

# FEATURES EXTRACTION ALGORITHM FROM SGML FOR CLASSIFICATION

Zailani Abdullah[1], Muhammad Suzuri Hitam[2]

[1,2] Department of Computer Science, Faculty of Science and Technology,
University Malaysia Terengganu, 12030 K.Terengganu, MALAYSIA

**ABSTRACT**

The basic phases in text categorization include preprocessing features, extracting relevant features against the features in a database, and finally categorizing a set of documents into predefined categories. Most of the researches in text categorization are focusing more on the development of algorithms and computer techniques. An algorithm for pre-processing features is seem to be like a "black-box" and ignored by them. Thus, it is significant and worthwhile to develop an algorithm for preprocessing features and finally can be used by other beginners before going in depth in the field of text categorization. This research proposes an algorithm for preprocessing features with capability of Microsoft .NET framework technology. The actual implementation shows that, this algorithm can extract interested features from the standard corpus of collection and upload into a relational database.

**Keyword:** *Preprocessing, text categorization, algorithm, .net*

## 1. INTRODUCTION

With the rapid growth of the online information, text categorization is playing undeniable contributions to many information organization and management tasks. According to Lewis[1], a general definition of text categorization is

"... the task of deciding whether a piece of text belongs to any of a set prescribe categories. It is a generic text processing task useful in indexing for later retrieval, as a stage in NLP system, for content analysis, and many other roles"

There are many standard test collections of text categorization. Reuters-21578 dataset [2] is one of them. This collection has been used widely in a number of studies especially in information retrieval, machine learning and other corpus-based research. The Reuters-21578 collection is freely available in the Internet. The files are in Standard Generalized Markup Language (SGML) format.

SGML, defined by ISO 8879, is a meta-language for defining markup languages for documents [3]. It is descendent of IBM's Generalized Markup Language (GML) created in the 1960s. As a markup language, it has a specific vocabulary (elements and attributes) and a declared syntax (defined grammars).

In 1998, World Wide Web Consortium (W3C) has published and recommended Extended Markup Language (XML) [4] for Internet community. XML is a profile or subset of SGML. It was designed to describe data and to focus on what data is. Due to a number of technical reasons in SGML, XML becomes more acceptable for serving documents over the web.

.NET [5] framework is a large set of class libraries that can be used by many programming languages, like Microsoft's C#, Visual Basic, Managed C++ and more. This class library provides the common system services and functions for managing .NET applications. It also enables integration with XML and XML Web service into .NET applications.

Microsoft designed C# to be the core programming language of the .NET platform and to use, leverage and extend the .NET Framework class library. As an object oriented programming language, C# has its own syntax, control structures and construction building statements. Compilation of C# generates managed code that will run under the .NET environment.

An algorithm [6] is a procedure or formula for accomplishing some task which, given an initial state, will terminate in a defined end-state. The



word derives from the name of the mathematician, Mohammed ibn-Musa al-Khwarizmi, lived from 780 to 850 in Baghdad. The algorithm is very important in programming because it depicts a complexity and efficiency of solving a problem in computing. In many programming languages, algorithms are implemented as functions or procedures.

Database Management System (DBMS) [7] is software that defines a database, stores the data, supports a query language, produce reports and create data entry screens. The most popular DBMS are IBM DB2, Oracle, Sybase, Microsoft SQL Server and Microsoft Access. The Microsoft Access is suitable for an applications running on Windows operating system. From the business point of views, more consideration will be given to the well established databases such as IBM DB2, Oracle, etc. rather than Microsoft Access.

An initial phase in TC is to extract important features from a corpus (texts collection). This task is also called preprocessing features. It involves "scanning" all texts from a corpus, and extracting useful features. Vector features of documents/words will be generated based on these features. Although there are many studies has been done to generate vector features, the detail algorithms are rarely shown. This paper presents a complete algorithm for extracting important features from the Reuters dataset and finally transforming all these features into relational database.

## 2. APPROACH AND METHODS

The "Reuters-21578, Distribution 1.0" corpus consist of stories appeared on the Reuters newswire in 1987. This corpus was first used in the CONSTRUE text categorization system (Hayes & Weinstein, 1990) based on a Reuters-22173. This new version was introduced in order to fix all the problems such as duplication of stories, typographical errors, etc.

### 2.1. Initialize All Parameters

At this stage, all public and private classes will be initialized. After initialization, these classes will be using by the .NET programming (C#).

### 2.2. Merge all SGML files

From 21,578 documents in Reuters-21578, only 12,902 of them can be used in TC experiments. The other 8,676 documents are not properly labeled. The corpus distributed in 22 files and each of the files has 1000 documents except the last only has 578. All these files are formatted in SGML labels to identify the category of EXCHANGE, ORGS, PEOPLE, PLACE and TOPICS. However, only TOPICS categories are used widely in the TC experiments. All 22 files will be merged together to form a file. Figure 1 shows apart of SGML file after merging of documents.

```
<!DOCTYPE lewis SYSTEM "lewis.dtd">
<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></PEOPLE>
<ORGS></ORGS>
<EXCHANGES></EXCHANGES>
<COMPANIES></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, ...
&#3;</BODY></TEXT>
</REUTERS>
```

Figure 1. SGML file

### 2.3. Convert SGML to XML

Since, Reuters-21578 is made up of SGML "meta" language, a mechanism to read and convert from SGML to XML formats is required. A SgmlReader version 1.7 will be employed to accomplish this task. The SgmlReader is an implementation of XmlReader API over any SGML document. In short, it will convert all .sgm files using the existing built in HTML DTD. Figure 2 shows an output in XML format after conversion.

```
  <?xml version="1.0" ?>
- <html>
- <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>C T f0704reute u f BC-
BAHIA-COCOA-REVIEW 02-26
0105</UNKNOWN>
- <TEXT>

  <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, ... Reuter
_____</BODY>
  </TEXT>
  </REUTERS>
```

Figure 2. Sample XML format

The output also depicts that there are no missing information during the conversion processes. This new corpus (XML format) will be used as a new source for the next phases of preprocessing features.

### 2.4. Extract important XML elements & data

An XML document consists of element, attribute and data. However, in this new corpus, not all of them are useful. For example the data in DATELINE element shows in Figure 2 has nothing to do for TC. Therefore, only significant and meaningful elements and data will be selected during refinement process. Figure 3 shows a new corpus of XML after further refinement.

```
  <?xml version="1.0" ?>
- <html>
- <REUTERS TOPICS="YES"
LEWISSPLIT="TRAIN"
CGISPLIT="TRAINING-SET"
OLDID="5544" NEWID="1">
- <TOPICS>
  <D>cocoa</D>
  </TOPICS>
- <TEXT>

  <TITLE>BAHIA COCOA REVIEW</TITLE>
  <BODY>Showers continued
throughout the week in the Bahia
cocoa zone, alleviating the drought
since early January and improving
prospects for the coming temporao,
... Reuter
_____</BOD
Y>
  </TEXT>
```

Figure 3. Sample XML format (Filtered)

This phase is needed to avoid more complexity of extracting data in the future processes.

### 2.5. Tokenize and Remove Stopping Words

Tokenize is a process of splitting the sentence into word tokens. It is considered as a sub-task of parsing input. An example of this process can be simulated based on sentences in element <TEXT> in Figure 3 as shown in Figure 4.

```
Showers continued throughout the
week in the Bahia cocoa zone,
alleviating the drought since
early January and improving
prospects for the coming
temporao
```

Figure 4. Sample Sentences

The sample words after tokenizing are shown in Table 1.



Table 1. Word After Tokenization

| Tokenized Word |
|---|
| showers |
| continued |
| throughout |
| the |
| week |
| in |
| the |
| bahia |
| cocoa |
| zone |
| alleviating |
| the |
| drought |
| since |
| early |
| January |
| and |
| improving |
| prospects |
| for |
| the |
| coming |
| temporao |

Table 1 shows that the tokenization process is not only splitting the words but also changing entire tokenized words into a lowercase format. Beside that, all unnecessary characters such as "," will be removed from the list.

All tokenized words will then undergo the process of removing stop words. A stop word is a type of word that appears very frequently in a text collection, meanwhile it has no great significance. In the Information Retrieval (IR) community, stop words are defined as grammatical or function words. For instance, prepositions, coordinators, determinants are stop-words. Common stop words include:

a, an, the, in, of, on, are, be, if, into, which

Those words are useless for search and retrieval purposes. This means, in the case of extracting, they must be eliminated in order to obtain the adequate key terms to describe the text. There are many stop words exist in the new corpus. Thus, to purge them out, a list of predefined stop words must be developed first. The program will then identify and finally remove all the stop words in the corpus based on the predefined list.

From the example in Table 1, the stop words detected are shown in Table 2.

Table 2. Detected Stop Words

| Stop Word |
|---|
| the |
| in |
| since |
| and |
| for |

**2.6. Word Stemming**

Stemming is a process of determining the morphological root of a given inflected word form. A stemmer for English identifies the string "stemmer", "stemming", "stemmed" as based on the root "stem".

Porter stemming algorithm [8] was first written by Martin Porter, and published in the July 1980 issue of the journal Program. It is a process for removing the commoner morphological and inflexional endings from words in English. Its main use is as part of a term normalization process in Information Retrieval systems. Nowadays, this stemmer became the de-facto standard algorithm used for English stemming.

The Porter stemming algorithm has been employed to normalize the terms in the new corpus collection. Examples of terms that have gone through the stemming processes are shown in Table 3.

Table 3. Stemming Words

| Original Word | Stemmed Word |
|---|---|
| showers | Shower |
| continued | continu |
| throughout | throughout |
| week | week |
| bahia | bahia |
| cocoa | cocoa |
| zone | zone |
| alleviating | allevi |
| drought | drought |
| since | sinc |
| early | earli |
| january | januari |
| improving | Improve |
| prospects | prospect |
| coming | come |
| temporao | temporao |



### 2.7. Generate Vector Features

The term frequency (tf) in the given document measures the importance of the term $t_i$ within the particular document.

$tf = (1)$

| Relations (tables) |
|---|
| TermFrequency |
| TermFerquencyTest |
| Reference |
| TopTenTopic |
| Thesauri |

with $n_i$ is the number of occurrences of the term, and the denominator is the number of occurrences of all terms.

The inverse document frequency (idf) is a measure of the general importance of the term. It is the logarithm of the number of all documents divided by the number of documents containing the term.

$$idf = \log \frac{|D|}{|(d_i \supset t_i|} \qquad (2)$$

with

- $|D|$: total number of documents in the corpus
- $|(d_i \supset t_i|$: number of documents where the term $t_i$ appears (that is $n_i \neq 0$).

Then

$$tf - idf = tf * idf \qquad (3)$$

The *tf-idf* weight often used in text mining. This weight is a statistical measure used to evaluate the important of a word is in a document. The importance increases proportionally to the number of times a word appears in the document but is offset by how common the word is in all of the documents in the corpus of collection.

### 2.8. Upload into Database

Before uploading entire vector features into Microsoft Access Database, developments of entity classes are required. During this process, normalization of all relations is crucial. Normalization is the process of converting complex data structures into well-structures relation. In these relations, amount of redundancy are very minimum. Table 4 shows important relations to upload vectors features and related information into database.

Table 4. List of Main Relations

Filtering of vector features based on threshold parameter is not focus at this time. It will be useful to upload entire vector features for further analysis on determining the suitable thresholds.

Summary of an algorithm for extracting features from the corpus of collection in SGML format is shown in Figure 5.

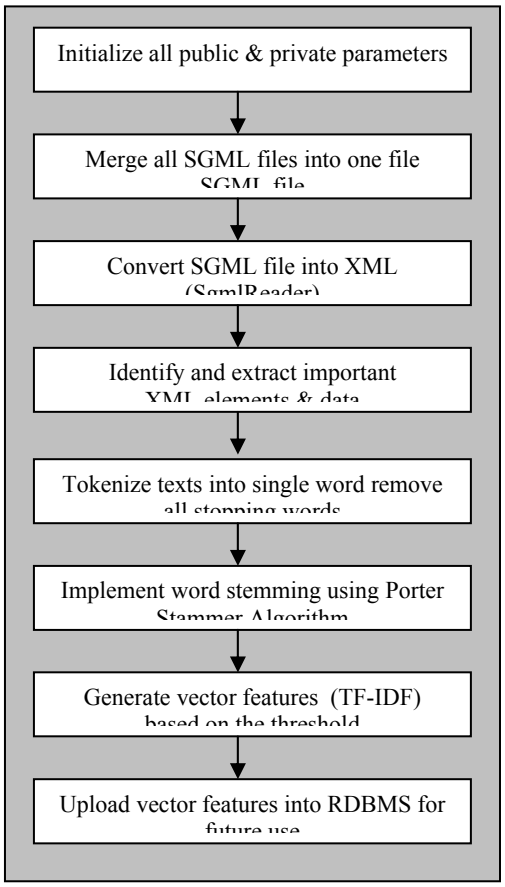


Figure 5. An Algorithm for Extracting Features

**RESULT & DISCUSSION**



From the 21,578 documents in Reuters-21578 and after detail analysis, only 12,902 of them can be used in TC experiments. There are 135 categories of documents and only top ten categories will be employed. Figure 6 shows list of selected categories with its description.

Table 6. Top Ten Category and Description

| No | Category | Description |
|---|---|---|
| 1 | earn | Earnings and Earnings Forecasts |
| 2 | Acq | Mergers/ Acquisitions |
| 3 | money-fx | Money/Foreign Exchange |
| 4 | grain | Grain (Commodity) |
| 5 | crude | Crude Oil |
| 6 | trade | Trade |
| 7 | interest | Interest Rates |
| 8 | ship | Shipping |
| 9 | wheat | Wheat (Commodity) |
| 10 | corn | Corn (Commodity) |

By following the top ten categories, out of 12,902 documents, only 71.82% will be selected for the purposes of training and testing. Table 7 shows number of documents versus purposes.

Table 7. Number of Documents versus Purpose

| Purpose | Num. of Docs | Per(%) |
|---|---|---|
| Training | 6,495 | 71.82 |
| Testing | 2,548 | 28.18 |
| | 9,043 | 100.00 |

From the selected 9,043 documents, the details numbers of documents against category are shown in Table 9. Since the same documents may become more than one category, the overall total of training and testing sets will definitely not tally with total figure as shown in Table 7.

Table 9. Number of Documents versus purpose

| No | Category | Training | Testing |
|---|---|---|---|
| 1 | earn | 2,877 | 1,088 |
| 2 | acq | 1,650 | 719 |
| 3 | money-fx | 539 | 180 |
| 4 | grain | 434 | 149 |
| 5 | crude | 391 | 189 |
| 6 | trade | 369 | 117 |
| 7 | interest | 347 | 133 |
| 8 | ship | 198 | 89 |
| 9 | wheat | 212 | 71 |
| 10 | corn | 183 | 56 |

Threshold value can be used to reduce the number of terms used for categorization. Not all terms will give a significant contribution for categorization. There are too many terms that only appear once in the document. Table 10 shows number of terms counted after removing unwanted terms based on the threshold value approach.

Table 10. Number of Terms versus Threshold

| Threshold value | Num. of Terms |
|---|---|
| ≥ 2 | 6,345 |
| ≥ 5 | 1,756 |
| ≥ 10 | 412 |

Based on 9,043 documents, 22,815 terms have been determined and extracted. The details number of terms involved for respective training and testing sets is shown in Table 8.

Table 8. Number of Terms versus Purpose

| Purpose | Num. of Terms | Per(%) |
|---|---|---|
| Training | 14,086 | 61.74 |
| Testing | 8,729 | 38.36 |
| | 22,815 | 100.00 |

## CONCLUSION

This algorithm provides a very clear spectrum on how to perform feature extractions processes for Reuter-21578. It also generates detail analysis about data being extracted. This analysis reveals characteristics of the corpus and it contributions in TC.

## ACKNOWLEDGEMENTS

Thanks to all colleagues for valuable comments during preparing this paper.

## REFERENCES

[1] D.Lewis. The reuters-21578 data set, 1987 http://www.davidlewis.com/resources/testcollections/-reuters21578/

[2] David D. Lewis, Reuters-21578 text categorization test collection http://www.daviddlewis.com/resources/testcollections/reuters21578/readme.txt



[3] International Standard ISO 8879 Information Processing – Text and office systems – Standard Genelized Markup Language (SGML), International Organization for Standard, Switzerland, 1986

[4] Bray, T., Paoli, J., and Sperberg-McQueen, C., M. Extensible Markup Language (XML) 1.0. W3C Recommendation, February1998.

[5] David Chappell (2006), Introducing .NET, Addison Wesley Professional, Essex, UK

[6] Algorithm, WhatIs.Com http://whatis.techtarget.com/definition

[7] Connolly,T and Begg, C (2005), Database Systems, Addison Wesley, Essex, UK.

[8] Porter, M.F. an algorithm for suffix stripping. Program 14, 3 (1980) 130-137