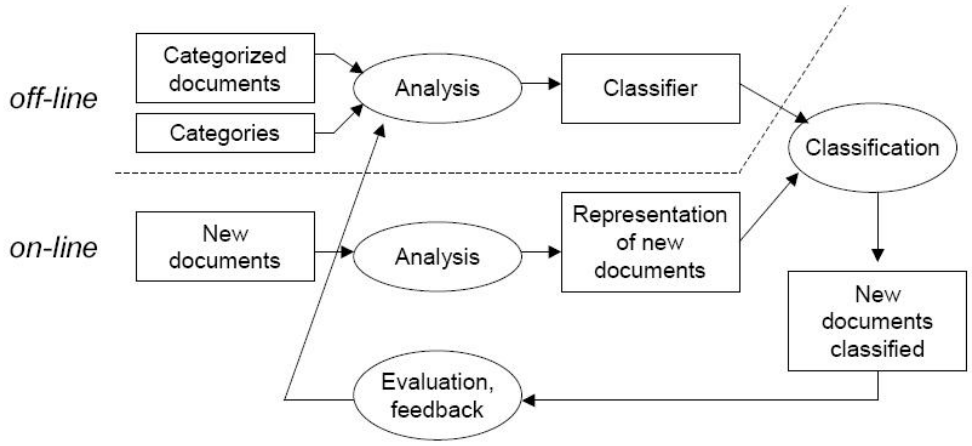
Prior to classifier construction, corpus is normally divided into three subsets as follows.

Training Set ($\mathbf{T_r}$) : on which the initial classifier is inductively built off-line.

Validation Set (T_v) : also known as hold-out set, on which the initial classifier is optimized off-line, and
Test Set (T_e) : on which the classifier is evaluated on-line.

The documents in T_e cannot participate in any way in the inductive construction of the classifiers, otherwise experimental results obtained would likely be unrealistically good and evaluation would then have no scientific characteristics. This is called the train-and-test approach and an alternative to it is the usually taken **k-fold cross validation** approach.

In k-fold cross validation, initial pre-classified corpus is partitioned into k disjoint sets $\mathbf{Te_1}$, … , $\mathbf{Te_k}$ and k different classifiers are built using test and train approach applied on T_r and T_v and then the final effectiveness of the classifier is then some average of individual classifiers Φ_1 , … , Φ_k.



Outline of ML approach applied to TC

Now, as we have introduced TC and have provided some basic information about ML apporahc to it we further discuss in detail different steps involved in the process.

2.2 Text Classification Life Cycle

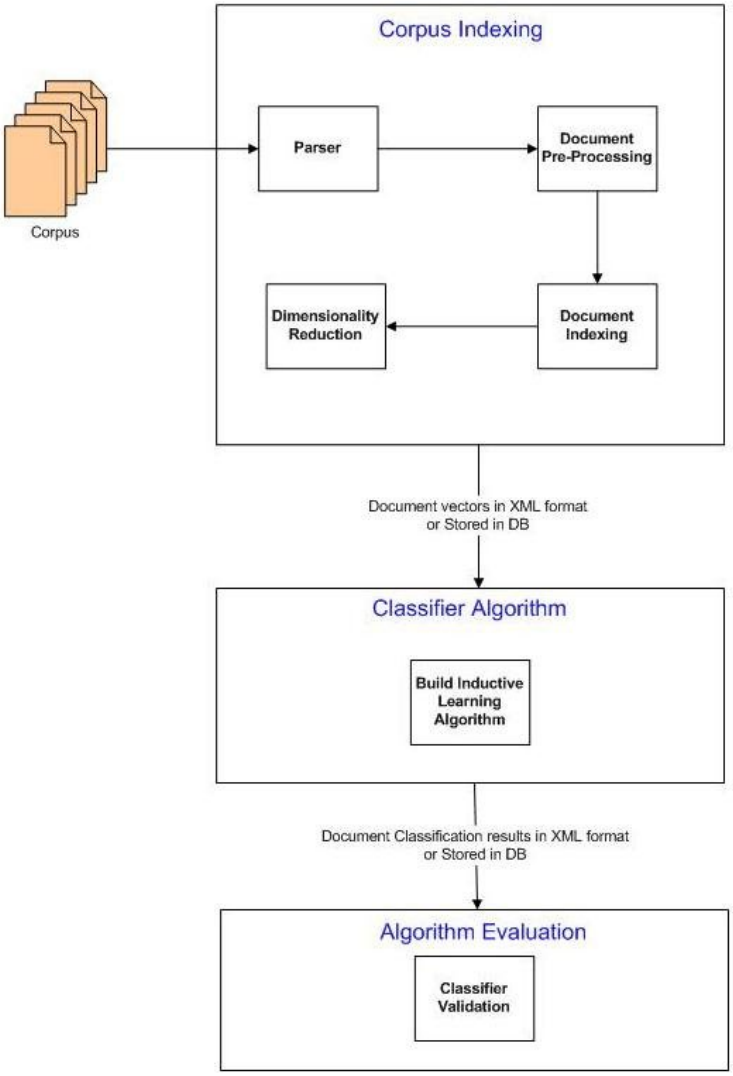


Figure 1 TC Life Cycle

2.2.1 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.

2.2.2 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.

2.2.3 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 $\mathbf{d_j = < w_{ij} , ...,w_{|T|j} >}$
- $\mathbf{w_{kj}}$ represents how much feature k "contributes" to the semantics of text $\mathbf{d_j}$

2.2.4 Dimensionality Reduction

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

3 aTCl Language

3.1 Features of aTCl

3.1.1 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

3.1.2 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.

3.1.3 Control Statements

Need conditional statements like "If" and loop such as "iterateAll"

3.1.4 Separators

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

3.2 Properties of aTCl

3.2.1 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.

3.2.2 Portable

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.

3.2.3 Robust

Compile time checks to detect various syntax errors.

4 aTCl Tutorial

4.1 Sample aTCl program

```
// aTCl sample program to read corpus from documents

// corpus initialization
// argument1: string literal      : corpus documents location
// argument2: string literal      : corpus type.
// argument3: string literal      : identifier to parse documents only in Training set or Test set.


Corpus reutersCorpusTraining, reutersCorpusTesting;

reutersCorpusTraining = createCorpus [“c:\ Reuters-21578_Raw\corpus”, “c:\ Reuters-21578_Raw\ 21578.g”,
”Training”];

reutersCorpusTesting = createCorpus [“c:\ Reuters-21578_Raw\corpus”, “c:\ Reuters-21578_Raw\ 21578.g”,
”Testing”];

// Now we have parsed all documents in the corpus and have created relevant vectors in the corpus type

// To remove noise from the corpus using Zipf’s Law
- reutersCorpusTraining;
- reutersCorpusTesting’

// To perform word stemming on the corpus.
+ reutersCorpusTraining stem;
+ reutersCorpusTesting;

// At this moment document pre-processing and indexing have been done and now we can apply Dimensionality
// Reduction techniques on the corpus. Term Selection, Term Extraction and Term Expansion are different methods
// used for DR. Only certain Term Selection techiniques are available in the first release of aTCl and can be applied
// alone or in any combination on the corpus.

// This applies Term Space Reduction (TSR) technique by using Document Frequency (DF) method retaining terms
// only appearing in 35% of documents.

dimensionalityReduction (reutersCorpusTraining, TS, DF,0.35);

// This would build and inductive classifier on the corpus
buildClassifier (reutersCorpusTraining, “Winnow, 2.0,0.5”);

// This would evaluate the corpus classifier
testClassifier (reutersCorpusTraining, reutersCorpusTesting,”Winnow”);

// Corpus can now be exported in different text files for later used by classification algorithm.

// This saves document vector from the corpus. Each row represents a unique Document in the corups and
// columns represent different terms appearing in the document.

saveCorpus (reutersCorpusTraining, “DOC”, “TERM”, “c:\corpusDocTermVector”);
saveCorpus (reutersCorpusTraining, DOC, CATEG, “c:\corpusDocCategVector”); // column now represnts
document category classification.
```

5 aTCl Language Reference Manual

5.1 Lexical Conventions

Lexical Conventions of aTCl are listed below.

5.1.1 Comments

aTCl supports only single line comments. All lines starting with "//" denote single-line comments.

5.1.2 Identifiers

An identifier starts with a letter or an underscore followed by any combination of letters, digits and underscores of arbitrary length. Identifiers are case sensitive so localVar and LOCALVAR are considered to be distinct identifiers.

5.1.3 Keywords

aTCl reserves following keywords:

for	if	else	do	stem	removeNoise	TS	TEX	TEP	DF
DIA	IG	MI	CS	NGLC	RS	OR	GSSC	DOC	
CATEG	TERM	true	false	Corpus					

5.1.4 Literals

aTCl has following type of literals in the language.

Integer:

Sequence of one or more consecutive digits with no decimal or exponent allowed. Integer literals are always unsigned and considered positive.

Decimal:

Sequence of one or more consecutive digits followed by an optional period and a decimal part which itself is one or more consecutive digits. Decimal literals don't support exponent and are always unsigned and positive.

String:

String literals are any sequence of acceptable characters found between double-quotes. "myString" is a string literal of following characters [mystring]. Double quotes cannot be embedded within string literals.

Bool :

True and False are two Boolean constants used in the language and represent logical TRUE and logical FALSE values.

5.1.5 Other Tokens

Following characters and sequence of charactes all have meanings:

{	}	(	)	[	]	.	,	=
<	<=	>	>=	==	!=	+	-	%
/	*	;	&	|	:	!		

5.1.6 Separators

Semi-colon is used to separate statements. Curly brackets ("{}") are used to delimit a piece of code.

5.2 Types

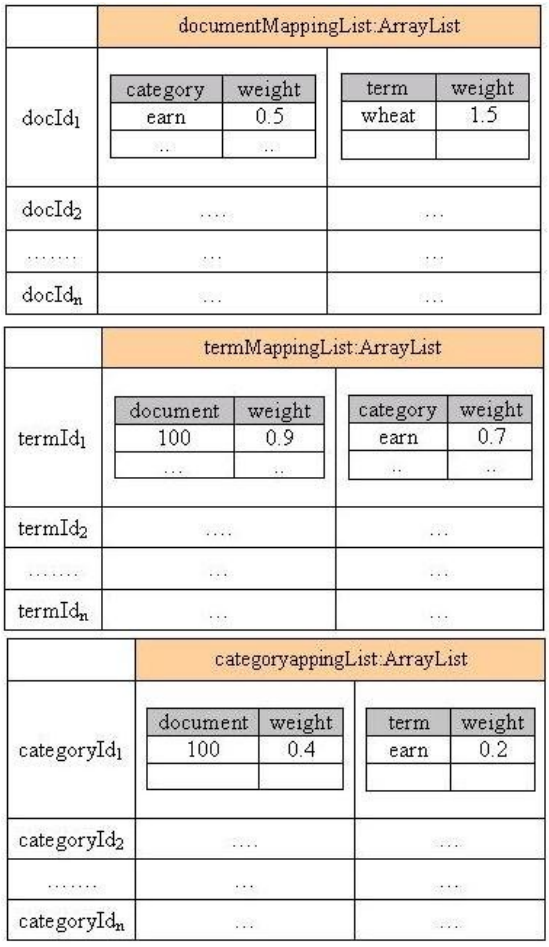
aTCl is a strongly typed language and all variables are bound to a type at declaration. Following types are supported in the language.

5.2.1 Corpus

Corpus is the representation of the set of documents, categories and terms. It has some other useful information to be used during dimensionality reduction and pre-processing modules.

Corpus		
Attribute	Type	Description
documentMap	Map	Contains mapping from documents to terms. Keys are the list of all documents in the corpus. documentMap.get (d1) returns a set in which first element is the document to category mapping and the second element is the document to term mapping.
termMap	Map	Contains mapping from terms to documents. Keys are the list of all terms in the corpus. termMap.get (t1) returns a set in which first element is the term to category mapping and the second element is the term to document mapping.
categroyMap	Map	Contains mapping from category to documents.Keys are the list of all categories in the corpus. categoryMap.get (c1) returns a set in which first element is the category to document mapping and the second element is the category to terms mapping.

Idea is to hide these maps from the normal user of the language and instead provide abstraction to do various operations at the corpus level in order to make writing programs simpler. However, these maps would also be available for use through Corpus type. Given picture displays the corpus structure which contains documents,terms and category.



5.3 Expressions

5.3.1 Primary Expressions

Primary expressions are literals, identifiers, access to corups, document and term types.

Literals
Literals are integers, strings or bools. Literal type is determined by an interpreter and they evaluate to their expressed value.

Identifiers
An identifier itself is a left-value expression. They are evaluated to the last bound value.

5.3.2 Function Calls

Functions are invoked by function identifier followed by a list of arguments enclosed within “(” and ”)”. Arguments are optional and each argument itself is an expression. Tye type of the function is the type returned by the function. Function call is a right-value expression.

5.3.3 Aritmetic Expressions

aTCl supports document addition and subtraction of sets.These expressions expressions as its operands.

Unary Arithmetic Operators
Unary operators “+” and “-” can be prefixed to a expression. They are applicable only to Corpus type with following meaning.

```
+corpus    // This applies stemming rules on the corpus using Porter Stemming Algorithm.
-corpus    // This removes noise from the corpus using Zipf’s law.
```

Binary Operators
+, -
These addition and substraction indicate addition and subtraction, respectively. Associate from left to right. Applicable to integers and sets.

5.3.4 Bool Expressions

Bool expressions always returns bool values and consist of relational expressions and logical expressions.

5.3.5 Relational Expression

Binary operators are grouped left to right and evaluated to true or false. Return bool values when applied to integers.

>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
!=	Not equal to

5.3.6 Logical Expression

Logical operators take relational expressions as operands and applicable only to bool values. These operators are short-ciruited. Listed in the order from highest precedence level to lowest.

AND	Logical and of two relational expressions.
OR	Logical or of two relational expressions.

5.4 Statements

A statement represents a command in the language and always terminated with a semi-colon (;). Statements are variable declaration, flow control construct, library function call or any other allowed expression in the language. Program flow starts from the first statement in the program to the last.

5.4.1 Identifier Declaration

<type> <identifier> = <library function> [<args>];

<type> <identifier> = <expression>;

where type is any legal aTCl type and identifier is any legal aTCl identifier. Expression is any aTCl legal expression evaluating to the identifier type.

5.4.2 If Statements

aTCl supports “if” statements to identify conditions.

if <bool-expression> { <statements> }

if <bool-expression> {<statements>} else {<statements>}

Statements are evaluated to bool expressions. An else is always connected to the most recent else-less if.

5.4.3 Iterative Statements

Supports “for” statement and is used to iterate over list of values.

```
iterate <int literal> <Set>
{
        <statements>
}
```

Iterates through all values in the set and keep updating the int_literal in each iteration.

5.4.4 Function invocation and Return Statements

Function call
Different from other expressions, a function call followed by a \;" can be a single statement.

5.5 Scope and Binding

aTCl has only file level scope with the exception of iterative statements. Any variable declared in the program is available for the complete program and its current value is the last bounded value. Varaibles defined within iterative statements are not visible after the code has exited the iterative statement block.

5.6 Library Functions

5.6.1 createCorpus ()

Corpus Declarations

Corpus reutersCorpusTraining = createCorpus [arg1, arg2, arg3, arg3, arg4];

// corpus initialization
// argument1: string literal : corpus documents location
// argument2: string literal : corpus type.

// argument3: string literal : identifier to parse documents only in Training set or Test set.

5.6.2 dimensionalityReduction ()

// This applies Term Space Reduction (TSR) technique by using Document Frequency (DF) method retaining terms
// only appearing in 35% of documents.

```
dimensionalityReduction (arg1, arg2, arg3,arg4);
```

arg1 [required]: This is the corpus identifier.
arg2 [required]: This identifies DR method. Possible values are TS for Term Selection, TEX for term extraction and TEP for term expansion.

arg3 [required]: Particular technique used in DR.
arg3 [required]: Useful parameters required for the method used in DR.

5.6.3 saveCorpus ()

```
saveCorpus (reutersCorpusTraining, DOC, TERM);
```

arg1 [required]: This is the corpus identifier.
arg2 [required]: This represents row.
arg3 [required]: This represents column.
Arg4 [required]: Output file location

6 aTCl Sample Programs and Results

6.1.1 Test1.atcl

```
Corpus reutersCorpusTraining;
Corpus reutersCorpusTesting;

reutersCorpusTraining =
createCorpus("C:\Documents and Settings\Administrator\My
Documents\Education\Columbia\Courses\Fall 2006\COMS W4252\Ass\Final Project\reuters_21578",
"REUTERS_21578","Training");

reutersCorpusTesting =
createCorpus("C:\Documents and Settings\Administrator\My
Documents\Education\Columbia\Courses\Fall 2006\COMS W4252\Ass\Final Project\reuters_21578",
"REUTERS_21578","Testing");

- reutersCorpusTraining;     // This removes Noise from the corpus using Zipf's Law
+ reutersCorpusTraining;     // This applies stemming rules on the corpus

buildClassifier(reutersCorpusTraining,"Winnow",2.0,0.5);
testClassifier(reutersCorpusTraining, reutersCorpusTesting,"Winnow");

saveCorpus (reutersCorpusTraining, "DOC", "TERM", "C:\corpusDocTermVector.tc");
saveCorpus (reutersCorpusTraining, "DOC", "CATEG","C:\corpusDocCategVector.tc");

//**********AST ************

( PROG
        ( DECL Corpus reutersCorpusTraining )

        ( DECL Corpus reutersCorpusTesting )

         ( = reutersCorpusTraining ( FUNC_CALL createCorpus
                ( ARG_LIST C:\Courses\Fall 2006\COMS W4115\reuters_21578 REUTERS_21578
                Training ) ) )

         ( = reutersCorpusTesting ( FUNC_CALL createCorpus
                ( ARG_LIST C:\Courses\Fall 2006\COMS W4115\reuters_21578 REUTERS_21578
                Testing ) ) )

        ( UNARYOP_MINUS reutersCorpusTraining )

        ( UNARYOP_PLUS reutersCorpusTraining )

        ( FUNC_CALL buildClassifier ( ARG_LIST reutersCorpusTraining Winnow 2.0 0.5 ) )

        ( FUNC_CALL testClassifier ( ARG_LIST reutersCorpusTraining reutersCorpusTesting ) )

        ( FUNC_CALL saveCorpus ( ARG_LIST reutersCorpusTraining DOC TERM
        C:\corpusDocTermVector.tc ) )

        ( FUNC_CALL saveCorpus ( ARG_LIST reutersCorpusTraining DOC CATEG
        C:\corpusDocTermVector.tc ) )
)
```

6.1.2 Test2.atcl

```
Corpus  reutersCorpusTraining, newsgroup20gTraining, reutersCorpusTesting, newsgroup20gTesting,
combinedCorpusTraining, combinedCorpusTesting;

reutersCorpusTraining = createCorpus("C:\Documents and Settings\Administrator\My
Documents\Education\Columbia\Courses\Fall 2006\COMS W4252\Ass\Final Project\reuters_21578",
"REUTERS_21578","Training");

newsgroup20gTraining = createCorpus ("C:\Documents and Settings\Administrator\My
Documents\Education\Columbia\Courses\Fall 2006\COMS W4252\Ass\Final Project\20_Newsgroup",
"NEWSGROUP_20","Training");

combinedCorpusTraining = reutersCorpusTraining + newsgroup20gTraining;

//-combinedCorpusTraining;     // This removes Noise from the corpus using Zipf's Law
//+combinedCorpusTraining;     // This applies stemming rules on the corpus

reutersCorpusTesting = createCorpus ("C:\Documents and Settings\Administrator\My
Documents\Education\Columbia\Courses\Fall 2006\COMS W4252\Ass\Final Project\reuters_21578",
"REUTERS_21578","Training");

newsgroup20gTesting = createCorpus ("C:\Documents and Settings\Administrator\My
Documents\Education\Columbia\Courses\Fall 2006\COMS W4252\Ass\Final Project\20_Newsgroup ",
"NEWSGROUP_20","Training");

combinedCorpusTesting = reutersCorpusTesting  + newsgroup20gTesting;

-combinedCorpusTesting;
+combinedCorpusTesting;

buildClassifier (reutersCorpusTraining,"Winnow",2.0,0.5);
testClassifier (reutersCorpusTraining, reutersCorpusTesting);

//**********AST ************
 ( PROG
            ( DECL Corpus reutersCorpusTraining newsgroup20gTraining reutersCorpusTesting
            newsgroup20gTesting combinedCorpusTraining combinedCorpusTesting )
            ( = reutersCorpusTraining ( FUNC_CALL createCorpus ( ARG_LIST C:\Courses\Fall
            2006\COMS W4115\reuters_21578 REUTERS_21578 Training ) ) )
            ( = newsgroup20gTraining ( FUNC_CALL createCorpus ( ARG_LIST C:\Courses\Fall
            2006\COMS W4115\reuters_21578 NEWSGROUP_20 Training ) ) )
            ( = combinedCorpusTraining ( + reutersCorpusTraining newsgroup20gTraining ) )
            ( UNARYOP_MINUS combinedCorpusTraining )
            ( UNARYOP_PLUS combinedCorpusTraining )
            ( = reutersCorpusTesting ( FUNC_CALL createCorpus ( ARG_LIST C:\Courses\Fall
            2006\COMS W4115\reuters_21578 REUTERS_21578 Training ) ) )
            ( = newsgroup20gTesting ( FUNC_CALL createCorpus ( ARG_LIST C:\Courses\Fall
            2006\COMS W4115\reuters_21578 NEWSGROUP_20 Training ) ) )
            ( = combinedCorpusTesting ( + reutersCorpusTesting newsgroup20gTesting ) )
            ( UNARYOP_MINUS combinedCorpusTesting )
            ( UNARYOP_PLUS combinedCorpusTesting )
            ( FUNC_CALL buildClassifier ( ARG_LIST reutersCorpusTraining Winnow 2.0 0.5 ) )
            ( FUNC_CALL testClassifier ( ARG_LIST reutersCorpusTraining reutersCorpusTesting ) )
)
```

6.1.3 Test3.atcl

```
Corpus reutersCorpusTraining;
Corpus reutersCorpusTesting;

reutersCorpusTraining = createCorpus("C:\Courses\Fall 2006\COMS W4115\reuters_21578",
"REUTERS_21578","Training");

reutersCorpusTesting = createCorpus("C:\Courses\Fall 2006\COMS W4115\reuters_21578",
"REUTERS_21578","Testing");

buildClassifier (reutersCorpusTraining,"Winnow",2.0,0.5);
testClassifier (reutersCorpusTraining, reutersCorpusTesting,"Winnow");

//*************TEST RESULTS *****************
//Total number of Mistakes made in Testing:821
//Total number of Correct in Testing:1747
//Total number of Not predicted in Testing:0

//*******************AST ********************
// ( PROG
//    ( DECL Corpus reutersCorpusTraining )
//    ( DECL Corpus reutersCorpusTesting )
//    ( = reutersCorpusTraining ( FUNC_CALL createCorpus ( ARG_LIST C:\Courses\Fall 2006\COMS
W4115\reuters_21578 REUTERS_21578 Training ) ) )
//    ( = reutersCorpusTesting ( FUNC_CALL createCorpus ( ARG_LIST C:\Courses\Fall 2006\COMS
W4115\reuters_21578 REUTERS_21578 Testing ) ) )
//    ( FUNC_CALL buildClassifier ( ARG_LIST reutersCorpusTraining Winnow 2.0 0.5 ) )
//    ( FUNC_CALL testClassifier ( ARG_LIST reutersCorpusTraining reutersCorpusTesting Winnow ) ) )
```

6.1.4 Test5.atcl

```
Corpus reutersCorpusTraining;
Corpus reutersCorpusTesting;

reutersCorpusTraining =
createCorpus("C:\Courses\Fall 2006\COMS W4115\reuters_21578", "REUTERS_21578","Training");

reutersCorpusTesting =
createCorpus("C:\Courses\Fall 2006\COMS W4115\reuters_21578", "REUTERS_21578","Testing");

- reutersCorpusTraining;     // This removes Noise from the corpus using Zipf's Law
+ reutersCorpusTraining;     // This applies stemming rules on the corpus

buildClassifier(reutersCorpusTraining,"Winnow",2.0,0.5);
testClassifier(reutersCorpusTraining, reutersCorpusTesting,"Winnow");

//*************TEST RESULTS *****************
//Total number of Mistakes made in Testing:1526
//Total number of Correct in Testing:1042
//Total number of Not predicted in Testing:0
```

6.1.5 Test6.atcl

```
Corpus newsgroup20gTraining,newsgroup20gTesting;

newsgroup20gTraining =
createCorpus("C:\Documents and Settings\Administrator\My
Documents\Education\Columbia\Courses\Fall 2006\COMS W4252\Ass\Final Project\20_Newsgroup",
"NEWSGROUP_20","Training");
-newsgroup20gTraining;

newsgroup20gTesting =
createCorpus("C:\Documents and Settings\Administrator\My
Documents\Education\Columbia\Courses\Fall 2006\COMS W4252\Ass\Final Project\20_Newsgroup",
"NEWSGROUP_20","Testing");
-newsgroup20gTesting;

buildClassifier(newsgroup20gTraining,"Winnow",2.0,0.5);
testClassifier(newsgroup20gTraining, newsgroup20gTesting,"Winnow");

//*************TEST RESULTS *****************
//Total number of Mistakes made in Testing:3005
//Total number of Correct in Testing:4522
//Total number of Not predicted in Testing:0
```

6.1.6 Test Results Summary

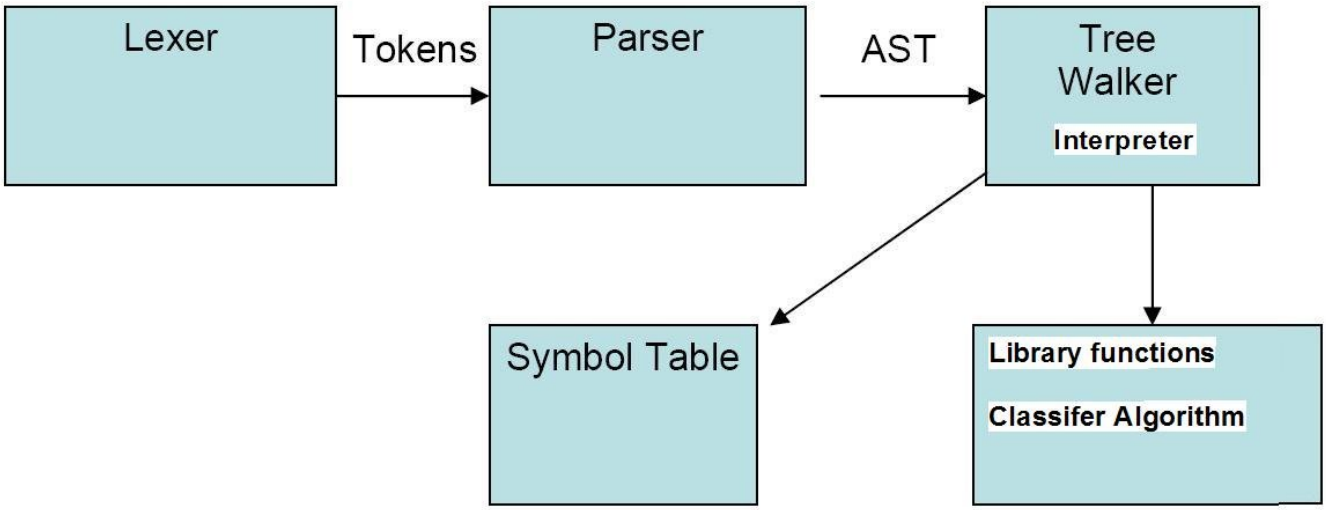
	Reuters-21578 (R52)	20Newsgroup
Training Set (T_r)	6532	11293
Test Set (T_e)	2658	7528

Number of Documents in the dataset

	Winnow ($\tau = \|T\|, \alpha=2.0, \beta=0.5$))		**Winnow** ($\tau = \|T\|/2, \alpha=2.0, \beta=0.5$)	
	Classified Correctly	Classified Incorrectly	Classified Correctly	Classified Incorrectly
Reuters-21578	2136	432	**2143 (80.62%)**	515
20Newsgroup	4369	3158	**5565 (73.92%)**	1962

Test Results

7 aTCL Language Architecture



The source program is passed to the lexical analyzer which produces tokens and passes them to the parser. The parser generates and Abstract Syntax Tree AST with root nodes controlled thorugh **"^"** operator in the parser grammar. Also, some other useless information is also being removed from the AST using **"!"**. This AST is then passed to the TreeWalker which uses AtclInterpreter to execute the program. A SymbolTable class is being written to handle all the variable declarations in the program. AtclInterpreter also maintains a symbol table for library functions. All library fuinctions are loaded at the time interpreter is instantiated and loaded into funcTable using Java reflection api. Adding a new library function now becomes a trivial issue as just new implementation in AtclFunctionutl needs to be provided without changing any grammar as some people have hardcoded library function names in the grammar. Also, every library function takes List as arguments which would then cover for optional arguments based on the logic in the function implementation. After, corpus is loaded, it then cleaned with **"+"** or **"-"** operators applied on the CorpusType and then inductive classifier is built using Winnow algorithm and evaluated.

8 aTCl Langauge Test Suite

Throughout the iterative development of the project, different aTCl program were used to test the lexer and parser. ErrorCodes have been defined for most specific type of errors.

```
import antlr.Token;
import antlr.TokenStreamException;
import antlr.RecognitionException;
import antlr.collections.AST;

import java.io.Reader;
import java.io.FileReader;
import java.io.FileNotFoundException;

/**
 * Created by IntelliJ IDEA.
 * User: JSK
 * Date: Nov 26, 2006
 * Time: 1:56:34 AM
 * To change this template use File | Settings | File Templates.
 */
```

```
public class AtclMain
{
   public static final String test1 = "programs/test5.atcl";

   public static void main(String[]args)
   {
      printLexerTokens(test1);
      printAST(test1);
   }

   public static void printLexerTokens(String file)
   {
      Token token;
      try
      {
         Reader reader = new FileReader(file);
         aTClLexer lexer = new aTClLexer(reader);
         while(true)
         {
            token = lexer.nextToken();
            if(token.getType() == Token.EOF_TYPE)
            {
               break;
            }

            System.out.println("Token: " + token.getText() + "");
         }
      }
      catch (TokenStreamException e)
      {
         System.err.println(e.toString());
      }
      catch(FileNotFoundException fnfe)
      {
         System.err.println(fnfe.toString());
      }
   }

   public static void printAST(String file)
   {
      try
      {
         Reader reader = new FileReader(file);
         aTClLexer lexer = new aTClLexer(reader);
         aTClParser parser = new aTClParser(lexer);
         parser.program();
         AST tree = parser.getAST();

         System.out.println(tree.toStringTree());
      }
      catch (RecognitionException e)
      {
         System.err.println(e.toString());
      }
      catch (TokenStreamException e)
      {
```

```
            System.err.println(e.toString());
        }
        catch(FileNotFoundException fnfe)
        {
            System.err.println(fnfe.toString());
        }
    }
}

import com.atcl.types.corpus.CorpusType;
import com.atcl.parser.DefaultCorpusParser;


/**
 * Created by IntelliJ IDEA.
 * User: JSK
 * Date: Dec 2, 2006
 * Time: 9:10:14 PM
 * To change this template use File | Settings | File Templates.
 */
public class ParserMain
{

    private static String corpusLocation = "C:\\Documents and Settings\\Administrator\\My
Documents\\Education\\Columbia\\Courses\\Fall 2006\\COMS W4252\\Ass\\Final
Project\\reuters_21578";
    private static String testSet = "r52-test-stemmed.txt";
    private static String trainingSet = "r52-train-stemmed.txt";

    private static String corpusLocationNg = "C:\\Documents and Settings\\Administrator\\My
Documents\\Education\\Columbia\\Courses\\Fall 2006\\COMS W4252\\Ass\\Final
Project\\20_Newsgroup";
    private static String testSetNg = "20ng-test-stemmed.txt";
    private static String trainingSetNg = "20ng-train-stemmed.txt";


    public static void main(String[] args)
    {
        if (args.length != 3)
        {
            System.out.println("Please provide three aruments as follows:");
            System.out.println("Argument 1: Directory Location of the Corpus location such as
c:\\reuters_21578");
            System.out.println("Argument 2: Test set file name such as r52-test-stemmed.txt");
            System.out.println("Argument 3: Train set file name such as r52-train-stemmed.txt");
            return ;
        }
        corpusLocation = args[0];
        testSet = args[1];
        trainingSet = args[2];

        CorpusType trainingCorpus = new CorpusType();
        CorpusType testCorpus = new CorpusType();

        DefaultCorpusParser parser = new DefaultCorpusParser();
```

```
        parser.parseCorpus(trainingCorpus,trainingSet);
        WinnowCorpusClassifier.buildClassifier(trainingCorpus,2.0f,0.5f);

        parser.parseCorpus(testCorpus,testSet);
        WinnowCorpusClassifier.testClassifier(trainingCorpus,testCorpus);

    }
}
```

9 aTCL Project Plan

9.1 Iterative Development

Project was 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
- Addition of Classifier module

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

9.2 Development Environment

This project was developed on Windows XP 64 bit platform using JRockit 5.0 VM. Antlr was used for generating Lexical Analyzer, Parser and Tree/Walker. Subversion with Tortoise SVN was used for version control.

9.3 Programming Style Guidelines

9.3.1 ANTLR Style guidelines

Tabs were used for indentation. Token names were written in all caps, with words separated with underscore and the rule names written in all small letters.

For and ANTLR rule with multiple alternatives, the first line contans the left hand side non-terminal symbol by itself, the next line should be followed by a colon (:) followed by the first alternative, and each subsequent line should align with a vertical bar (|) followed by the next alternative. All lines excluding first should be indented by the tab.

All actions are also listed immediately beneath the alternative with which they are associated, indeted with a tab. If the action contains more than one line of code then java conventions are followed with open and close braces on each line and stamtents indented inside them.

9.3.2 Java Style guidelines

Class names begin with Capital letters, method and local variable begins with lowercase letters. If a name is composed of multiple words, each word after the first begins with a capital letter. Comments were added to the code to explain the algorithm or to give the general idea about the functionality. Wherever, possible meaningful names for variables and methods were used.

9.3.3 aTCl Style guidelines

Varaible names begin with lowercase letters followed by the capital letter in each subsequent words in the variable names. Lines should be properly indented using tabs or spaces.

9.4 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

9.5 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

9.6 Project Log

September 26, 2006	Language White Paper.
October 19, 2006	Language Reference Manual.
October 25, 2006	Language front-end developed.
November 10, 2006	Utility functions related to corpus parsing were written.
November 15, 2006	Work on language back-end was started.
November 20,2006	Classifier algorithm was developed and testetd.
November 30,2006	Code refactoring was performed and some factory classes were generated.
December 10,2006	Tree walker and interpreter was written.
December 19, 2006	Final report submitted

10 aTCL Lessons Learned

ANTLR proves much needed help in realizng my idea to build a language for text categorization process. I started with thinking to implement only the document pre-processing module but end up implementing the complete text categorization system with fairly good results for unseen data. I should have started focusing on it earlier which would definitely give me more time to work on the project. Overall, it was a good experience and knowledge gained as to how compiler works.

11 Appendix

11.1 grammar.g

////////////////// aTCl LEXER //////////////////

```
class aTClLexer extends Lexer;
options
{
   k=2;
   exportVocab=aTCl;
   testLiterals=false;
   charVocabulary='\3' ..'\377';
}

tokens
{
        DECIMAL;
        IF              =  "if";
        ITERATE         =  "iterate";
        FOR             =  "for";
        TO              =  "to";
        RETURN          =  "return";
        CORPUS          =  "Corpus";
        SET             =  "Set";
        MAP             =  "Map";
        TRAINING        =  "Training";
        TS              =  "TS";
        DF              =  "DF";
        CATEGORY        =  ""
}

LPAREN          :  '(';
RPAREN          :  ')';
LBRACE          :  '{';
RBRACE          :  '}';
LBRACKET        :  '[';
RBRACKET        :  ']';
PERIOD          :  '.';
COMMA           :  ',';
ASGN            :  '=';
LT              :  '<';
LEQ             :  "<=";
GT              :  '>';
GEQ             :  ">=";
EQ              :  "==";
NEQ             :  "!=";
PLUS            :  '+';
MINUS           :  '-';
MOD             :  '%';
DIV             :  '/';
MULT            :  '*';
SEMI            :  ';';
COLON           :  ':';
AND             :  '&';
```

```
OR                  :   '|';
NOT                 :   '!';

WHITE_SPACE     :   (' ' | '\t')+
                        { $setType(Token.SKIP); }
                    ;

NEW_LINE            :  ('\n' | ('\r' '\n') => '\r' '\n' | '\r')
                        { $setType(Token.SKIP); newline();}
                    ;

COMMENT             :  "//" (~('\n' | '\r'))*
                        { $setType(Token.SKIP); }
                    ;
protected
DIGIT               :  '0' .. '9';

protected
ALPHA               :   ('a' .. 'z' | 'A' .. 'Z' | '_');

ID options  {testLiterals=true;}
                    :   ALPHA (ALPHA|DIGIT)*
                    ;

/* INTEGER example:
   1,1.0,0.1
*/
INTEGER             :  (DIGIT)+ (PERIOD {$setType(DECIMAL);} (DIGIT)+)? ;

STRING              :  '"'! ( ~('"' | '\n') | ('"'! '"'))* '"'! ;


////////////////// aTCl PARSER /////////////////
class aTClParser extends Parser;
options
{
        k                       = 2;
        buildAST                = true;
        exportVocab         = aTCl;
}

tokens
{
        ROOT;
        DECL;
        ASSGN;
        FUNC_CALL;
        ARG_LIST;
        UNARYOP_PLUS;
        UNARYOP_MINUS;
}

program             : (statement)* EOF!
                        {#program = #([ROOT,"PROG"],program); }
                    ;
```

```
statement       :   decl
                |   if_stmt {System.out.println("A new If statement");}
                |   iterate_stmt {System.out.println("A new Iterate statement");}
                |   func_call_stmt {System.out.println("A new Func Call statement");}
                |   assign_stmt
                |   unary_stmt
                ;

decl            :   types ID (COMMA! ID)* SEMI!
                    {#decl = #([DECL,"DECL"], decl);}
                ;

if_stmt         :  IF^ LPAREN! expression RPAREN! LBRACE! (statement)* RBRACE!
                    (options {greedy = true;}: ELSE^ LBRACE! (statement)* RBRACE!)?
                ;

iterate_stmt    : ITERATE^ LPAREN! ID COMMA! ("C"|"T"|"D") RPAREN!
                  LBRACE! (statement)* RBRACE!
                ;

func_call_stmt  : func_call SEMI!
                ;

func_call       : ID LPAREN! (arg_list)? RPAREN!
                  {#func_call = #([FUNC_CALL,"FUNC_CALL"], func_call);}
                ;

arg_list        :  expression (COMMA! expression)*
                {#arg_list = #([ARG_LIST,"ARG_LIST"],arg_list);}
                ;

assign_stmt     :  l_value ASGN^ expression SEMI!
                ;

unary_stmt      : PLUS! r_value SEMI!
                  {#unary_stmt = #([UNARYOP_PLUS,"UNARYOP_PLUS"],unary_stmt);}
                |   MINUS!  r_value SEMI!
                  {#unary_stmt = #([UNARYOP_MINUS,"UNARYOP_MINUS"],unary_stmt);}
                ;

types           : CORPUS | SET | MAP
                ;

expression      : expr (OR^ expr)*
                ;

expr            : rel_expr (AND^ rel_expr)*
                ;

rel_expr        : arith_expr ((GEQ^ | LEQ^ | GT^ | LT^ | EQ^ | NEQ^) arith_expr)?
                ;

arith_expr      : arith_term ((PLUS^ | MINUS^) arith_term)*
                ;

arith_term      : func_call
```

```
                | ID PERIOD! ("C"|"T"|"D")
                |   ID
                |   f
                |   t
                |   STRING
                |   INTEGER
                |   DECIMAL
                ;

r_value         :   ID
                ;

l_value         :   ID
                ;

t               : "true"
                |   "TRUE"
                |   "1"
                ;

f               :   "false"
                |   "FALSE"
                |   "0"
                ;
```

11.2 walker.g

```
header
{
        import java.util.List;
        import java.util.ArrayList;
        import com.atcl.types.*;
        import com.atcl.util.*;
}

class aTClTreeWalker extends TreeParser;
options
{
        importVocab         = aTCl;
}
{
        AtclInterpreter ipt;
        public aTClTreeWalker(AtclInterpreter ipt)
        {
                this();
                this.ipt = ipt;
           }
}

program         : #(ROOT (statement)*)
                ;

statement       : decl
                | unaryop
                | binaryop
                | func_call
```

```
                        | assign
                        ;

decl                    : #(decl:DECL type:CORPUS (var_name:ID
                        {ipt.createNewVaraible(type.getText(),var_name.getText());})+)
                        ;

unaryop
{
        AtclType a = null;
}

                        : #(UNARYOP_PLUS a=expr)        {ipt.unaryOp(a,"+");}
                        | #(UNARYOP_MINUS a=expr)       {ipt.unaryOp(a,"-");}
                                ;

binaryop returns [AtclType ret]
{
        AtclType a,b = null;
        ret = null;
}
                        : #(PLUS a=expr b=expr) { ret = ipt.binaryOp(a,b,"+");}
                        ;

func_call returns [AtclType ret]
{
        ret = null;
        AtclType a,b = null;
        List <AtclType> arguments = new ArrayList();
}
                        : #(funcCall:FUNC_CALL
                        {
                                // First child is always the FuncName
                                   AST node = funcCall.getFirstChild();
                                String funcName = node.getText();
                                a=ipt.getFuncByName(funcName);
                                // Next child is ARGS_LIST
                                node = node.getNextSibling();
                                arguments=arg_list(node);
                                ret = ipt.invokeFunction(a,arguments);
                        })
                        ;

arg_list returns [List <AtclType> ret]
{
        ret = new ArrayList<AtclType>();
        AtclType arg;
}
                        : #(args:ARG_LIST (arg=expr {ret.add(arg);})*)
                        ;

assign
{
        AtclType a,b,c = null;
}
                        : #(ASGN a=expr ((b=func_call {ipt.assignVariable(a,b);})
```

```
                        | #(PLUS b=expr c=expr
                        {
                        // This will update the mapping of a with the new type returned from the func_call
                                ipt.assignVariable(a,ipt.binaryOp(b,c,"+"));
                        }))
                        ;

expr  returns [AtclType ret]
{
        ret = null;
        AtclType a;
}
                        : "true"      {ret = new AtclBoolType(true);}
                        | "false"     {ret = new AtclBoolType(false);}
                        | name:ID     {ret = ipt.getVarByName(name.getText());}
                        | str:STRING  {ret = ipt.createString(str.getText());}
                        | num:NUMBER  {ret = ipt.createNumber(num.getText());}
                        | dec:DECIMAL {ret= ipt.createDecimal(dec.getText());}
                        ;
```

11.3 Framework objects

```
package com.atcl;

import com.atcl.types.AtclType;
import com.atcl.util.AtclException;

public class AtclObject
{
   public static boolean DEBUG = true;

   public void error(String errorCondition, AtclType atclType)
   {
      String error = "ERROR: " + errorCondition +  (atclType == null? " [Type: null]"
                        : " [Type: " + atclType.getName() + " @Value: " + atclType.toString() + "]");
      throw new AtclException(error);
   }

   public void debug(String message)
   {
      System.out.println(message);
   }

   public static void setDebugOn(boolean DEBUG)
   {
      AtclObject.DEBUG = DEBUG;
   }
}

package com.atcl.util;
public class ErrorCodes
{
        public static final String ATCL_ERR_CREATE_CORPUS_INVALID_ARGUMENTS =
        "ATCL_ERR_CREATE_CORPUS_INVALID_ARGUMENTS";
```

```
        public static final String ATCL_ERR_BUILD_CLASSIFIER_INVALID_ARGUMENTS =
        "ATCL_ERR_BUILD_CLASSIFIER_INVALID_ARGUMENTS";

        public static final String ATCL_ERR_TEST_CLASSIFIER_INVALID_ARGUMENTS =
        "ATCL_ERR_TEST_CLASSIFIER_INVALID_ARGUMENTS";

        public static final String ATCL_ERR_SAVE_CORPUS_INVALID_ARGUMENTS =
        "ATCL_ERR_SAVE_CORPUS_INVALID_ARGUMENTS";

        public static final String ATCL_ERR_DISPLAY_INVALID_ARGUMENTS =
        "ATCL_ERR_DISPLAY_INVALID_ARGUMENTS";


        public static final String ATCL_ERR_VAR_ALREADY_DEFINED =
        "ATCL_ERR_VAR_ALREADY_DEFINED";

        public static final String ATCL_VARAIBLE_NOT_INITIALIZED =
        "ATCL_VARAIBLE_NOT_INITIALIZED";

        public static final String ATCL_ASSGN_LEFT_VARAIBLE_NOT_INITIALIZED =
        "ATCL_ASSGN_LEFT_VARAIBLE_NOT_INITIALIZED";

        public static final String ATCL_ASSGN_RIGHT_VARAIBLE_NOT_INITIALIZED =
        "ATCL_ASSGN_RIGHT_VARAIBLE_NOT_INITIALIZED";

        public static final String ATLC_VARIABLE_NOT_DECLARED =
        "ATLC_VARIABLE_NOT_DECLARED";

        public static final String ATLC_FUNCTION_NOT_DECLARED =
        "ATLC_FUNCTION_NOT_DECLARED";

        public static final String ATCL_OPERATION_NOT_SUPPORTED =
        "ATCL_OPERATION_NOT_SUPPORTED";

        public static final String ATCL_NOT_VALID_CORPUS_TYPE =
        "ATCL_NOT_VALID_CORPUS_TYPE";

        public static final String ATCL_NOT_MATCHING_PARSER_FOUND_FOR_CORPUS =
        "ATCL_NOT_MATCHING_PARSER_FOUND_FOR_CORPUS";

        public static final String ATCL_NOT_VALID_CLASSIFIER_FOUND =
        "ATCL_NOT_VALID_CLASSIFIER_FOUND";
}

package com.atcl.util;
/**
 * Created by IntelliJ IDEA.
 * User: JSK
 * Date: Dec 16, 2006
 * Time: 8:08:59 PM
 * To change this template use File | Settings | File Templates.
 */
public interface AtclConstants
{
    // Library Functions
    public static final String CREATE_CORPUS = "createCorpus";
```

```
    public static final String CREATE_SET = "createSet";
    public static final String DIMENSIONALITY_REDUCTION = "dimensionalityReduction";
    public static final String BUILD_CLASSIFIER = "buildClassifier";
    public static final String TEST_CLASSIFIER = "testClassifier";
    public static final String EVALUATE_CLASSIFIERS = "evaluateClassifiers";
    public static final String SAVE_CORPUS = "saveCorpus";
    public static final String DISPLAY = "display";


    // aTCl Types
    public static final String STRING_TYPE = "String";
    public static final String CORPUS_TYPE = "Corpus";
    public static final String LIST_TYPE = "List";
    public static final String MAP_TYPE = "Map";
    public static final String BOOL_TYPE = "Bool";
    public static final String INTEGER_TYPE = "Integer";
    public static final String DECIMAL_TYPE = "Decimal";
    public static final String VOID_TYPE = "Void";

    //Corpus Types
    public static final String REUTERS_21578_CORPUS = "REUTERS_21578";
    public static final String NEWSGROUP_20_CORPUS = "NEWSGROUP_20";

    //Corpus split
    public static final String CORPUS_TRAINING_SPLIT = "Training";
    public static final String CORPUS_VALIDATION_SPLIT = "Validation";
    public static final String CORPUS_TEST_SPLIT = "Test";

    public static final String PLUS = "+";
    public static final String MINUS = "-";

    public static final String DOCUMENT = "Document";
    public static final String TERM = "Term";
    public static final String CATEGORY = "Category";

    // Classifier Names
    public static final String WINNOW = "Winnow";
    public static final String PERCEPTRON = "Perceptron";

    public final String trainFiler52 = "r52-train-stemmed.txt";
    public final String testFiler52 = "r52-test-stemmed.txt";

    public String testFileng = "20ng-test-stemmed.txt";
    public String trainingFileng = "20ng-train-stemmed.txt";

    // ADD GET BY NAME METHOD HERE.
}

package com.atcl.util;

/**
 * Created by IntelliJ IDEA.
 * User: JSK
 * Date: Dec 16, 2006
 * Time: 8:26:40 PM
 * To change this template use File | Settings | File Templates.
```

```
 */
public class AtclException extends RuntimeException
{
    public AtclException(String message)
    {
        super(message);
    }
}

package com.atcl.util;
import com.atcl.types.AtclType;
import java.util.HashMap;

/**
 * Created by IntelliJ IDEA.
 * User: JSK
 * Date: Dec 2, 2006
 * Time: 5:15:22 AM
 * To change this template use File | Settings | File Templates.
 */
public class AtclSymbolTable extends HashMap<String, AtclType>
{
    public static final String serialVersionUID = "__AtclSymbolTable__";


    public AtclSymbolTable()
    {
    }

    /*
    * This method returns the boolean value indicating if the variable with varName already exists in the
symbol table.
    * */
    public boolean containsVariable(String varName)
    {
        return containsKey(varName);
    }

    /*
    * This method returns the value of the variable with varName from the symbol table.
    * */
    public AtclType getValue(String varName)
    {
        return this.get(varName);

    }

    /*
    * This method add new variable in the symbol table. Key is the variable name and value is its data-
type.
    * */
    public void setValue(String varName, AtclType dataType)
    {
        this.put(varName,dataType);
    }
```

```
    /*
    * This method removes variable from the symbol table.
    * */
    public void removeValue(String varName)
    {
        this.remove(varName);
    }

}
```

11.4 Library function Utility

```
package com.atcl.util;

import com.atcl.types.corpus.CorpusType;
import com.atcl.types.AtclStringType;
import com.atcl.types.AtclDecimalType;
import com.atcl.AtclObject;
import com.atcl.classifier.BaseCorpusClassifier;
import com.atcl.classifier.CorpusClassifierFactory;
import com.atcl.parser.DefaultCorpusParser;

import java.util.*;
import java.io.FileWriter;
import java.io.BufferedWriter;
import java.io.IOException;

/**
 * Created by IntelliJ IDEA.
 * User: JSK
 * Date: Dec 16, 2006
 * Time: 8:21:51 PM
 * To change this template use File | Settings | File Templates.
 */
public class AtclFunctionUtil extends AtclObject
{
    private static AtclFunctionUtil myInstance = new AtclFunctionUtil();

    private AtclFunctionUtil()
    {
        super();
    }

    public static AtclFunctionUtil getInstance()
    {
        return myInstance;
    }

    public CorpusType createCorpus (List args) throws AtclException
    {
        if(DEBUG)
            debug("In createCorpus");

        if ((args == null) || (args.size() < 3))
            error(ErrorCodes.ATCL_ERR_CREATE_CORPUS_INVALID_ARGUMENTS,null);
```

```
    CorpusType corpus = new CorpusType();
    DefaultCorpusParser corpusParser = new DefaultCorpusParser();
    String split,corpusType;

    corpusType = ((AtclStringType)args.get(1)).toString();
    split = ((AtclStringType)args.get(2)).toString();

    if(AtclConstants.REUTERS_21578_CORPUS.equals(corpusType))
          corpusParser.loadDocuments(corpus,((AtclStringType)args.get(0)).toString(),split,
          AtclConstants.trainFiler52,AtclConstants.testFiler52);
    else
       corpusParser.loadDocuments(corpus,((AtclStringType)args.get(0)).toString(),split,
          AtclConstants.trainingFileng,AtclConstants.testFileng);

    corpus.initialized = true;
    return corpus;
}

public void buildClassifier(List args)
{
    if(DEBUG)
       debug("In buildClassifier");

    if ((args == null) || (args.size() < 4))
       error(ErrorCodes.ATCL_ERR_BUILD_CLASSIFIER_INVALID_ARGUMENTS,null);

   BaseCorpusClassifier corpusClassifier =
   CorpusClassifierFactory.getClassifier(((AtclStringType)args.get(1)).toString());

    if(corpusClassifier == null)
       error(ErrorCodes.ATCL_NOT_VALID_CLASSIFIER_FOUND,null);

    corpusClassifier.buildClassifier((CorpusType)args.get(0),
          ((AtclDecimalType)args.get(2)).decimalValue(),
          ((AtclDecimalType)args.get(3)).decimalValue());
}

public void testClassifier(List args)
{
    if (DEBUG)
       debug("In testClassifier");

    if ((args == null) || (args.size() < 3))
       error(ErrorCodes.ATCL_ERR_BUILD_CLASSIFIER_INVALID_ARGUMENTS,null);

   BaseCorpusClassifier corpusClassifier =
   CorpusClassifierFactory.getClassifier(((AtclStringType)args.get(2)).toString());

    if(corpusClassifier == null)
       error(ErrorCodes.ATCL_NOT_VALID_CLASSIFIER_FOUND,null);

    corpusClassifier.testClassifier((CorpusType)args.get(0),(CorpusType)args.get(1));
}

public void display(List args)
{
```

```
    if(DEBUG)
        debug("In display");

    if ((args == null) || (args.size() < 1))
            error(ErrorCodes.ATCL_ERR_DISPLAY_INVALID_ARGUMENTS,null);

    String operation;


}
```

```
public void saveCorpus(List args)
   {
      if(DEBUG)
         debug("In saveCorpus");

      if(DEBUG)
         debug("In buildClassifier");

      if ((args == null) || (args.size() < 4))
         error(ErrorCodes.ATCL_ERR_SAVE_CORPUS_INVALID_ARGUMENTS,null);

      CorpusType corpus = (CorpusType)args.get(0);
      String row = ((AtclStringType)args.get(1)).toString();
      String column = ((AtclStringType)args.get(2)).toString();

      String filePath = ((AtclStringType)args.get(3)).toString();
      StringBuffer strBuffer = new StringBuffer();

      Iterator iter = null;
      Map.Entry entry = null;
      ArrayList mappingList = null;
      HashMap map = null;

      if (row.equals(AtclConstants.CATEGORY))
      {
         iter = corpus.getCategories().getCategoryMapping().entrySet().iterator();

         while(iter.hasNext())
         {
            entry = (Map.Entry)iter.next();
            mappingList = (ArrayList)entry.getValue();

            strBuffer.append(entry.getKey());
            strBuffer.append("\t");

            if(column.equals(AtclConstants.TERM))
            {
               map = (HashMap)mappingList.get(1);
               strBuffer.append(map.toString());
            }
            else if (column.equals(AtclConstants.DOCUMENT))
            {
               map = (HashMap)mappingList.get(0);
               strBuffer.append(map.toString());
            }
            strBuffer.append("\n");
         }
      }
      else if (row.equals(AtclConstants.DOCUMENT))
      {
         iter = corpus.getDocuments().getDocumentMapping().entrySet().iterator();

         while(iter.hasNext())
         {
            entry = (Map.Entry)iter.next();
            mappingList = (ArrayList)entry.getValue();
```

```
            strBuffer.append(entry.getKey());
            strBuffer.append("\t");

            if(column.equals(AtclConstants.TERM))
            {
               map = (HashMap)mappingList.get(1);
               strBuffer.append(map.toString());
            }
            else if (column.equals(AtclConstants.CATEGORY))
            {
               map = (HashMap)mappingList.get(0);
               strBuffer.append(map.toString());
            }
            strBuffer.append("\n");
         }
      }
      else if (row.equals(AtclConstants.TERM))
      {
         iter = corpus.getTerms().getTermMapping().entrySet().iterator();

         while(iter.hasNext())
         {
            entry = (Map.Entry)iter.next();
            mappingList = (ArrayList)entry.getValue();
            strBuffer.append(entry.getKey());
            strBuffer.append("\t");
            if(column.equals(AtclConstants.DOCUMENT))
            {
               map = (HashMap)mappingList.get(1);
               strBuffer.append(map.toString());
            }
            else if (column.equals(AtclConstants.CATEGORY))
            {
               map = (HashMap)mappingList.get(0);
               strBuffer.append(map.toString());
            }
            strBuffer.append("\n");
         }
      }

      try
      {
         BufferedWriter out = new BufferedWriter(new FileWriter(filePath));
         out.write(strBuffer.toString());
         out.close();
      }
      catch (IOException e)
      {
         e.printStackTrace();
      }
   }
}
```

11.5 aTCl Interpreter

```
package com.atcl.util;

import com.atcl.types.*;
import com.atcl.types.corpus.CorpusType;
import com.atcl.AtclObject;

import java.util.List;
import java.lang.reflect.Method;

/**
 * Created by IntelliJ IDEA.
 * User: JSK
 * Date: Dec 2, 2006
 * Time: 6:50:46 AM
 * To change this template use File | Settings | File Templates.
 */
public class AtclInterpreter extends AtclObject
{
    private AtclSymbolTable symbTable;
    private AtclSymbolTable funcTable;

    public static long docId = 100; // This would help in doing binary plus operator on the Corpus.
DocumentIds won't conflict now while combining documents.

    public AtclInterpreter()
    {
        symbTable = new AtclSymbolTable();
        funcTable = new AtclSymbolTable();
        loadFuncTable();
    }

    public AtclType createNewVaraible(String type,String variable)
    {
        if(DEBUG)
            debug("In createNewVariable");

        AtclType ret = null;

        if (symbTable.getValue(variable) != null)
            error(ErrorCodes.ATCL_ERR_VAR_ALREADY_DEFINED,null);

        CorpusType corpusType = new CorpusType();

        corpusType.setName(variable);
        symbTable.setValue(variable,corpusType);
        return ret;
    }
```

```
public AtclType assignVariable(AtclType lVariable, AtclType rVariable)
   {
      if(DEBUG)
         debug("In assignVariable");

      if(lVariable == null)
         error(ErrorCodes.ATCL_ASSGN_LEFT_VARAIBLE_NOT_INITIALIZED,lVariable);

      else if (rVariable == null)
         error(ErrorCodes.ATCL_ASSGN_RIGHT_VARAIBLE_NOT_INITIALIZED,rVariable);

      rVariable.setName(lVariable.getName());
      //lVariable.assign(rVariable);

      symbTable.setValue(lVariable.getName(),rVariable);
      return rVariable;
   }

   public AtclType invokeFunction(AtclType func, List<AtclType> argList)
   {
      // func is of Type AtclFunctionType
      if(DEBUG)
         debug("In invokeFunction");

      AtclFunctionType callingFunc = (AtclFunctionType)funcTable.getValue(func.getName());

      if( callingFunc != null)
         return callingFunc.invoke(argList);

      return null;
   }

   public AtclType getVarByName(String varName)
   {
      AtclType ret = (AtclType)symbTable.getValue(varName);
      if(ret == null)
         error(ErrorCodes.ATLC_VARIABLE_NOT_DECLARED,ret);

      return ret;
   }

   public AtclType getFuncByName(String funcName)
   {
      AtclType ret = (AtclType)funcTable.getValue(funcName);

      if(ret == null)
         error(ErrorCodes.ATLC_FUNCTION_NOT_DECLARED + " FuncName:" + funcName,ret);

      return ret;
   }
```

```
public AtclType createString(String str)
{
   return new AtclStringType(str);
}

public AtclType createNumber(String number)
{
   return new AtclIntType(Integer.parseInt(number));
}

public AtclType createDecimal(String decimal)
{
   return new AtclDecimalType(Float.parseFloat(decimal));
}

public void unaryOp(AtclType variable,String operator)
{
   if(DEBUG)
      debug("In unaryOp");

   if(operator.equals(AtclConstants.PLUS))
      variable.unaryPlus();

   else if(operator.equals(AtclConstants.MINUS))
      variable.unaryMinus();
}

public AtclType binaryOp(AtclType a, AtclType b,String operator)
{
   if(DEBUG)
      debug("In binaryAdd");

   if(operator.equals(AtclConstants.PLUS))
      return a.binaryPlus(b);   //return ((CorpusType)a)._binaryPlus(b);

  error(ErrorCodes.ATCL_OPERATION_NOT_SUPPORTED,null);
   return null;
}

private void loadFuncTable()
{
   AtclFunctionUtil funcUtil = AtclFunctionUtil.getInstance();

   Class funcUtilClass = funcUtil.getClass();
   Class [] args = new Class[] {List.class};
   Method method = null;

   try
   {
      method = funcUtilClass.getMethod(AtclConstants.CREATE_CORPUS,args);
         funcTable.setValue(AtclConstants.CREATE_CORPUS, new
         AtclFunctionType(AtclConstants.CREATE_CORPUS,method,funcUtil));

      method = funcUtilClass.getMethod(AtclConstants.DIMENSIONALITY_REDUCTION,args);
         funcTable.setValue(AtclConstants.DIMENSIONALITY_REDUCTION, new
         AtclFunctionType(AtclConstants.DIMENSIONALITY_REDUCTION,method,funcUtil));
```

```
            method = funcUtilClass.getMethod(AtclConstants.BUILD_CLASSIFIER,args);
               funcTable.setValue(AtclConstants.BUILD_CLASSIFIER, new
               AtclFunctionType(AtclConstants.BUILD_CLASSIFIER,method,funcUtil));

            method = funcUtilClass.getMethod(AtclConstants.TEST_CLASSIFIER,args);
               funcTable.setValue(AtclConstants.TEST_CLASSIFIER, new
               AtclFunctionType(AtclConstants.TEST_CLASSIFIER,method,funcUtil));

            method = funcUtilClass.getMethod(AtclConstants.EVALUATE_CLASSIFIERS,args);
               funcTable.setValue(AtclConstants.EVALUATE_CLASSIFIERS, new
               AtclFunctionType(AtclConstants.EVALUATE_CLASSIFIERS,method,funcUtil));

            method = funcUtilClass.getMethod(AtclConstants.SAVE_CORPUS,args);
               funcTable.setValue(AtclConstants.SAVE_CORPUS, new
               AtclFunctionType(AtclConstants.SAVE_CORPUS,method,funcUtil));

            method = funcUtilClass.getMethod(AtclConstants.DISPLAY,args);
               funcTable.setValue(AtclConstants.DISPLAY, new
               AtclFunctionType(AtclConstants.DISPLAY,method,funcUtil));
         }
         catch (NoSuchMethodException e)
         {
            e.printStackTrace();
         }
      }
}
```

11.6 Corpus Parser

```
package com.atcl.parser;

import com.atcl.types.corpus.CorpusType;
import com.atcl.util.AtclConstants;
import com.atcl.util.AtclInterpreter;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.*;

/**
 * Created by IntelliJ IDEA.
 * User: JSK
 * Date: Dec 2, 2006
 * Time: 3:51:38 AM
 * To change this template use File | Settings | File Templates.
 */
public class DefaultCorpusParser
{


   public DefaultCorpusParser()
   {
      super();
```

```
    }

    public void loadDocuments(CorpusType corpus, String corpusLocation, String corpusSplit,String
trainFile,String testFile)
    {
        String docsFile = "";

        if(corpusSplit.equals(AtclConstants.CORPUS_TRAINING_SPLIT))
            docsFile = trainFile;

        else
            docsFile = testFile;

        try
        {
            BufferedReader in = new BufferedReader(new FileReader(corpusLocation + "\\" + docsFile));
            String str;

            while ((str = in.readLine()) != null)
            {
                //corpus.getCategories().addNewCategory(Category.encode(categType+str));
                //System.out.println(str);

                AtclInterpreter.docId += 1;
                loadDocument(corpus,AtclInterpreter.docId,str);
            }
            in.close();
        }
        catch (IOException e)
        {
            e.printStackTrace();
        }
    }

    private void loadDocument(CorpusType corpus, long docId, String doc)
    {
        Long documentId = new Long(docId);
        int i = 0;
        String term = null;
        String category = null;

        Float termDocWeight = 0.0f;
        Float  termCategWeight = 0.0f;
        Float documentWeight = 1.0f;
        Float categoryWeight = 1.0f;

        ArrayList categoryMappingList = null;
        ArrayList termMappingList = null;

        List documentMappingList  = new ArrayList();
        documentMappingList.add(new HashMap());     // Category Mapping
        documentMappingList.add(new HashMap());     // Terms Mapping

        StringTokenizer st = new StringTokenizer(doc);
        category = st.nextToken(); // always categroy is there
```

```
      ((HashMap)documentMappingList.get(0)).put(category, new Float(1.0));
      loadCategory(corpus,category, docId, documentWeight);

      while (st.hasMoreTokens())
      {
         term = st.nextToken();

         // These are all terms of the document.
         loadTerm(corpus,term,category,documentId,documentWeight,categoryWeight);

         loadTermCategory(corpus,term,category,documentId,documentWeight,categoryWeight);

         if (((HashMap)documentMappingList.get(1)).containsKey(term))
         {
            // This term occurs more than once in the document. Update its counter.

            termDocWeight = (Float)((HashMap)documentMappingList.get(1)).get(term);
            ((HashMap)documentMappingList.get(1)).put(term, new Float(termDocWeight.floatValue() +
1.0));
         }
         else
            ((HashMap)documentMappingList.get(1)).put(term, new Float(1.0));

         corpus.getDocuments().getDocumentMapping().put(docId,documentMappingList);
      }
   }

   private void loadCategory(CorpusType corpus, String category, Long documentId, Float
documentWeight)
   {
      Map mapZero,mapOne,mapTwo = null;
      ArrayList categoryMappingList = null;

      if ( ((HashMap)corpus.getCategories().getCategoryMapping()).containsKey(category))
      {
         // This category already exists. Add documentId mapping.

         categoryMappingList =
(ArrayList)((HashMap)corpus.getCategories().getCategoryMapping()).get(category);
         ((HashMap)categoryMappingList.get(0)).put(documentId,documentWeight);
      }
      else
      {
         categoryMappingList = new ArrayList();
         mapZero = new HashMap(); mapZero.put(documentId,documentWeight);
         mapOne = new HashMap();
         mapTwo = new HashMap();

         categoryMappingList.add(mapZero);
         categoryMappingList.add(mapOne);    // This is for terms mapping
         categoryMappingList.add(mapTwo);    // This is for category classifier

         ((HashMap)corpus.getCategories().getCategoryMapping()).put(category,categoryMappingList);
      }
   }
```

```
   private void loadTermCategory(CorpusType corpus, String term, String category, Long documentId,
Float documentWeight,
                        Float categoryWeight)
   {
      ArrayList categoryMappingList = null;

      categoryMappingList =
(ArrayList)((HashMap)corpus.getCategories().getCategoryMapping()).get(category);

      if(((HashMap)categoryMappingList.get(1)).containsKey(term))
      {
         // Add term mapping for category.
         Float termCategFrequency = (Float)((HashMap)categoryMappingList.get(1)).get(term);
         ((HashMap)categoryMappingList.get(1)).put(term,new Float(termCategFrequency.floatValue() +
1.0));
      }
      else
      {
         ((HashMap)categoryMappingList.get(1)).put(term, new Float(1.0));
      }
   }

   private void loadTerm(CorpusType corpus, String term, String category, Long documentId, Float
documentWeight,
                        Float categoryWeight)
   {
      ArrayList termMappingList = null;
      Map mapZero,mapOne = null;

      if ( ((HashMap)corpus.getTerms().getTermMapping()).containsKey(term))
      {
         // This term already exists. Add documentId and CategoryId mapping.
         termMappingList = (ArrayList)((HashMap)corpus.getTerms().getTermMapping()).get(term);

         if (((HashMap)termMappingList.get(0)).containsKey(documentId))
         {
             // This terms occurs more than once for the document.
            Float termDocFrequency = (Float)((HashMap)termMappingList.get(0)).get(documentId);
            ((HashMap)termMappingList.get(0)).put(documentId,new
Float(termDocFrequency.floatValue() + 1.0));
         }
         else
         {
            ((HashMap)termMappingList.get(0)).put(documentId,new Float(1.0));
         }

         if (((HashMap)termMappingList.get(1)).containsKey(category))
         {
            // This term occurs more than once for this category. Update its counter.
            Float termCategFrequency = (Float)((HashMap)termMappingList.get(1)).get(category);
            ((HashMap)termMappingList.get(1)).put(category,new Float(termCategFrequency.floatValue()
+ 1.0));
         }
         else
```

```
            {
                ((HashMap)termMappingList.get(1)).put(category, new Float(1.0));
            }
        }
        else
        {
            termMappingList = new ArrayList();
            mapZero = new HashMap(); mapZero.put(documentId,documentWeight);
            mapOne = new HashMap(); mapOne.put(category,categoryWeight);
            termMappingList.add(mapZero);
            termMappingList.add(mapOne);
            ((HashMap)corpus.getTerms().getTermMapping()).put(term,termMappingList);
        }
    }
}
```

11.7 Stemmer

```
package com.atcl.stemmer;

import com.atcl.AtclObject;

/**
 * Created by IntelliJ IDEA.
 * User: JSK
 * Date: Dec 2, 2006
 * Time: 3:53:05 AM
 * To change this template use File | Settings | File Templates.
 */
public class BaseCorpusStemmer extends AtclObject
{
}
package com.atcl.stemmer.porter;
import com.atcl.stemmer.BaseCorpusStemmer;
import java.io.*;

/**
 * Created by IntelliJ IDEA.
 * User: JSK
 * Date: Dec 2, 2006
 * Time: 3:54:23 AM
 * To change this template use File | Settings | File Templates.
 */

/**
 * Stemmer, implementing the Porter Stemming Algorithm
 * <p/>
 * The Stemmer class transforms a word into its root form.  The input
 * word can be provided a character at time (by calling add()), or at once
 * by calling one of the various stem(something) methods.
 * The original paper is in Porter, 1980, An algorithm for suffix stripping, Program, Vol. 14,
 * no. 3, pp 130-137, See also http://www.tartarus.org/~martin/PorterStemmer
 */

public class PorterCorpusStemmer extends BaseCorpusStemmer
```

```
{
   private char[] b;
   private int i,     /* offset into b */
           i_end, /* offset to end of stemmed word */
           j, k;

   private static final int INC = 50; /* unit of size whereby b is increased */

   public PorterCorpusStemmer()
   {
      b = new char[INC];
      i = 0;
      i_end = 0;
   }

   /**
    * Add a character to the word being stemmed.  When you are finished
    * adding characters, you can call stem(void) to stem the word.
    */

   public void add(char ch)
   {
      if (i == b.length)
      {
         char[] new_b = new char[i + INC];
         for (int c = 0; c < i; c++)
         {
            new_b[c] = b[c];
         }
         b = new_b;
      }
      b[i++] = ch;
   }


   /**
    * Adds wLen characters to the word being stemmed contained in a portion
    * of a char[] array. This is like repeated calls of add(char ch), but
    * faster.
    */

   public void add(char[] w, int wLen)
   {
      if (i + wLen >= b.length)
      {
         char[] new_b = new char[i + wLen + INC];
         for (int c = 0; c < i; c++)
         {
            new_b[c] = b[c];
         }
         b = new_b;
      }
      for (int c = 0; c < wLen; c++)
      {
         b[i++] = w[c];
      }
```

```
}

/**
 * Returns the length of the word resulting from the stemming process.
 */
public int getResultLength()
{
   return i_end;
}

/**
 * Returns a reference to a character buffer containing the results of
 * the stemming process.  You also need to consult getResultLength()
 * to determine the length of the result.
 */
public char[] getResultBuffer()
{
   return b;
}

/* cons(i) is true <=> b[i] is a consonant. */

private final boolean cons(int i)
{
   switch (b[i])
   {
      case'a':
      case'e':
      case'i':
      case'o':
      case'u':
         return false;
      case'y':
         return (i == 0) ? true : !cons(i - 1);
      default:
         return true;
   }
}

/* m() measures the number of consonant sequences between 0 and j. if c is
   a consonant sequence and v a vowel sequence, and <..> indicates arbitrary
   presence,

      <c><v>       gives 0
      <c>vc<v>     gives 1
      <c>vcvc<v>   gives 2
      <c>vcvcvc<v> gives 3
      ....
*/

private final int m()
{
   int n = 0;
   int i = 0;
   while (true)
   {
```

```
        if (i > j)
        {
            return n;
        }
        if (!cons(i))
        {
            break;
        }
        i++;
    }
    i++;
    while (true)
    {
        while (true)
        {
            if (i > j)
            {
                return n;
            }
            if (cons(i))
            {
                break;
            }
            i++;
        }
        i++;
        n++;
        while (true)
        {
            if (i > j)
            {
                return n;
            }
            if (!cons(i))
            {
                break;
            }
            i++;
        }
        i++;
    }
}

/* vowelinstem() is true <=> 0,...j contains a vowel */

private final boolean vowelinstem()
{
    int i;
    for (i = 0; i <= j; i++)
    {
        if (!cons(i))
        {
            return true;
        }
    }
    return false;
```

```
}

/* doublec(j) is true <=> j,(j-1) contain a double consonant. */

private final boolean doublec(int j)
{
   if (j < 1)
   {
      return false;
   }
   if (b[j] != b[j - 1])
   {
      return false;
   }
   return cons(j);
}

/* cvc(i) is true <=> i-2,i-1,i has the form consonant - vowel - consonant
   and also if the second c is not w,x or y. this is used when trying to
   restore an e at the end of a short word. e.g.

      cav(e), lov(e), hop(e), crim(e), but
      snow, box, tray.

*/

private final boolean cvc(int i)
{
   if (i < 2 || !cons(i) || cons(i - 1) || !cons(i - 2))
   {
      return false;
   }
   {
      int ch = b[i];
      if (ch == 'w' || ch == 'x' || ch == 'y')
      {
         return false;
      }
   }
   return true;
}

private final boolean ends(String s)
{
   int l = s.length();
   int o = k - l + 1;
   if (o < 0)
   {
      return false;
   }
   for (int i = 0; i < l; i++)
   {
      if (b[o + i] != s.charAt(i))
      {
         return false;
      }
```

```
    }
    j = k - l;
    return true;
}

 /* setto(s) sets (j+1),...k to the characters in the string s, readjusting
k. */

 private final void setto(String s)
{
    int l = s.length();
    int o = j + 1;
    for (int i = 0; i < l; i++)
    {
        b[o + i] = s.charAt(i);
    }
    k = j + l;
}

 /* r(s) is used further down. */

 private final void r(String s)
{
    if (m() > 0)
    {
        setto(s);
    }
}

 /* step1() gets rid of plurals and -ed or -ing. e.g.

       caresses  ->  caress
       ponies    ->  poni
       ties      ->  ti
       caress    ->  caress
       cats      ->  cat

       feed      ->  feed
       agreed    ->  agree
       disabled  ->  disable

       matting   ->  mat
       mating    ->  mate
       meeting   ->  meet
       milling   ->  mill
       messing   ->  mess

       meetings  ->  meet

*/

 private final void step1()
{
    if (b[k] == 's')
    {
        if (ends("sses"))
```

```
        {
            k -= 2;
        }
        else if (ends("ies"))
        {
            setto("i");
        }
        else if (b[k - 1] != 's')
        {
            k--;
        }
    }
    if (ends("eed"))
    {
        if (m() > 0)
        {
            k--;
        }
    }
    else if ((ends("ed") || ends("ing")) && vowelinstem())
    {
        k = j;
        if (ends("at"))
        {
            setto("ate");
        }
        else if (ends("bl"))
        {
            setto("ble");
        }
        else if (ends("iz"))
        {
            setto("ize");
        }
        else if (doublec(k))
        {
            k--;
            {
                int ch = b[k];
                if (ch == 'l' || ch == 's' || ch == 'z')
                {
                    k++;
                }
            }
        }
        else if (m() == 1 && cvc(k))
        {
            setto("e");
        }
    }
}

/* step2() turns terminal y to i when there is another vowel in the stem. */

private final void step2()
{
```

```
    if (ends("y") && vowelinstem())
    {
       b[k] = 'i';
    }
  }

  /* step3() maps double suffices to single ones. so -ization ( = -ize plus
-ation) maps to -ize etc. note that the string before the suffix must give
m() > 0. */

  private final void step3()
  {
    if (k == 0)
    {
       return; /* For Bug 1 */
    }
    switch (b[k - 1])
    {
       case'a':
          if (ends("ational"))
          {
             r("ate");
             break;
          }
          if (ends("tional"))
          {
             r("tion");
             break;
          }
          break;
       case'c':
          if (ends("enci"))
          {
             r("ence");
             break;
          }
          if (ends("anci"))
          {
             r("ance");
             break;
          }
          break;
       case'e':
          if (ends("izer"))
          {
             r("ize");
             break;
          }
          break;
       case'l':
          if (ends("bli"))
          {
             r("ble");
             break;
          }
          if (ends("alli"))
```

```
    {
        r("al");
        break;
    }
    if (ends("entli"))
    {
        r("ent");
        break;
    }
    if (ends("eli"))
    {
        r("e");
        break;
    }
    if (ends("ousli"))
    {
        r("ous");
        break;
    }
    break;
case'o':
    if (ends("ization"))
    {
        r("ize");
        break;
    }
    if (ends("ation"))
    {
        r("ate");
        break;
    }
    if (ends("ator"))
    {
        r("ate");
        break;
    }
    break;
case's':
    if (ends("alism"))
    {
        r("al");
        break;
    }
    if (ends("iveness"))
    {
        r("ive");
        break;
    }
    if (ends("fulness"))
    {
        r("ful");
        break;
    }
    if (ends("ousness"))
    {
        r("ous");
```

```
                break;
            }
            break;
        case't':
            if (ends("aliti"))
            {
                r("al");
                break;
            }
            if (ends("iviti"))
            {
                r("ive");
                break;
            }
            if (ends("biliti"))
            {
                r("ble");
                break;
            }
            break;
        case'g':
            if (ends("logi"))
            {
                r("log");
                break;
            }
    }
}

/* step4() deals with -ic-, -full, -ness etc. similar strategy to step3. */

private final void step4()
{
    switch (b[k])
    {
        case'e':
            if (ends("icate"))
            {
                r("ic");
                break;
            }
            if (ends("ative"))
            {
                r("");
                break;
            }
            if (ends("alize"))
            {
                r("al");
                break;
            }
            break;
        case'i':
            if (ends("iciti"))
            {
                r("ic");
```

```
                break;
            }
            break;
        case'l':
            if (ends("ical"))
            {
                r("ic");
                break;
            }
            if (ends("ful"))
            {
                r("");
                break;
            }
            break;
        case's':
            if (ends("ness"))
            {
                r("");
                break;
            }
            break;
    }
}

/* step5() takes off -ant, -ence etc., in context <c>vcvc<v>. */

private final void step5()
{
    if (k == 0)
    {
        return; /* for Bug 1 */
    }
    switch (b[k - 1])
    {
        case'a':
            if (ends("al"))
            {
                break;
            }
            return;
        case'c':
            if (ends("ance"))
            {
                break;
            }
            if (ends("ence"))
            {
                break;
            }
            return;
        case'e':
            if (ends("er"))
            {
                break;
            }
```

```
    return;
case'i':
    if (ends("ic"))
    {
        break;
    }
    return;
case'l':
    if (ends("able"))
    {
        break;
    }
    if (ends("ible"))
    {
        break;
    }
    return;
case'n':
    if (ends("ant"))
    {
        break;
    }
    if (ends("ement"))
    {
        break;
    }
    if (ends("ment"))
    {
        break;
    }
    /* element etc. not stripped before the m */
    if (ends("ent"))
    {
        break;
    }
    return;
case'o':
    if (ends("ion") && j >= 0 && (b[j] == 's' || b[j] == 't'))
    {
        break;
    }
    /* j >= 0 fixes Bug 2 */
    if (ends("ou"))
    {
        break;
    }
    return;
    /* takes care of -ous */
case's':
    if (ends("ism"))
    {
        break;
    }
    return;
case't':
    if (ends("ate"))
```

```
            {
                break;
            }
            if (ends("iti"))
            {
                break;
            }
            return;
        case'u':
            if (ends("ous"))
            {
                break;
            }
            return;
        case'v':
            if (ends("ive"))
            {
                break;
            }
            return;
        case'z':
            if (ends("ize"))
            {
                break;
            }
            return;
        default:
            return;
    }
    if (m() > 1)
    {
        k = j;
    }
}

/* step6() removes a final -e if m() > 1. */

private final void step6()
{
    j = k;
    if (b[k] == 'e')
    {
        int a = m();
        if (a > 1 || a == 1 && !cvc(k - 1))
        {
            k--;
        }
    }
    if (b[k] == 'l' && doublec(k) && m() > 1)
    {
        k--;
    }
}

/**
 * Stem the word placed into the Stemmer buffer through calls to add().
```

```
     * Returns true if the stemming process resulted in a word different
     * from the input.  You can retrieve the result with
     * getResultLength()/getResultBuffer() or toString().
     */
    public void stem()
    {
        k = i - 1;
        if (k > 1)
        {
            step1();
            step2();
            step3();
            step4();
            step5();
            step6();
        }
        i_end = k + 1;
        i = 0;
    }

    /**
     * After a word has been stemmed, it can be retrieved by toString(),
     * or a reference to the internal buffer can be retrieved by getResultBuffer
     * and getResultLength (which is generally more efficient.)
     */
    public String toString()
    {
        return new String(b, 0, i_end);
    }
}
```

11.8 aTCl Types

11.8.1 AtclType

```
package com.atcl.types;

import java.lang.reflect.Method;
import java.lang.reflect.InvocationTargetException;
import java.util.List;

/**
 * Created by IntelliJ IDEA.
 * User: JSK
 * Date: Dec 17, 2006
 * Time: 9:53:52 PM
 * To change this template use File | Settings | File Templates.
 */
public class AtclFunctionType extends AtclType
{
    private Method methodBody;
    private Object methodClass;

    public AtclFunctionType(String name)
    {
```

```
    super(name);
}

public AtclType _copy()
{
    return null;  //To change body of implemented methods use File | Settings | File Templates.
}

public AtclType _assign(AtclType newValue)
{
    return null;  //To change body of implemented methods use File | Settings | File Templates.
}

public AtclType _and(AtclType otherValue)
{
    return null;  //To change body of implemented methods use File | Settings | File Templates.
}

public AtclType _not(AtclType otherValue)
{
    return null;  //To change body of implemented methods use File | Settings | File Templates.
}

public AtclType _or(AtclType otherValue)
{
    return null;  //To change body of implemented methods use File | Settings | File Templates.
}

public void _unaryPlus()
{
    //To change body of implemented methods use File | Settings | File Templates.
}

public void _unaryMinus()
{
    //To change body of implemented methods use File | Settings | File Templates.
}

public AtclType _binaryPlus(AtclType otherValue)
{
    return null;  //To change body of implemented methods use File | Settings | File Templates.
}

public AtclFunctionType(Method methodBody, Object methodClass)
{
    this.methodBody = methodBody;
    this.methodClass = methodClass;
}

public AtclFunctionType(String name, Method methodBody, Object methodClass)
{
    super(name);
    this.methodBody = methodBody;
    this.methodClass = methodClass;
}
```

```
    public AtclFunctionType(String name, boolean initialized, Method methodBody, Object methodClass)
    {
        super(name, initialized);
        this.methodBody = methodBody;
        this.methodClass = methodClass;
    }

    public AtclType invoke(List<AtclType> args)
    {
        AtclType ret = null;
        try
        {
            ret = (AtclType)methodBody.invoke(methodClass,args);
        }
        catch (IllegalAccessException e)
        {
            e.printStackTrace();
        }
        catch (InvocationTargetException e)
        {
            e.printStackTrace();
        }
        return ret;
    }
}
```

11.8.2 AtclBoolType

```
package com.atcl.types;

import com.atcl.util.AtclConstants;

/**
 * Created by IntelliJ IDEA.
 * User: JSK
 * Date: Dec 17, 2006
 * Time: 3:51:42 AM
 * To change this template use File | Settings | File Templates.
 */
public class AtclBoolType extends AtclType
{
    private boolean booleanValue;

    public AtclBoolType(boolean booleanValue)
    {
        this.booleanValue = booleanValue;
    }

    public String getTypeName()
    {
       return AtclConstants.BOOL_TYPE;
    }

    public AtclType _assign(AtclType newValue)
    {
        return null;
```

```
    }

    public AtclType _copy()
    {
        return null;
    }

    public AtclType _and(AtclType otherValue)
    {
        return null;
    }

    public AtclType _or(AtclType otherValue)
    {
        return null;
    }

    public void _unaryPlus()
    {
        //To change body of implemented methods use File | Settings | File Templates.
    }

    public void _unaryMinus()
    {
        //To change body of implemented methods use File | Settings | File Templates.
    }

    public AtclType _binaryPlus(AtclType otherValue)
    {
        return null;  //To change body of implemented methods use File | Settings | File Templates.
    }

    public AtclType _not(AtclType otherValue)
    {
        return null;
    }

    public String toString()
    {
        return Boolean.toString(booleanValue);
    }
}
```

11.8.3 AtclIntType

```
package com.atcl.types;

import com.atcl.util.AtclConstants;

/**
 * Created by IntelliJ IDEA.
 * User: JSK
 * Date: Dec 17, 2006
 * Time: 7:18:06 PM
 * To change this template use File | Settings | File Templates.
 */
```

```
public class AtclIntType extends AtclType
{
   private int intValue;


   public AtclIntType(int intValue)
   {
      this.intValue = intValue;
   }

   public String getTypeName()
   {
     return AtclConstants.INTEGER_TYPE;
   }

   public AtclType _assign(AtclType newValue)
   {
      return null;
   }

   public AtclType _copy()
   {
      return null;
   }

   public AtclType _and(AtclType otherValue)
   {
      return null;
   }

   public AtclType _or(AtclType otherValue)
   {
      return null;
   }

   public void _unaryPlus()
   {
      //To change body of implemented methods use File | Settings | File Templates.
   }

   public void _unaryMinus()
   {
      //To change body of implemented methods use File | Settings | File Templates.
   }

   public AtclType _binaryPlus(AtclType otherValue)
   {
      return null;  //To change body of implemented methods use File | Settings | File Templates.
   }

   public AtclType _not(AtclType otherValue)
   {
      return null;
   }

   public String toString()
```

```
    {
        return Integer.toString(intValue);
    }
}
```

11.8.4 AtclFunctionType

```
package com.atcl.types;

import java.lang.reflect.Method;
import java.lang.reflect.InvocationTargetException;
import java.util.List;

/**
 * Created by IntelliJ IDEA.
 * User: JSK
 * Date: Dec 17, 2006
 * Time: 9:53:52 PM
 * To change this template use File | Settings | File Templates.
 */
public class AtclFunctionType extends AtclType
{
    private Method methodBody;
    private Object methodClass;

    public AtclFunctionType(String name)
    {
        super(name);
    }

    public AtclType _copy()
    {
        return null;  //To change body of implemented methods use File | Settings | File Templates.
    }

    public AtclType _assign(AtclType newValue)
    {
        return null;  //To change body of implemented methods use File | Settings | File Templates.
    }

    public AtclType _and(AtclType otherValue)
    {
        return null;  //To change body of implemented methods use File | Settings | File Templates.
    }

    public AtclType _not(AtclType otherValue)
    {
        return null;  //To change body of implemented methods use File | Settings | File Templates.
    }

    public AtclType _or(AtclType otherValue)
    {
        return null;  //To change body of implemented methods use File | Settings | File Templates.
    }

    public void _unaryPlus()
```

```
    {
        //To change body of implemented methods use File | Settings | File Templates.
    }

    public void _unaryMinus()
    {
        //To change body of implemented methods use File | Settings | File Templates.
    }

    public AtclType _binaryPlus(AtclType otherValue)
    {
        return null;  //To change body of implemented methods use File | Settings | File Templates.
    }

    public AtclFunctionType(Method methodBody, Object methodClass)
    {
        this.methodBody = methodBody;
        this.methodClass = methodClass;
    }

    public AtclFunctionType(String name, Method methodBody, Object methodClass)
    {
        super(name);
        this.methodBody = methodBody;
        this.methodClass = methodClass;
    }

    public AtclFunctionType(String name, boolean initialized, Method methodBody, Object methodClass)
    {
        super(name, initialized);
        this.methodBody = methodBody;
        this.methodClass = methodClass;
    }

    public AtclType invoke(List<AtclType> args)
    {
        AtclType ret = null;
        try
        {
            ret = (AtclType)methodBody.invoke(methodClass,args);
        }
        catch (IllegalAccessException e)
        {
            e.printStackTrace();
        }
        catch (InvocationTargetException e)
        {
            e.printStackTrace();
        }
        return ret;
    }
}
```

11.8.5 AtclDecimalType

```
package com.atcl.types;

import com.atcl.util.AtclConstants;

/**
 * Created by IntelliJ IDEA.
 * User: JSK
 * Date: Dec 17, 2006
 * Time: 7:20:56 PM
 * To change this template use File | Settings | File Templates.
 */
public class AtclDecimalType extends AtclType
{
    private float decimalValue;


    public AtclDecimalType(float decimalValue)
    {
        this.decimalValue = decimalValue;
    }

    public String getTypeName()
    {
       return AtclConstants.DECIMAL_TYPE;
    }

    public AtclType _assign(AtclType newValue)
    {
        return null;
    }

    public AtclType _copy()
    {
        return null;
    }

   public AtclType _and(AtclType otherValue)
    {
        return null;
    }

    public AtclType _or(AtclType otherValue)
    {
        return null;
    }

    public void _unaryPlus()
    {
        //To change body of implemented methods use File | Settings | File Templates.
    }

    public void _unaryMinus()
    {
        //To change body of implemented methods use File | Settings | File Templates.
    }
```

```
    public AtclType _binaryPlus(AtclType otherValue)
    {
        return null;  //To change body of implemented methods use File | Settings | File Templates.
    }

    public AtclType _not(AtclType otherValue)
    {
        return null;
    }

    public float decimalValue()
    {
        return decimalValue;
    }

    public String toString()
    {
        return Float.toString(decimalValue);
    }
}
```

11.8.6 AtclStringType

```
package com.atcl.types;

import com.atcl.util.AtclConstants;

/**
 * Created by IntelliJ IDEA.
 * User: JSK
 * Date: Dec 16, 2006
 * Time: 8:25:44 PM
 * To change this template use File | Settings | File Templates.
 */
public class AtclStringType extends AtclType
{
    private String strValue;

    public AtclStringType(String strValue)
    {
        this.strValue = strValue;
    }
    public String getTypeName()
    {
       return AtclConstants.STRING_TYPE;
    }

    public AtclType _assign(AtclType newValue)
    {
        return null;
    }

    public AtclType _copy()
    {
        return null;
    }
```

```
    public AtclType _and(AtclType otherValue)
    {
        return null;
    }

    public AtclType _or(AtclType otherValue)
    {
        return null;
    }

    public void _unaryPlus()
    {
        //To change body of implemented methods use File | Settings | File Templates.
    }

    public void _unaryMinus()
    {
        //To change body of implemented methods use File | Settings | File Templates.
    }

    public AtclType _binaryPlus(AtclType otherValue)
    {
        return null;  //To change body of implemented methods use File | Settings | File Templates.
    }

    public AtclType _not(AtclType otherValue)
    {
        return null;
    }

    public String toString()
    {
        return this.strValue;
    }
}
```

11.8.7 CorpusType

```
package com.atcl.types.corpus;

import com.atcl.types.AtclType;
import com.atcl.util.AtclConstants;
import com.atcl.util.ErrorCodes;
import com.atcl.stemmer.porter.PorterCorpusStemmer;

import java.util.*;

/**
 * Created by IntelliJ IDEA.
 * User: JSK
 * Date: Dec 2, 2006
 * Time: 3:45:03 AM
 * To change this template use File | Settings | File Templates.
 */
public class CorpusType extends AtclType
```

```
{
   private boolean isTrainingSet;

   private Categories categories;
   private Documents documents;
   private Terms terms;

   public CorpusType()
   {
      categories = new Categories();
      documents = new Documents();
      terms = new Terms();
   }

   public AtclType _copy()
   {
      return null;  //To change body of implemented methods use File | Settings | File Templates.
   }

   public AtclType _assign(AtclType newValue)
   {
      return null;  //To change body of implemented methods use File | Settings | File Templates.
   }

   public AtclType _equal(AtclType otherValue)
   {
      return null;  //To change body of implemented methods use File | Settings | File Templates.
   }

   public AtclType _and(AtclType otherValue)
   {
      return null;  //To change body of implemented methods use File | Settings | File Templates.
   }

   public AtclType _not(AtclType otherValue)
   {
      return null;  //To change body of implemented methods use File | Settings | File Templates.
   }

   public AtclType _or(AtclType otherValue)
   {
      return null;  //To change body of implemented methods use File | Settings | File Templates.
   }

   public CorpusType(Categories categories, Documents documents, Terms terms)
   {
      this.categories = categories;
      this.documents = documents;
      this.terms = terms;
   }

   public boolean isTrainingSet()
   {
      return isTrainingSet;
   }
```

```
public void setTrainingSet(boolean trainingSet)
{
   isTrainingSet = trainingSet;
}

public Categories getCategories()
{
   return categories;
}

public void setCategories(Categories categories)
{
   this.categories = categories;
}

public Documents getDocuments()
{
   return documents;
}

public void setDocuments(Documents documents)
{
   this.documents = documents;
}

public Terms getTerms()
{
   return terms;
}

public void setTerms(Terms terms)
{
   this.terms = terms;
}

public String getTypeName()
{
  return AtclConstants.CORPUS_TYPE;
}

/*
* This performs Corpus stemming
* */
public void _unaryPlus()
{
   if(DEBUG)
      debug("In unaryPlus of CorpusType");

   List termsList = new ArrayList(this.getTerms().getTermMapping().keySet());
   Iterator terms = termsList.iterator();
   //Iterator terms = this.getTerms().getTermMapping().c.entrySet().iterator();

   PorterCorpusStemmer stemmer = new PorterCorpusStemmer();
   String entry = null;
   String term,stemmedTerm = null;
   ArrayList termMappingList = null;
```

```
    while(terms.hasNext())
    {
        entry = (String)terms.next();
        stemmer.add(entry.toCharArray(),entry.length());
        stemmer.stem();
        stemmedTerm = stemmer.toString();

       if(entry.equals(stemmedTerm))
            continue;

           updateCorpusWithStemmedTerm(entry,stemmedTerm,(ArrayList)((HashMap)this.getTerms().g
           etTermMapping()).get(entry));
           terms.remove();
    }
}

/*
* This removes Noise from the Corpus using Zipf's Law
* */
public void _unaryMinus()
{
    if(DEBUG)
        debug("In unaryMinus of CorpusType");

    // Removing all the terms from the corpus which occurs in less than equal to 5 documents or more
    than equal to 4/5 of documents

    removeNoiseTerms(5,this.getDocuments().getDocumentMapping().size()*(4/5));
}

public AtclType _binaryPlus(AtclType otherValue)
{
    if (!(otherValue instanceof CorpusType))
        error(ErrorCodes.ATCL_NOT_VALID_CORPUS_TYPE,otherValue);

    Categories retCategs = new Categories(new
    HashMap(this.getCategories().getCategoryMapping()));

    Documents retDocuments = new Documents(new
    HashMap(this.getDocuments().getDocumentMapping()));

    Terms retTerms = new Terms(new HashMap(this.getTerms().getTermMapping()));

    CorpusType ret = new CorpusType(retCategs,retDocuments,retTerms);

    // Now we have to add documents, terms and categories from the otherValue corpus.

    Iterator otherValueTermsIter =
    ((CorpusType)otherValue).getTerms().getTermMapping().entrySet().iterator();

    String otherTerm = null;
    ArrayList termMappingList,otherTermMappingList = null;

    Iterator otherTermCategories,otherTermDocuments;
```

```
        Map.Entry entry1,entry2,entry3 = null;


ret.getDocuments().getDocumentMapping().putAll(((CorpusType)otherValue).getDocuments().
                getDocumentMapping());

ret.getCategories().getCategoryMapping().putAll(((CorpusType)otherValue).getCategories().getCategory
Mapping());

        while(otherValueTermsIter.hasNext())
        {
            entry1 = (Map.Entry)otherValueTermsIter.next();
            otherTerm = (String)entry1.getKey();

            otherTermMappingList = (ArrayList)entry1.getValue();
            otherTermCategories = ((HashMap)otherTermMappingList.get(1)).entrySet().iterator();
            otherTermDocuments = ((HashMap)otherTermMappingList.get(0)).entrySet().iterator();

            if(ret.getTerms().getTermMapping().containsKey(otherTerm))
            {
                // This term already there in the corpus. Update document, category mapping both ways
                termMappingList = (ArrayList)ret.getTerms().getTermMapping().get(otherTerm);

                while(otherTermDocuments.hasNext())
                {
                    entry2 = (Map.Entry)otherTermDocuments.next();
                    ((HashMap)termMappingList.get(0)).put(entry2.getKey(),entry2.getValue());
                }

                while(otherTermCategories.hasNext())
                {
                    entry2 = (Map.Entry)otherTermCategories.next();
                    ((HashMap)termMappingList.get(1)).put(entry2.getKey(),entry2.getValue());
                }
            }
            else
            {
                ((HashMap)ret.getTerms().getTermMapping()).put(otherTerm,otherTermMappingList);
            }
        }
        return ret;
    }

    private void removeNoiseTerms(int minimumThreshold,int maximumThreshold)
    {
        Iterator terms = this.getTerms().getTermMapping().entrySet().iterator();
        Map.Entry entry = null;
        String term = null;

        ArrayList termMappingList = null;
        Iterator termCategories,termDocuments;

        while(terms.hasNext())
        {
            entry = (Map.Entry)terms.next();
            term = (String)entry.getKey();
```

```
        termMappingList = (ArrayList)entry.getValue();

        if( (((HashMap)termMappingList.get(0)).size() <= minimumThreshold) ||
        (((HashMap)termMappingList.get(0)).size() >= maximumThreshold))
        {
            // Remove this term from the corpus.

            termMappingList = (ArrayList)this.getTerms().getTermMapping().get(term);

            termCategories = ((HashMap)termMappingList.get(1)).keySet().iterator();

            termDocuments = ((HashMap)termMappingList.get(0)).keySet().iterator();

            while(termCategories.hasNext())
            {
            ((HashMap)this.getCategories().getCategoryMapping().get(termCategories.next()).get(1)).rem
            ove(term);
            }

            while(termDocuments.hasNext())
            {
             ((HashMap)this.getDocuments().getDocumentMapping().get(termDocuments.next()).get(1)).
            remove(term);
            }
            terms.remove();
        }
    }
  }

  private void updateCorpusWithStemmedTerm(String oldTerm, String stemmedTerm,ArrayList
oldTermMappingList)
  {
     Float weight = 1.0f;
     Float frequency = 0.0f;

     Iterator oldTermDocuments = ((HashMap)oldTermMappingList.get(0)).keySet().iterator();
     Iterator oldTermCategories = ((HashMap)oldTermMappingList.get(1)).keySet().iterator();

     ArrayList stemmedTermMappingList = null;

     Map mapZero,mapOne = null;
     Long documentId = null;
     String category = null;


     /*
    * New Term is not there in the Term mapping
    * Update Term -> document (add OldTerm document if its not there)
    * Update Term -> Category (add OldTerm categroy if its not there)
    * Update Reverse mappings from Document - > Term and Category -> Term
    * Remove old term from the corpus and add new Term with the oldTerm information.
    * */
     if (!(((HashMap) this.getTerms().getTermMapping()).containsKey(stemmedTerm)))
     {
        stemmedTermMappingList = new ArrayList();
```

```
    mapZero = new HashMap();
    mapOne = new HashMap();

    stemmedTermMappingList.add(mapZero);
    stemmedTermMappingList.add(mapOne);

    ((HashMap)this.getTerms().getTermMapping()).put(stemmedTerm,stemmedTermMappingList);
}

// This term already exists. Add documentId and CategoryId mapping from oldTerm.
stemmedTermMappingList = (ArrayList) ((HashMap)
this.getTerms().getTermMapping()).get(stemmedTerm);

 while (oldTermDocuments.hasNext())
{
    documentId = (Long) oldTermDocuments.next();

        ((HashMap) ((ArrayList)
       this.getDocuments().getDocumentMapping().get(documentId)).get(1)).remove(oldTerm);

    if (((HashMap) stemmedTermMappingList.get(0)).containsKey(documentId))
    {
        // This stemmedTerm occurs more than once for the document.

        //Update Term to Document mappping
        frequency = (Float) ((HashMap) stemmedTermMappingList.get(0)).get(documentId);

       ((HashMap) stemmedTermMappingList.get(0)).put(documentId, new
       Float(frequency.floatValue() + 1.0));

        // Update Document to Term mapping
       weight = (Float) ((HashMap) ((ArrayList)
       this.getDocuments().getDocumentMapping().get(documentId)).get(1)).get(stemmedTerm);

       ((HashMap) ((ArrayList)
       this.getDocuments().getDocumentMapping().get(documentId)).get(1)).put(stemmedTerm, new
       Float( 1.0));
    }
    else
    {
        ((HashMap) stemmedTermMappingList.get(0)).put(documentId, new Float(1.0));

       ((HashMap) ((ArrayList)
       this.getDocuments().getDocumentMapping().get(documentId)).get(1)).put(stemmedTerm,
       weight);
    }
}
while (oldTermCategories.hasNext())
{
    category = (String) oldTermCategories.next();

        ((HashMap) ((ArrayList)
       this.getCategories().getCategoryMapping().get(category)).get(1)).remove(oldTerm);

    if (((HashMap) stemmedTermMappingList.get(1)).containsKey(category))
    {
```

```
            // This term occurs more than once for this category. Update its counter.
            Float termCategFrequency = (Float) ((HashMap)
            stemmedTermMappingList.get(1)).get(category);
            ((HashMap) stemmedTermMappingList.get(1)).put(category, new
            Float(termCategFrequency.floatValue() + 1.0));

            weight = (Float) ((HashMap) ((ArrayList)
            this.getCategories().getCategoryMapping().get(category)).get(1)).get(stemmedTerm);

            ((HashMap) ((ArrayList)
            this.getCategories().getCategoryMapping().get(category)).get(1)).put(stemmedTerm, new
            Float(1.0));
          }
          else
          {
             ((HashMap) stemmedTermMappingList.get(1)).put(category, new Float(1.0));
             ((HashMap) ((ArrayList)
            this.getCategories().getCategoryMapping().get(category)).get(1)).put(stemmedTerm, weight);

          }
        }
        this.getTerms().getTermMapping().remove(oldTerm);
        /*

       * if(this.getTerms().getTermMapping().containsKey(stemmedTerm))
       * New Term is already there in the Terms mapping.
       * Update Term -> document (add OldTerm document if its not there)
       * Update Term -> Category (add OldTerm categroy if its not there)
       * Update Reverse mappings from Document - > Term and Category -> Term
       * Remove old term from the corpus.
       * */
    }

    public String toString()
    {
        return "";
    }
}

package com.atcl.types.corpus;

import java.util.HashMap;
import java.util.ArrayList;
import java.util.List;

/**
 * Created by IntelliJ IDEA.
 * User: JSK
 * Date: Dec 2, 2006
 * Time: 3:45:34 AM
 * To change this template use File | Settings | File Templates.
 */
public class Categories
{
    private HashMap<Long, List> categoryMapping;
```

```
    public Categories()
    {
        categoryMapping = new HashMap();
    }

    public Categories(HashMap<Long, List> categoryMapping)
    {
        this.categoryMapping = categoryMapping;
    }

    public void addNewCategory(Long categoryId)
    {
        categoryMapping.put(categoryId,new ArrayList());
    }


    public HashMap<Long, List> getCategoryMapping()
    {
        return categoryMapping;
    }

}

package com.atcl.types.corpus;

import java.util.List;
import java.util.HashMap;

/**
 * Created by IntelliJ IDEA.
 * User: JSK
 * Date: Dec 2, 2006
 * Time: 3:45:25 AM
 * To change this template use File | Settings | File Templates.
 */
public class Terms
{
     private HashMap<Long, List> termMapping;

    public Terms()
    {
        termMapping = new HashMap();
    }


    public Terms(HashMap<Long, List> termMapping)
    {
        this.termMapping = termMapping;
    }

    public static void addNewTerm(CorpusType corpus, long docId, char [] docStr)
    {
        //documentMapping.put(document.getId(),new ArrayList());

    }
```

```
    public HashMap<Long, List> getTermMapping()
    {
        return termMapping;
    }
}

package com.atcl.types.corpus;

import java.util.List;
import java.util.HashMap;
import java.util.ArrayList;

/**
 * Created by IntelliJ IDEA.
 * User: JSK
 * Date: Dec 2, 2006
 * Time: 3:45:17 AM
 * To change this template use File | Settings | File Templates.
 */
public class Documents
{
    private HashMap<Long, List> documentMapping;

    public Documents()
    {
        documentMapping = new HashMap();
    }


    public Documents(HashMap<Long, List> documentMapping)
    {
        this.documentMapping = documentMapping;
    }

    public static void addNewDocument(CorpusType corpus, long docId, char [] docStr)
    {
        //documentMapping.put(document.getId(),new ArrayList());
    }


    public HashMap<Long, List> getDocumentMapping()
    {
        return documentMapping;
    }

}
```

11.9 Classifier

```
package com.atcl.classifier;

import com.atcl.types.corpus.CorpusType;
```

```
import java.util.*;

/**
 * Created by IntelliJ IDEA.
 * User: JSK
 * Date: Dec 11, 2006
 * Time: 9:54:07 PM
 * To change this template use File | Settings | File Templates.
 */
public class WinnowCorpusClassifier extends BaseCorpusClassifier
{
   private static final String WINNOW = "WINNOW";
   private static int m,tao;


   public void buildClassifier(CorpusType corpus, float learningRatePromo,float learningRateDemo)
   {
      // build classifier for all categories
      Iterator categories = corpus.getCategories().getCategoryMapping().keySet().iterator();

      HashMap hypVector = null;
      String category = null;

      while(categories.hasNext())
      {
         category = (String)categories.next();
         hypVector = new HashMap();

         // Initializze Hypothesis Vector with all weights to 1.
         initHypVector(hypVector, corpus.getTerms().getTermMapping().keySet());
         trainWinnow(hypVector,category, corpus, learningRatePromo,learningRateDemo);

         // update category hypothesis vector in the corpus
         //System.out.println("HypVector for category " + category + ":" + hypVector );

            ((HashMap)((ArrayList)corpus.getCategories().getCategoryMapping().get(category)).get(2)).pu
            t(WINNOW,hypVector);
      }
   }

   public void testClassifier(CorpusType trainingCorpus, CorpusType testCorpus)
   {
      Iterator testDocuments = testCorpus.getDocuments().getDocumentMapping().entrySet().iterator();
      Map.Entry entry = null;
      String predictedDocumentCategory = null;
      int mistakes = 0, correct = 0, notPredicted = 0;
      while(testDocuments.hasNext())
      {
         entry = (Map.Entry)testDocuments.next();

         predictedDocumentCategory = predictDocumentCategory(trainingCorpus, entry);

         if (((HashMap)((ArrayList)entry.getValue()).get(0)).containsKey(predictedDocumentCategory))
         {
            // Predicted Correct Category for the document.
            correct++;
```

```
        }
        else if(predictedDocumentCategory == "")
            notPredicted ++;
        else
            mistakes ++;
    }
    printTCEvaluation(mistakes,correct,notPredicted);
}

private String predictDocumentCategory(CorpusType trainingcorpus, Map.Entry documentEntry)
{
    String predictedCategory = "", currentCategory = null;
    float highestPredictedScore = 0.0f;
    float tempScore = 0.0f;
    Map.Entry categEntry = null;
    Iterator trainingCategories =

        trainingcorpus.getCategories().getCategoryMapping().entrySet().iterator();

    while(trainingCategories.hasNext())
    {
        categEntry = (Map.Entry)trainingCategories.next();
        currentCategory = (String)categEntry.getKey();
        tempScore =
        testWithCategory(((HashMap)((HashMap)((ArrayList)categEntry.getValue()).get(2)).get(WINN
        OW)),documentEntry);

        if ( (tempScore - highestPredictedScore) > 0.02 )
        {
            highestPredictedScore = tempScore;
            predictedCategory = currentCategory;
        }
    }
    return predictedCategory;
}

private float testWithCategory(HashMap hypVector, Map.Entry documentEntry)
{
    float score = 0.0f;
    String term = null;
    Iterator documentTermsIter =
((HashMap)((ArrayList)documentEntry.getValue()).get(1)).keySet().iterator();

    while(documentTermsIter.hasNext())
    {
        term = (String) documentTermsIter.next();
        if(hypVector.containsKey(term))
            score +=((Float)hypVector.get(term)).floatValue();
    }
    return score;
}

private void trainWinnow (Map hypVector, String category, CorpusType corpus, float
learningRatePromo,float learningRateDemo)
{
    float categoryHyp = 0.0f;
```

```
    boolean positiveExample = false;
    Long documentId = null;
    int mistake = 0;
    Iterator trainingdocuments = corpus.getDocuments().getDocumentMapping().entrySet().iterator();
    while(trainingdocuments.hasNext())
    {
       Map.Entry entry = (Map.Entry)trainingdocuments.next();
       documentId = (Long)entry.getKey();
       categoryHyp = classifyDocument(hypVector,(HashMap)((ArrayList)entry.getValue()).get(1));
       positiveExample = ((HashMap)((ArrayList)entry.getValue()).get(0)).containsKey(category);

       if ( (positiveExample) && (categoryHyp > 0.02))
       {
          // h(x) = c(x) everything seems to be working fine, don't change weights.
       }
       else
       {
          mistake++;
          if (positiveExample)
             adjustWeights(hypVector,learningRatePromo,documentId,corpus); //Promotion step
          else
             adjustWeights(hypVector,learningRateDemo,documentId,corpus ); // Demotetion step
       }
    }
    //System.out.println("Mistakes for Categroy " + category + ":" + mistake);
 }

 private void adjustWeights (Map hypVector, float learningRate, Long documentId, CorpusType
corpus)
 {
          HashMap documentTermMapping =
          (HashMap)((ArrayList)corpus.getDocuments().getDocumentMapping().get(documentId)).get(1)
          ;

    Iterator documentTerms = documentTermMapping.keySet().iterator();

    String term = null;
    float weight = 1.0f;
    while(documentTerms.hasNext())
    {
       term = (String)documentTerms.next();
       if (hypVector.containsKey(term))
       {
          weight = ((Float)hypVector.get(term)).floatValue();
          hypVector.put(term, new Float(weight+learningRate));
       }
    }
 }

 private float classifyDocument(Map hypVector, HashMap documentTerms)
 {
    String categ = null;
    Iterator terms = documentTerms.keySet().iterator();
    float weights = 0.0f;
    while(terms.hasNext())
    {
```

```
            weights += ((Float)hypVector.get((String)terms.next())).floatValue();
        }
        if(weights > tao)
            return 1;
        else
            return 0;
    }


    private void initHypVector(HashMap hypVector, Set terms)
    {
        m = terms.size();
        tao = m/2;
        Iterator iter = terms.iterator();
        while (iter.hasNext())
        {
            hypVector.put(iter.next(),new Float(1.0));
        }
    }

    private void printTCEvaluation(int mistakes,int correct,int notPredicted)
    {
        System.out.println("Total number of Mistakes made in Testing:" + mistakes);
        System.out.println("Total number of Correct in Testing:" + correct);
        System.out.println("Total number of Not predicted in Testing:" + notPredicted);
    }
}
```

11.10 Main Driver Program

```
import antlr.TokenStreamException;
import antlr.RecognitionException;
import antlr.collections.AST;

import java.io.Reader;
import java.io.FileReader;
import java.io.FileNotFoundException;

import com.atcl.util.AtclInterpreter;

public class Atcl
{

    public static void main(String[] args)
    {
        if (!(args.length ==1))
        {
            printUsage();
            return;
        }
        runProgram(args[0]);

    }
```

```
    public static void runProgram(String fileName)
    {
        try
        {
            Reader reader = new FileReader(fileName);
            aTClLexer lexer = new aTClLexer(reader);
            aTClParser parser = new aTClParser(lexer);
            parser.program();
            AST tree = parser.getAST();
            AtclInterpreter ipt = new AtclInterpreter();
            aTClTreeWalker walker = new aTClTreeWalker(ipt);
            walker.program(tree);
        }
        catch (RecognitionException e)
        {
            System.err.println(e.toString());
        }
        catch (TokenStreamException e)
        {
            System.err.println(e.toString());
        }
        catch(FileNotFoundException fnfe)
        {
            System.err.println(fnfe.toString());
        }

    }
}
```

11.11 Reuters 21578 Corpus Document Structure

SGML format is being used to describe documents in each file.

11.11.1 Sample Document

```
<REUTERS TOPICS="YES" LEWISSPLIT="TRAIN" CGISPLIT="TRAINING-SET" OLDID="5544" NEWID="1">
    <DATE>26-FEB-1987 15:01:01.79</DATE>
    <TOPICS>
        <D>cocoa</D>
    </TOPICS>
    <PLACES>
        <D>el-salvador</D>
        <D>usa</D>
        <D>uruguay</D>
    </PLACES>
    <PEOPLE/>
    <ORGS/>
    <EXCHANGES/>
    <COMPANIES/>
    <UNKNOWN>
        &#5;&#5;&#5;C T
        &#22;&#22;&#1;f0704&#31;reute
        u f BC-BAHIA-COCOA-REVIEW   02-26 0105
    </UNKNOWN>
    <TEXT>&#2;
        <TITLE>BAHIA COCOA REVIEW</TITLE>
        <DATELINE>    SALVADOR, Feb 26 - </DATELINE>
        <BODY>Showers continued throughout the week in the Bahia cocoa zone, alleviating the drought since early January and
        improving prospects for the coming temporao, although normal humidity levels have not been restored, Comissaria Smith said in
        its weekly review.     The dry period means the temporao will be late this year.     Arrivals for the week ended February 22 were
        155,221 bags of 60 kilos making a cumulative total for the season of 5.93 mln against 5.81 at the same stage last year. Again it
        seems that cocoa delivered earlier on consignment was included in the arrivals figures. Comissaria Smith said there is still some
        doubt as to how much old crop cocoa is still available as harvesting has practically come to an end. With total Bahia crop
```

```
estimates around 6.4 mln bags and sales standing at almost 6.2 mln there are a few hundred thousand bags still in the hands of
farmers, middlemen, exporters and processors. There are doubts as to how much of this cocoa would be fit for export as
shippers are now experiencing dificulties in obtaining +Bahia superior+ certificates. In view of the lower quality over recent
weeks farmers have sold a good part of their cocoa held on consignment. Comissaria Smith said spot bean prices rose to 340 to
350 cruzados per arroba of 15 kilos. Bean shippers were reluctant to offer nearby shipment and only limited sales were booked
for March shipment at 1,750 to 1,780 dlrs per tonne to ports to be named. New crop sales were also light and all to open ports
with June/July going at 1,850 and 1,880 dlrs and at 35 and 45 dlrs under New York july, Aug/Sept at 1,870, 1,875 and 1,880 dlrs
per tonne FOB. Routine sales of butter were made. March/April sold at 4,340, 4,345 and 4,350 dlrs. April/May butter went at
2.27 times New York May, June/July at 4,400 and 4,415 dlrs, Aug/Sept at 4,351 to 4,450 dlrs and at 2.27 and 2.28 times New
York Sept and Oct/Dec at 4,480 dlrs and 2.27 times New York Dec, Comissaria Smith said. Destinations were the U.S.,
Covertible currency areas, Uruguay and open ports. Cake sales were registered at 785 to 995 dlrs for March/April, 785 dlrs for
May, 753 dlrs for Aug and 0.39 times New York Dec for Oct/Dec. Buyers were the U.S., Argentina, Uruguay and convertible
currency areas. Liquor sales were limited with March/April selling at 2,325 and 2,380 dlrs, June/July at 2,375 dlrs and at 1.25
times New York July, Aug/Sept at 2,400 dlrs and at 1.25 times New York Sept and Oct/Dec at 1.25 times New York Dec,
Comissaria Smith said. Total Bahia sales are currently estimated at 6.13 mln bags against the 1986/87 crop and 1.06 mln bags
against the 1987/88 crop. Final figures for the period to February 28 are expected to be published by the Brazilian Cocoa Trade
Commission after carnival which ends midday on February 27. Reuter
&#3;
</BODY>
</TEXT>
</REUTERS>
```

11.11.2 Document format

Each document is enclosed in the following REUTERS tag.

```
<REUTERS TOPICS=?? LEWISSPLIT=?? CGISPLIT=?? OLDID=?? NEWID=??>
        <DATE> </DATE> // occurs once
        <TOPICS> <D></D> //atleast once </TOPICS> // occurs once

</REUTERS>
```

TOPICS: The possible values are YES, NO, and BYPASS

LEWISSPLIT: The possible values are TRAINING, TEST, and NOT-USED

CGISPLIT: The possible values are TRAINING-SET and PUBLISHED-TESTSET

OLDID: The identification number (ID) the story had in the Reuters-22173 collection

NEWID: The identification number (ID) the story has in the Reuters-21578

1) <DATE> </DATE> [ONCE, SAMELINE]: Encloses the date and time of the document, possibly followed by some non-date noise material.

2) <MKNOTE> </MKNOTE> [VARIABLE] : Notes on certain hand corrections that were done to the original Reuters corpus by Steve Finch.

3) <TOPICS> </TOPICS> [ONCE, SAMELINE]: Encloses the list of TOPICS categories, if any, for the document. If TOPICS categories are present, each will be delimited by the tags <D> and </D>.

4) <PLACES> </PLACES> [ONCE, SAMELINE]: Same as <TOPICS> but for PLACES categories.

5) <PEOPLE> </PEOPLE> [ONCE, SAMELINE]: Same as <TOPICS> but for PEOPLE categories.

6) <ORGS> </ORGS> [ONCE, SAMELINE]: Same as <TOPICS> but for ORGS categories.

7) <EXCHANGES> </EXCHANGES> [ONCE, SAMELINE]: Same as <TOPICS> but for EXCHANGES categories.

8) <COMPANIES> </COMPANIES> [ONCE, SAMELINE]: These tags always appear adjacent to each other, since there are no COMPANIES categories assigned in the collection.

9) <UNKNOWN> </UNKNOWN> [VARIABLE]: These tags bracket control characters and other noisy and/or somewhat mysterious material in the Reuters stories.

10) <TEXT> </TEXT> [ONCE]: We have attempted to delimit all the textual material of each story between a pair of these tags. Some control characters and other "junk" material may also be included. The whitespace structure of the text has been preserved. The <TEXT> tag has the following attribute:

TYPE: This has one of three values: NORM, BRIEF, and UNPROC. NORM is the default value and indicates that the text of the story had a normal structure. In this case the TEXT tag appears simply as <TEXT>. The tag appears as <TEXT TYPE="BRIEF"> when the story is a short one or two line note. The tags appears as <TEXT TYPE="UNPROC"> when the format of the story is unusual in some fashion that limited our ability to further structure it.

The following tags optionally delimit elements inside the TEXT element. Not all stories will have these tags:

a. <AUTHOR>, </AUTHOR>: Author of the story.
b. <DATELINE>, </DATELINE>: Location the story originated from, and day of the year.
c. <TITLE>, </TITLE>: Title of the story. We have attempted to capture the text of stories with TYPE="BRIEF" within a <TITLE> element.
d. <BODY>, </BODY>: The main text of the story.

11.11.3 Category Set

Category Set	Number of Categories	Number of Categories w/ 1+ Occurrences	Number of Categories w/ 20+ Occurrences
**********	**********	**********	**********
EXCHANGES	39	32	7
ORGS	56	32	9
PEOPLE	267	114	15
PLACES	175	147	60
TOPICS	135	120	57