P-2579

# AUTO SUMMARIZATION TOOL

A PROJECT REPORT

*Submitted by*


N. BALA ARCHANA 71205205010

R. ESTHER 71205205016

M. SATHIYAVATHI 71205205051


*in partial fulfillment for the award of the degree*

*of*

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY




KUMARAGURU COLLEGE OF TECHNOLOGY, COIMBATORE

ANNA UNIVERSITY : CHENNAI 600 025


APRIL 2009

# ANNA UNIVERSITY : CHENNAI 600 025

## BONAFIDE CERTIFICATE

Certified that this project report "**AUTO SUMMARIZATION TOOL**" is the bonafide work of "**N. BALA ARCHANA, R. ESTHER and M. SATHIYAVATHI**" who carried out the project work under my supervision.

**SIGNATURE**

**Ms. S.Kavitha M. E.**

**SUPERVISOR,**

Dept of Information Technology,

Kumaraguru College of Technology,

Coimbatore – 641 006.

**SIGNATURE**

**Dr. Thangasamy, B.E(Hons)., Ph.D.**

**DEAN,**

Dept of Computer Science and Engineering,

Kumaraguru College of Technology,

Coimbatore – 641 006.

The candidates with University Register No 71205205010,71205205016 and 71205205051 were examined by us in the project viva-voice examination held on .27.4.09

INTERNAL EXAMINER

EXTERNAL EXAMINER

# DECLARATION

We,

| | | |
|---|---|---|
| **N. BALA ARCHANA** | **Reg.No:** | **71205205010** |
| **R. ESTHER** | **Reg.No:** | **71205205016** |
| **M. SATHIYAVATHI** | **Reg.No:** | **71205205051** |

hereby declare that the project entitled **"AUTO SUMMARIZATION TOOL"**, submitted in partial fulfillment to Anna University as the project work of Bachelor of Technology (Information Technology) degree, is a record of original work done by us under the supervision and guidance of Department of Information Technology, Kumaraguru College of Technology, Coimbatore.

Place: Coimbatore

Date: 27.04.09

N. Bala Archana  R. Esther  M. Sathiyavathi

[N. Bala Archana]  [R. Esther]  [M. Sathiyavathi]

Project Guided by

S. Kavitha

[Ms. **S. Kavitha M.E.**]

[Ms. **L.S Jayashree M.E., Ph.D**]

# ACKNOWLEDGEMENT

We express our sincere thanks to our Chairman **Padmabhushan Arutselvar Dr. N. Mahalingam B.Sc., F.I.E.,** Vice Chairman **Dr.K. Arumugam B.E., M.S., M.I.E.,** Correspondent **Shri.M.Balasubramaniam and Joint Correspondent Dr.A.Selvakumar** for all their support and ray of strengthening hope extended. We are immensely grateful to our Vice Principal **Prof. R. Annamalai,** for his invaluable support to the outcome of this project.

We are deeply obliged to **Dr.S.Thangasamy,** Dean, Department of Computer Science and Engineering for his valuable guidance and useful suggestions during the course of this project.

We also extend our heartfelt thanks to our project co-ordinator, **L.S Jayashree M.E., Ph.D.,** Asso.Prof, Department of Information Technology for providing us her support which really helped us.

We are indebted to our project guide and extend our gratitude towards **Ms. S.Kavitha M.E.,** Lecturer, Department of Information Technology for her helpful guidance and valuable support given to us throughout this project.

We thank the teaching and non-teaching staffs of our Department for providing us the technical support during the course of this project. We also thank all of our friends who helped us to complete this project successfully.

## ABSTRACT

Automatic text summarization is to compress an original document into an abridged version by extracting almost all of the essential concepts with text mining techniques. There are two common approaches employed in automatic summary extraction: the statistical approach and the linguistic approach. Statistics-based approaches derive weights of key terms and determine the sentence importance by the total weight the sentence contains, whereas linguistics-based approaches identify term relationship in the document through part-of-speech tagging, grammar analysis, thesaurus usage, and the like, and extract meaningful sentences. The proposed approach is a hybrid automatic text summarization approach, KCS, to enhance the quality of summaries.

KCS employs the K-mixture probabilistic model to establish term weights in a statistical sense, and further identifies the term relationships to derive the connective strength (CS) of nouns. Sentences are ranked and their CS values are determined. Sentences with significant connective strength nouns are extracted to form the summary accordingly.

# *CONTENTS*

## TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# *INTRODUCTION*

# 1. INTRODUCTION

## 1.1 GENERAL:

Text mining refers to the discovery of new, previously unknown information by automatically extracting information from different written resources. Therefore, it sometimes alternatively referred to as text data mining. It tries to analyze large quantities of text and find valuable patterns hidden in the textual documents. Similar to the various analysis types of data mining, text mining includes discovery of patterns and trends in data, associations among entities, predictive rules, etc.

Text mining process mainly consists of three steps: text preparation, text processing, and text analysis. Text preparation is to preprocess the text and extract the meaningful terms, or "features", from the text. In the text processing step, data mining techniques are applied to identify interesting patterns from the preprocessed data. Finally, in text analysis step, the output from the previous step is evaluated to examine whether the extracted knowledge is important. In recent years, text mining has been studied in many fields, which includes term association discovery, document clustering, **text summarization**, and text categorization, etc.

Text summarization is to compress an original document into a shortened version by extracting the most important information out of the document. Instead of keyword extraction, text summarization identifies the most important paragraphs or sentences from a given document, excluding unimportant detailed information.

## 1.2 PROBLEM DEFINITION:

Due to the rapid growth of the World Wide Web, information is much easier to disseminate and acquire than before. Finding useful and favored documents from the huge text repository creates significant challenges for users. It is therefore essential to develop tools to efficiently assist users in identifying desired documents. The ability to summarize information automatically and present results to the end user in a compressed, yet complete form would help to solve this problem. Text summarization is the process of automatically creating a compressed version of a given text that provides useful information for the user. One possible means is to utilize automatic text summarization. Automatic text summarization has received a great deal of attention in recent research. Automatic text summarization is a text-mining task that extracts essential sentences to cover almost all the concepts of a document. It is to reduce users' consuming time in document reading without losing the general issues for users' comprehension. With document summary available, users can easily decide its relevancy to their interests and acquire desired documents with much less mental loads involved.

Automatic text summarization becomes more and more important in many applications. For example, present search engines usually provide a short summary for each retrieved document in order that users can quickly skim through the main content of the page. Therefore it saves users time and improves the search engine's service quality. Automatic text summarization technologies also help for the development of the Knowledge Grid[2] . According to Fran Berman, the Knowledge Grid is a mechanism that can

synthesize knowledge from data through mining and reference methods and enable search engines to make references, answer questions, and draw conclusions from masses of data.

Text summarization can be performed in two different approaches: extraction and abstraction. The extraction approach is to construct the summary by taking the most important sentences out of the original document. In contrast, the abstraction approach is to form summary by paraphrasing sections of the original document. In general, abstraction is more powerful and generates closer results to those composed by humans than extraction, but less addressed in literature though. Instead of trying to compose an abstract summary as human does, we focus on text summarization are limited to a comparatively easy method - extraction, which selects pieces from the source document and concatenates them as a summary.

## 1.3 OBJECTIVE:

The Objective of this project is to develop a tool that summarizes text documents. A characteristic of natural languages is their inherent vagueness and uncertainty. To deal with this uncertainty, we use a hybrid approach (KCS) which yields quality summaries. The need for such a facility is made more acute by the huge amount of various documents on the Internet.

We use a K-mixture connective-strength-based (KCS) approach to enhance the quality of document summary results. The K-mixture probabilistic model is used to determine the term weights. Term relationships are then investigated to develop the connective strength of nouns that manifests sentence semantics. Sentences with significant connective strength nouns are extracted to form the summary accordingly.

# *LITERATURE REVIEW*

# 2. LITERATURE REVIEW

Inspired by the article "A hybrid approach to automatic text summarization" published in IEEE Transactions on Computer and Information Technology, on July 2008, we have taken up this project. Though this project is not the complete implementation of the paper, it can be treated as the base work.

## 2.1 Text summarization:

Extensive use of internet is one of the reasons why automatic text summarization draws substantial interest. It provides a solution to the information overload problem people face in this digital era. Text Summarization is not a new idea. Text summarization provides users with summaries of document contents, allowing them to quickly understand the main ideas of documents. Research on automatic text summarization has a very long history, which can date back at least 40 years ago, from the first system built at IBM. Several researchers continued investigating various approaches to this problem through the seventies and eighties. Recently it attracts much attention, the goal is still elusive and far from reach. Many innovative approaches began to be explored such as statistical and linguistic approaches. Early text summarization techniques are mainly based on shallow textual features (statistic) to evaluate sentence salience. For example, using term frequency as term weight and treating a sentence as salient if it included important terms. Cue phrases, position, and term frequencies are used to depict sentence salience. With the development of natural language processing technology, some advanced technologies such

as lexical chains, anaphora resolution, and natural language generation have been applied to document summarization. A summary is called abstractive if it is produced by above advanced techniques. Those technologies can improve the quality of summarization, but they themselves are not mature yet. Therefore, up to the present, most summaries are extractive which just extract the original salient sentences into the summary output. An extractive summary is readable and has encouraging results in comparison to the abstractive one. Our project is about improving the quality and the coherence of extractive summary by making use of hybrid approach which makes use of both statistical and certain linguistic features.

## 2.2 Existing approaches:

Summarization methods can be roughly grouped into two categories: Statistical approach, linguistic approach.

### Statistical Approach:

The statistical approach summarizes without understanding; it relies on the statistical distribution of certain features. Statistical techniques base themselves on term frequency to determine the term importance. Sentences with more important terms are extracted in higher priorities. Common ways to determine term importance include TF–IDF, entropy, mutual information, and $\chi 2$ statistics. Sometimes, term importance is reinforced if the terms belong to title words, cue-phrases, and/or capitalized words. Moreover, sentence importance can also be adjusted according to its length and where it locates in the document. Briefly speaking the statistical approach is domain-independent and fast, but has an inherent upper bound of performance.

### Limitations:

Statistical approach is domain-independent. The output summary generated by statistical techniques is not very coherent.

**Linguistic Approach:**

Summarization based on these methods need linguistic knowledge so that the computer can analyze the sentences semantically and then decides which sentences to choose considering the position of the verb, subject, noun and etc. Linguistic techniques extract sentences by means of natural language processing such as grammatical property analysis and term semantics examination. Linguistic techniques generally start from term parsing and part-of-speech tagging in order to find key terms. Those terms are then extended using thesaurus (e.g. Wordnet) to explore term relationships. Important sentences are extracted based on their semantics derived from those term relationships. They yield better summary than statistical methods.

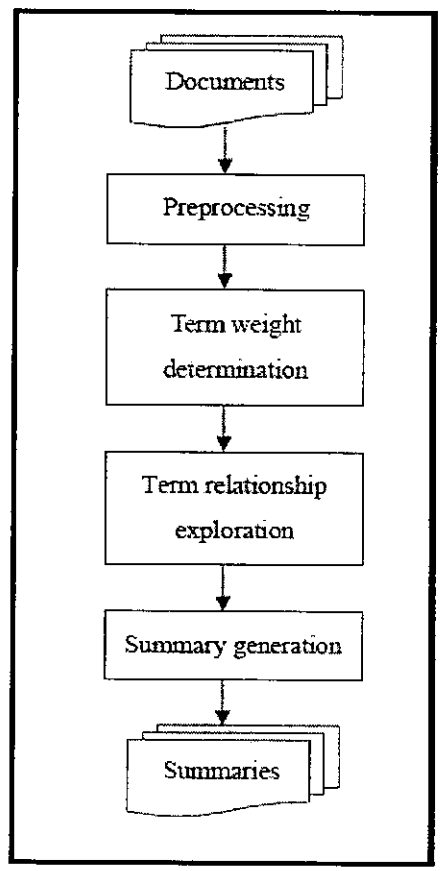
**Limitations:**

Linguistic methods are more difficult than statistical methods. They are more complicated and take more time to generate summaries. Such delays may create unnecessary annoyance for the users.

## 2.3 Proposed approach and its advantages:

We use a hybrid approach to text summarization. We use K-mixture connective-strength-based (KCS) approach to enhance the quality of document summary results. The K-mixture probabilistic model is used to determine the term weights. Term relationships are then investigated to develop the connective strength of nouns that manifests sentence semantics. Sentences with significant connective strength nouns are extracted to form the summary accordingly. Our proposed approach, as shown in Figure, basically consists of four main tasks: document preprocessing, term weight determination, term relationship exploration, and summary generation.

**Overview of Proposed approach**



**Advantages:**

- The approach combines the K-mixture term weighting scheme, which bases itself on a mathematical (probabilistic) ground, and the linguistic technique that explores term relationships by finding the connective strength of nouns that signifies term and sentence semantics.
- It allows choosing a low summary proportion without worrying much about the performance deterioration.
- It enhances the categorization performance.
- It generates a coherent summary of desired proportion in a short time.

## 2.4 Hardware Requirements:

| | |
|---|---|
| Processor | : Intel Processor IV |
| RAM | : 256 MB RAM |
| Hard disk | : 20GB |
| Monitor | : 15' Samtron color |
| Keyboard | : Standard 101/102 Keyboard |
| Mouse | : Optical mouse |

## 2.5 Software Requirements:

| | |
|---|---|
| Operating System | : Windows XP/2000 |
| Language used | : Java |

## 2.6 Software Overview:

| | |
|---|---|
| Front End | : Java |
| Back End | : Microsoft SQL Server 2000 |


Java:

Java is a robust, object oriented, multi-threaded, distributed, secure and platform independent language. It has a wide variety of packages to implement our requirement and number of classes and methods that can be utilized for easier and better programming.

The features of Java are as follows,

- Core java contains the concepts like Exception handling, Multithreading; Streams can be well utilized in the project environment.
- The Exception handling can be done with predefined exception and has provision for writing custom exception for our application.
- Garbage collection is done automatically, so that it is very secure in memory management.
- The user interface can be done with the Abstract Window tool kit and also Swing class. This has variety of classes for components and containers. We can make instance of these classes and this instances denotes particular object that can be utilized in our program.
- Event handling can be performed with Delegate Event model. The objects are assigned to the Listener that observe for event, when the event takes place, the corresponding methods to handle that event will be called by Listener which is in the form of interfaces and executed.
- This application makes use of ActionListener interface and the event click event gets handled by this. The separate method actionPerformed() method contains details about the response of event.

# *DETAILS OF METHODOLOGY EMPLOYED*

## 3. DETAILS OF METHODOLOGY EMPLOYED

Our approach, basically consists of four main tasks:

- Document preprocessing
- Term weight determination
- Term relationship exploration
- Summary generation

### 3.1. Preprocessing

This step is to preprocess collected documents to extract key terms. The sentences are tagged using the proposed tagger by [3] that utilizes conditional probability to attach the most likely part-of-speech to a term. After that, we then adopt the stemming procedure proposed by [4] to remove commoner morphological and inflexion endings from words in English. Different from traditional practice, we extract nouns and verbs to explore their relations that highlight sentence semantics.

#### 3.1.1 POS Tagging:

Part-of-speech tagging (POS tagging or POST), also called grammatical tagging or word-category disambiguation, is the process of marking up the words in a text as corresponding to a particular part of speech, based on both its definition, as well as its context—i.e., relationship with adjacent and related words in a phrase, sentence, or paragraph. A POS Tagger[5] is a piece of software that reads text in some language and assigns parts of speech to each word.

The software used in our project is a Java implementation of the log-linear part-of-speech tagger.

The tagger uses Penn Treebank tag set for tagging a document.

**Table 3.1.1.1: Penn Treebank Tagset**

| | |
|---|---|
| **CC** | Coordinating conjunction |
| **CD** | Cardinal Number |
| **DT** | Determiner |
| **EX** | Existential there |
| **FW** | Foreign Word |
| **IN** | Preposision or subordinating conjunction |
| **JJ** | Adjective |
| **JJR** | Adjective, comparative |
| **JJS** | Adjective, superlative |
| **LS** | List Item Marker |
| **MD** | Modal |
| **NN** | Noun, singular or mass |
| **NNP** | Proper Noun, singular |
| **NNPS** | Proper Noun, plural |
| **NNS** | Noun, plural |
| **PDT** | Predeterminer |
| **POS** | Possessive Ending |
| **PRP** | Personal Pronoun |
| **PRP$** | Possessive Pronoun |
| **RB** | Adverb |
| **RBR** | Adverb, comparative |
| **RBS** | Adverb, superlative |
| **RP** | Particle |
| **SYM** | Symbol |
| **TO** | To |
| **UH** | Interjection |
| **VB** | Verb, base form |
| **VBD** | Verb, past tense |
| **VBG** | Verb, gerund or persent participle |
| **VBN** | Verb, past participle |

| | |
|---|---|
| **VBP** | Verb, non-3rd person singular present |
| **VBZ** | Verb, 3rd person singular present |
| **WDT** | Wh-determiner |
| **WP** | Wh-pronoun |
| **WP$** | Possessive wh-pronoun |
| **WRB** | Wh-adverb |

### 3.1.2 Stemming:

Stemming is the process for reducing inflected (or sometimes derived) words to their stem, base or root form – generally a written word form. The stem need not be identical to the morphological root of the word; it is usually sufficient that related words map to the same stem, even if this stem is not in itself a valid root.

Frequently, the performance of an IR system will be improved if term groups such as this are conflated into a single term. This may be done by removal of the various suffixes -ED, -ING, -ION, IONS to leave the single term CONNECT. In addition, the suffix stripping process will reduce the total number of terms in the IR system, and hence reduce the size and complexity of the data in the system, which is always advantageous.

Stemming programs are commonly referred to as stemming algorithms or stemmers. The Java implementation of Porter's Stemmer[6] is used in our project.

## 3.2. Term weight determination

This step is to assign weights to each term extracted. As stated, we employ Katz's K-mixture probability model [7] to derive the probability of

term occurrence. In this model, the probability of a term $t$ with k occurrences in a document is denoted by

$$P_t(k) = (1-\alpha)\,\delta_{k,0} + (\alpha/B_1)\,(1-1/B_1)^{k-1}\,(1-\delta_{k,0}) \qquad (1)$$

where $\delta_{k,0} = 1$, if $k = 0$; otherwise, $\delta_{k,0} = 0$,

$\alpha$ is the probability of having at least one occurrence,

$B_1$ is the expected number of occurrences among the documents with at least one term occurrence.

Parameters in equation (1) can be estimated using the collected documents. Let $x$ be the proportion of documents that contain term $t$, and $y$ be the average extra terms per document among the documents term $t$ occurs. Then $x$ is used to estimate $\alpha$ and $y$ to estimate $B_1-1$. Equation (1) thus becomes

$$P_t(k) = (1-x)\,\delta_{k,0} + (x/(y+1))\,(y/(y+1))^{k-1}\,(1-\delta_{k,0}) \qquad (2)$$

Furthermore, $x$ and $y$ can be treated analogically to traditional TFIDF concepts. If TF is defined as $cf/N$ and IDF as $\log 2\ N/df$, then

$$y = (cf - df)/df = TF \times 2IDF - 1 \qquad (3)$$

$$x = df/N = TF/(y+1) \qquad (4)$$

where $cf$ is the total number of term $t$ occurrences among all documents,

$df$ is the number of documents term $t$ occurs,

$N$ is the total number of documents.

The K-mixture model was first proposed by [7] to simplify the negative binomial distribution. Equivalently, it could be described as the mixture of an infinite number of Poisson distributions. They had shown that the K-mixture model was more accurate than the Poisson distribution. Later on, text summarization and text categorization were viewed as a high-level data-cleaning tool in the knowledge discovery process. The K-mixture model is used in our summarizer because it has been proved to be efficient than the traditional measures like TF−IDF, entropy, mutual information, and $\chi 2$ statistics.

### 3.3. Term relationship exploration

Chen and Chen [8] proposed a text partition model to divide the text into plausible coherent discourse segments. The proposed approach took the noun-noun relations and noun-verb relations into consideration since they postulated that noun-verb relations were predicate-argument relations within the sentence level and noun-noun relations were associated on the discourse level. To segment the coherent discourse, they first evaluated the importance of a participating word (noun and verb) using the IDF weighting scheme. Then, the association norms of noun-verb and noun-noun pairs were calculated based on the importance of the words and the distance1 between each other. From the association norms, the connective strength of a noun was derived. Finally, the coherent score of sentences within a moving window was defined by averaging the connective strengths of nouns that were included in those sentences. A high score corresponded to a potential boundary of the discourse.

In this step, we mainly apply [8]'s approach to exploring term relationships between nouns and nouns, as well as nouns and verbs. The term weights, however, are determined using the K-mixture model from the previous step rather than using the simple IDF[1] method adopted in their approach. We also take the distance between two terms into account because relevant events are usually located closely in the same discourse. The distance of two terms is calculated by the difference of cardinal numbers that are assigned serially to each term in the sentences in a sentence. We then compute the association norms of noun-noun (SNN) and noun-verb (SNV) pairs penalized by their distance as follows.

$$\mathbf{SNN(N_i)} = \sum_{j} \frac{\mathbf{P(N_i)*P(N_j)}}{\mathbf{D(N_i,N_j)}}$$

$$\mathbf{SNV(N_i)} = \sum_{j} \frac{\mathbf{P(N_i)*P(V_j)}}{\mathbf{D(N_i,V_j)}}$$

Next, a noun's association norms (SNN and SNV) are summed up as in (7) to be its connective strength (CS), which denotes its semantic relationship significance.

$$CS(N_i) = SNN(N_i) + SNV(N_i) \qquad (7)$$

Finally, the sentence importance is indicated by the total CSs of nouns in that sentence.

### 3.4. Summary generation

Finally, the sentences with the top k scores are selected to form the summary. The parameter k is determined by the summary proportion of the original document. For example, if the original document contains 20 sentences and 50% proportion is desired, then we extract k = 10 sentences for the summary. The user can select the summary proportion required. If computational time and summary size matter, one may take a tradeoff between summary proportion and summary quality. That is, with KCS approach we can choose a low summary proportion to efficiently generate summaries without worrying too much about the performance deterioration.

## 3.5 Database Design:

**Table 3.5.1 : Term Weight Determination**

| Field | Type | Description |
|---|---|---|
| Term | Text | Extracted noun/verb from the document |
| Count | Int | Total no. of occurrences of the term |
| TWeight | Float | Term Weight |
| POS | Text | Part-of-Speech (Noun/Verb) |

**Table 3.5.2: Sentence Scores**

| Field | Type | Description |
|---|---|---|
| No | Int | Sentence no. |
| Score | Float | Score of the sentence |

**Table 3.5.3 : Top-K-Sentences**

| Field | Type | Description |
|---|---|---|
| No | Int | Sentence no. |

# *PERFORMANCE EVALUATION*

# 4. PERFORMANCE EVALUATION

## 4.1 Testing:

Software testing is the process used to assess the quality of computer software. Software testing is an empirical technical investigation conducted to provide stakeholders with information about the quality of the product or service. They examine and change the software engineering process itself to reduce the amount of faults that end up in defect rate. Software testing can be done by software testers.

Software testing is used in association with verification and validation:

1. Verification: Have we built the software right(i.e., does it match the specification)?
2. Validation: Have we built the right software(i.e., is this what the customer wants)?

Testing can involve some or all of the following factors.

- Business requirements
- Functional design requirements
- Technical design requirements
- Regularly requirements
- Programmer code
- System administration standards and restrictions
- Corporate standards
- Professional or trade association best practices

- Hardware configuration
- Cultural issues and language differences

Quality has three sets of factors – functionality, engineering and adaptability. These three sets of factors can be taken as dimensions in the software quality space. Each dimension may be broken down into its component factors and their considerations in detail.

| **Functionality (exterior quality)** | **Engineering (interior quality)** | **Adaptability (future quality)** |
|---|---|---|
| Correctness | Efficiency | Flexibility |
| Reliability | Testability | Reusability |
| Usability | Documentation | Maintainability |
| Integrity | Structure | |

**Table4.1.Typical Software Quality Factors**

Good testing provides measures for all relevant factors.Software tsting answers questions that development testing and code reviews can't.

- Does it really work as expected?
- Does it meet the user's requirements?
- Is it what the user expect?
- Do the users like it?
- Is it compatible with our other systems?
- How does it perform?
- How does it scale when more users are added?
- Which areas need more work?
- Is it ready for release?

### 4.1.1 Unit Testing:

A series of stand-alone tests are conducted during Unit Testing. Each test examines an individual component that is new or has been modified. A unit test is also called a module test because it tests the individual units of code that comprise the application. Each test validates a single module that, based on the technical design documents, was built to perform a certain task with the expectation that it will behave in a specific way or produce specific results.

Unit tests focus on functionality and reliability, Unit Testing is done in a test environment prior to system integration. If a defect is discovered during a unit test, the severity of the defect will dictate whether or not it will be fixed before the module is approved.

### 4.1.2 System Testing:

System Testing tests all components and modules that are new, changed,affected by a change, or needed to form the complete application. The system test may require involvement of other systems but this should be minimized as much as possible to reduce the risk of externally-induced problems.

Testing the interaction with other parts of the complete system comes in Integration Testing. The system test is validating and verifying the functional design specification and seeing how all the modules work together.

### 4.1.3 Integration Testing

Integration testing examines all the components and modules that are new, changed, affected by a change, or needed to form a complete system. Where system testing tries to minimize outside factors, integration testing requires involvement of other systems and interfaces with other applications, including those owned by an outside vendor, external partners, or the customer.

### 4.1.4 Implementation Details:

The main form is created with JFrame that contains other low level components. The 'summarize' menu item opens another new window. It allows the user to browse through the text files and select the required for summarization. They can also select the desired summary proportion (25% or 50% or 75%). The summaries will be generated in accordance with the selected summary proportion. During the process of summary generation, three intermediate files are created: tagged file, stemmed file and finally the summary. Many other details like total no. of lines in the original input file, no. of lines extracted to form the summary, the execution time are also displayed to the users. The view menu allows the users to view the files like input, tagged, stemmed or summary file.

## 4.2 Evaluation:

We examine the performance of our proposed approach, KCS. The summarization approaches under comparison include the commonly used tf-idf technique, K-mixture term weighting technique, linguistic technique with noun connective strengths (CS), and KCS.

In our study, we use the recall and precision as the performance measures, which are commonly employed in traditional information retrieval tasks. Precision is the proportion of returned documents that are targets, while recall is the proportion of target documents returned.

| Category $i$ | | Expert judgement | |
|---|---|---|---|
| | | **TRUE** | **FALSE** |
| Classifier judgement | **TRUE** | $TP_i$ | $FP_i$ |
| | **FALSE** | $FN_i$ | $TN_i$ |

Formally

$$P_i = \frac{TP_i}{TP_i + FP_i}$$

$$R_i = \frac{TP_i}{TP_i + FN_i}$$

where TP : True Positives

TN : True Negatives

FP : False Positives

FN : False Negatives

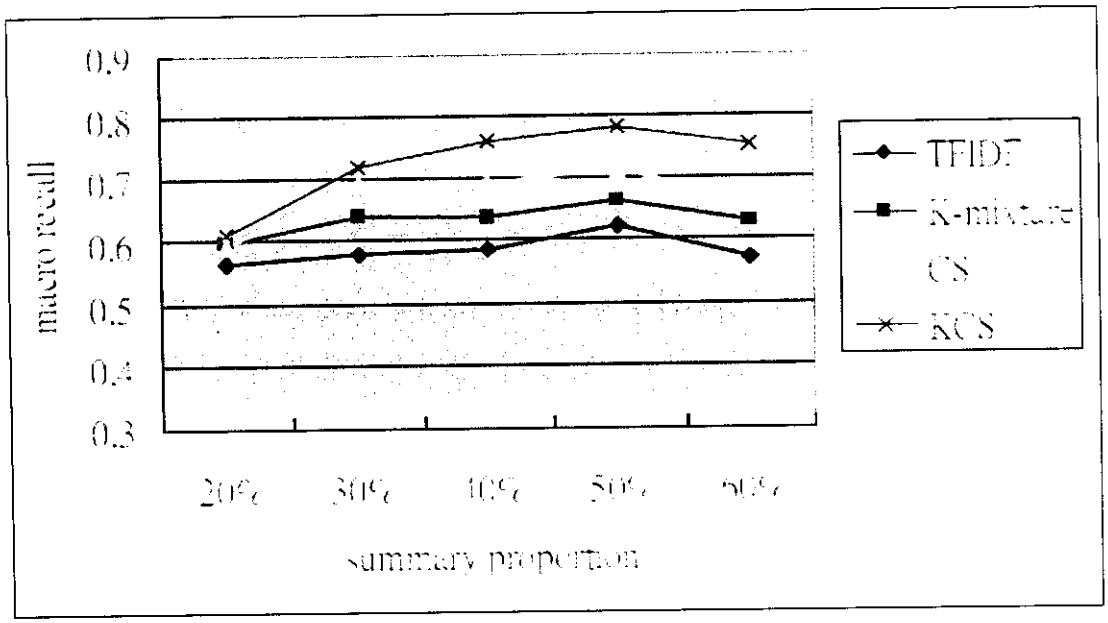



Figure 4.2.1 Macro-recall comparison under different approaches

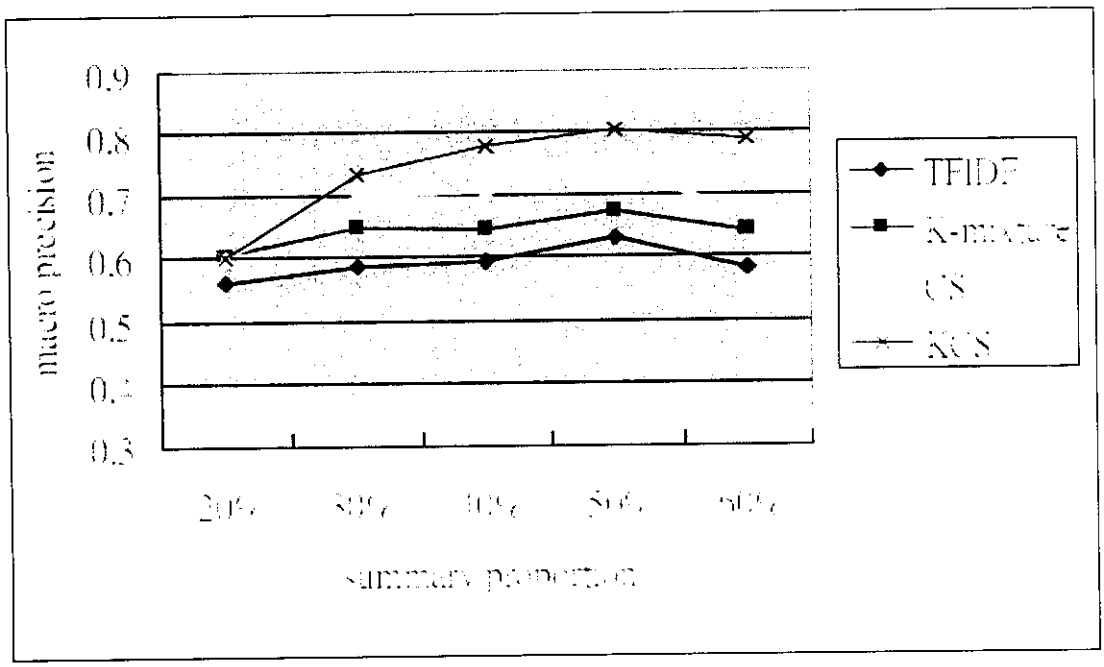



Figure 4.2.2 Macro-precision comparisons under different approaches

# *CONCLUSION*

## 5. Conclusion

The objective of this work was to develop a tool that uses a hybrid approach (KCS) to generate quality summaries automatically. The results obtained demonstrate this context. The method described is automated, inherently simple and tailored for its practical usage. It combines the K-mixture term weighing scheme, which bases itself on a mathematical (probabilistic) ground, and the linguistic technique that explores term relationships by finding the connective strength of nouns that signifies term and sentence semantics. It allows the user to decide the summary proportion required. Whatever be the summary proportion KCS performs better and allows choosing a lower summary proportion without worrying about the performance deterioration.

# *FUTURE ENHANCEMENTS*

# 6. FUTURE ENHANCEMENTS

The tool developed generates summaries of text documents and presents it to the user. Some other enhancements that can be implemented are to

- generate summaries of different types of documents like pdf documents, word documents, etc.,
- generate summaries of multiple documents.
- develop a more intuitive graphical user interface.

To generate summaries of pdf and word documents, it is required to preprocess the input files and convert them to standard text format before summarization.

# *APPENDICES*

# 7. APPENDIX

## 7.1 Source Code:

### Term Weight Determination:

### DataStore.java

```
package TWD;

import java.sql.*;
import java.io.*;
import java.lang.*;

public class DataStore
{
  public int co=0;
  Connection con,con1,con2;

  public void createtable()
  {
    try
    {
     con=DriverManager.getConnection("jdbc:odbc:MyDSN");
       String str="CREATE TABLE summ (Name VARCHAR(25),Count
INTEGER,Tweight FLOAT,Pos VARCHAR(10),Doccount
INTEGER,Notify INTEGER,Kv INTEGER)";
       Statement st=con.createStatement();
       st.execute(str);
       con.close();
    }
    catch(Exception e)
    {
       System.out.println("Error" + e);
    }
  }
```

```
  public void setnotify(String str)
 {
  try
  {
     con=DriverManager.getConnection("jdbc:odbc:MyDSN");
     String str1="UPDATE summ SET Notify=1 WHERE Name=?";
     PreparedStatement ps=con.prepareStatement(str1);
     ps.setString(1,str);
     ps.executeUpdate();
     con.close();
  }
  catch(Exception e)
  {
     System.out.println("Error" + e);
  }
}

  public void resetnotify()
{
  try
  {
     con=DriverManager.getConnection("jdbc:odbc:MyDSN");
     con1=DriverManager.getConnection("jdbc:odbc:MyDSN");
     String str1="SELECT * FROM summ";
     String str2="UPDATE summ SET Notify=0 WHERE Name=?";
     Statement ps=con.createStatement();
     ResultSet rs=ps.executeQuery(str1);
     while(rs.next())
     {
           String name=rs.getString("Name");
           PreparedStatement ps1=con1.prepareStatement(str2);
           ps1.setString(1,name);
           ps1.executeUpdate();
     }
     con.close();
     con1.close();
}
catch(Exception e)
{
```

```
        System.out.println("Error" + e);
    }
  }

}
```

**ScanFile.java**

```
package TWD;

import java.sql.*;
import java.io.*;
import java.lang.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class ScanFile
{
   FileInputStream fin;
   String str="";
        String filename,fn1;
   DataStore db=new DataStore();
   int i=0;boolean flag=false;
   boolean avail=false;
   public ScanFile()
   {}

   public void readfile(String fname,String fn)
   {
        filename=fname;
fn1=fn;
        //final JFrame f=new JFrame("Term Weight Determination");
        System.out.println("\nScanning file " + fname + "...");
      System.out.println("Extracting nouns and verbs from the file");
        //new Thread(){
//      public void run(){
     try{

        fin=new FileInputStream(filename);
```

```
//      ProgressMonitorInputStream pm=new
ProgressMonitorInputStream(f,"Extracting nouns and verbs",fin);
    do
    {
      i=fin.read();
       if(i != -1)
       {
         if((char)i == '/')
         {
            i=fin.read();
            if((char)i == 'n')
        {
              avail=db.check(str);
              if(avail == true)
                   if(filename.compareTo(fn1)==0)
                   db.updatetable(str,1);
                   else
                   db.updatetable(str,0);
              else
                   if(filename.compareTo(fn1)==0)
                   db.inserttable(str,1,0.0,"N",0,0,1);
                   else
                   db.inserttable(str,1,0.0,"N",0,0,0);
            }
            if((char)i == 'v')
        {
           avail=db.check(str);
              if(avail == true)
                   if(filename.compareTo(fn1)==0)
                   db.updatetable(str,1);
                   else
                   db.updatetable(str,0);
              else
                   if(filename.compareTo(fn1)==0)
                   db.inserttable(str,1,0.0,"V",0,0,1);
                   else
                   db.inserttable(str,1,0.0,"V",0,0,0);
            }
            db.updatedoccount(str);
            db.setnotify(str);
```

```
                while((char)i != ' ')
            {
                        i=fin.read();
                    if(i == -1)
                            break;
                }
                str="";
            }
            else
            {
                str= str + (char)i;
            }
          }
       }while(i != -1);
         db.resetnotify();
         flag=false;
    }
    catch(Exception e)
   {
    System.out.println("Error" + e);
   }
}
}.start();
f.dispose();
f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
f.setVisible(true);

 }
}
```

**TermWeight.java:**

```
package TWD;

import java.sql.*;
import java.io.*;
import java.lang.*;

public class TermWeight
{
```

```
Connection con,con1;

  public void findtw(int N)
{
  float x,y,tw;
  int k,del,cf,df,subt;
  String ss;
  try
  {
     con=DriverManager.getConnection("jdbc:odbc:MyDSN");
     con1=DriverManager.getConnection("jdbc:odbc:MyDSN");
     String str1="SELECT * FROM summ";
     String str2="UPDATE summ SET Tweight=? WHERE Name=?";
     Statement st=con.createStatement();
     ResultSet rs=st.executeQuery(str1);
     System.out.println("Term weights determination");
     while(rs.next())
     {
       ss=rs.getString("Name");
       cf=rs.getInt("Count");
       df=rs.getInt("Doccount");
       x=(float)df/N;
       y=((float)cf-(float)df)/(float)df;
       k=rs.getInt("Kv");
       if(k==0)
            del=1;
       else
            del=0;
       tw=(float)((1-x) * del) + (float)(x/(y+1)) *
(float)Math.pow((float)(y/(y+1)) ,cf-1) * (1-del);
       PreparedStatement ps1=con1.prepareStatement(str2);
       ps1.setFloat(1,tw);
       ps1.setString(2,ss);
       ps1.executeUpdate();
     }
     con.close();
     con1.close();
     System.out.println("Term weights determination successful");
  }
  catch(Exception e)
```

```
    {
        System.out.println("Error" + e);
    }
  }
}
```

**Term Relationship Exploration:**

**Scorer.java**

```
package Scoring;

import java.io.*;
import java.lang.*;
import java.sql.*;

public class Scorer
{
 FileInputStream fin;
 Connection con,con1,con2,con3;
 String nv[]=new String[50];
 int cardinal[]=new int[50];
 float tw[]=new float[50];
 String noun[]=new String[25];
 float CS[]=new float[25];
 float senscore=(float)0.0;
 int i=0,j=0,k=0;
 int f=0,flag=0;
 String str="";
 int senno=0;
 public static int no_sen=0;

 public void score(String fname)
 {
       try
       {
       con=DriverManager.getConnection("jdbc:odbc:MyDSN");
       String str1="SELECT * FROM summ WHERE Name=?";
       PreparedStatement ps1=con.prepareStatement(str1);
       fin=new FileInputStream(fname);
```

```
        create();
        System.out.println("Exploring the term relationships..");
        System.out.println("Determination of association norms of noun-noun
and noun-verb pairs...");
        System.out.println("calculation the connective strength of nouns in
each sentence....");
        System.out.println("Scoring the sentences.....");
        while(f!=-1)
        {
          f=fin.read();
          if(f==-1)
               break;
          if((char)f=='/')
          {
               f=fin.read();
               if((char)f=='n')
               {
                 nv[i]=str;
                cardinal[i]=i+1;
                 noun[j]=str;
                 ps1.setString(1,str);
                 ResultSet rs1=ps1.executeQuery();
                 rs1.next();
                 tw[i]=rs1.getFloat("Tweight");
                i++;
                 j++;
               }
               if((char)f=='v')
               {
                 nv[i]=str;
                 ps1.setString(1,str);
                 ResultSet rs2=ps1.executeQuery();
                 rs2.next();
                 tw[i]=rs2.getFloat("Tweight");
                 cardinal[i]=i+1;
                 i++;
               }
               while((char)f!=' ')
               {
                 f=fin.read();
```

```
        System.out.println("Scoring successful.....");
        FileOutputStream fout=new FileOutputStream("sen_len.dat");
        DataOutputStream dos=new DataOutputStream(fout);
        dos.writeInt(no_sen);
    }

    public void findCS()
    {
        int x=0,y=0,z=0,a=0;
        float SNV=(float)0.0;
        try
        {
          for(x=0;x<j;x++)
          {
                for(a=0;a<i;a++)
                  if(noun[x].compareTo(nv[a])==0)
                  {
                        z=a;
                        break;
                  }
                for(y=0;y<i;y++)
                {
                        if(Math.max(cardinal[z],cardinal[y])-
Math.min(cardinal[z],cardinal[y])==0)
                        SNV+=0;
                        else

        SNV+=(float)(tw[z]*tw[y])/(float)(Math.max(cardinal[z],cardinal[y])-
(float)Math.min(cardinal[z],cardinal[y]));
                }
                CS[x]=SNV;
           SNV=(float)0.0;
          }
        }
        catch(Exception e)
        {
                System.out.println("Error" +e);
        }
    }
```

```
  public void create()
  {
        try
        {
          droptable();
          con1=DriverManager.getConnection("jdbc:odbc:MyDSN2");
          String str1="CREATE TABLE sentencescore(No INTEGER,Score
FLOAT)";
          Statement st=con1.createStatement();
          st.execute(str1);
          con1.close();
        }
        catch(Exception e)
        {
         System.out.println("Error" +e);
        }
  }

  public void store(int senno,float sc)
  {
        try
        {
          con2=DriverManager.getConnection("jdbc:odbc:MyDSN2");
          String str1="INSERT INTO sentencescore(No,Score)
VALUES(?,?)";
          PreparedStatement ps=con2.prepareStatement(str1);
          ps.setInt(1,senno);
          ps.setFloat(2,sc);
          ps.executeUpdate();
          con2.close();
        }
        catch(Exception e)
        {
          System.out.println("Error" + e);
        }
  }
```

**Summary Generation:**

**Gen_summ.java**

```
package summarize;

import java.io.*;
import java.lang.*;
import java.sql.*;

public class gen_summ
{
  int percent,filesize;
  Connection con1,con2,con3;
  public void generate(int per)
  {
    try
    {
      DataInputStream dis=new DataInputStream(new
FileInputStream("sen_len.dat"));
      filesize=dis.readInt();
      percent=(per*filesize)/100;
      create();
      findmax();
      summary();
    }
    catch(Exception e)
    {
      System.out.println("Error" + e);
    }
  }
  public void summary()
  {
    int flag=0,count1=0,temp;
    String str="";
    try
    {
      con1=DriverManager.getConnection("jdbc:odbc:MyDSN3");
      String str1="SELECT * FROM summ_gen ORDER BY No ASC";
```

```
Statement st=con1.createStatement();
ResultSet rs=st.executeQuery(str1);
rs.next();
temp=rs.getInt("No");
FileInputStream fin=new FileInputStream("tagged.txt");
FileOutputStream fout=new FileOutputStream("summary.txt");
int c=0;
System.out.println("\n\t\tSummary\n");
while(c!=-1)
{
  c=fin.read();
  if(c==-1)
    break;
  if((char)c=='/')
  {
  while((char)c!=' ')
  {
    c=fin.read();
    if(c==-1)
    {
       flag=1;
       break;
    }
    if((char)c=='\n')
    {
       count1++;
       if(count1==temp)
       {
         str+=' ';
         System.out.print(str);
         for(int z=0;z<str.length();z++)
              if(str.charAt(z)!='\b')
              fout.write(str.charAt(z));
         if(rs.next())
              temp=rs.getInt("No");
     }
     str="";
     break;
   }
  }
```

```
        }
        if(flag==1)
          break;
        if((char)c!='\n')
        {
            if((char)c=='.' || (char)c==',')
              str+='\b';
            str+=(char)c;
        }
      }
     con1.close();
    }
    catch(Exception e)
    {
      System.out.println("Error" + e);
    }
  }
  public void create()
  {
      try
      {
        droptable();
        con1=DriverManager.getConnection("jdbc:odbc:MyDSN3");
        String str1="CREATE TABLE summ_gen(No INTEGER)";
        Statement st=con1.createStatement();
        st.execute(str1);
        con1.close();
      }
      catch(Exception e)
      {
       System.out.println("Error" +e);
      }
  }
public void store(int senno)
  {
      try
      {
        con2=DriverManager.getConnection("jdbc:odbc:MyDSN3");
        String str1="INSERT INTO summ_gen(No) VALUES(?)";
        PreparedStatement ps=con2.prepareStatement(str1);
```

```
      ps.setInt(1,senno);
      ps.executeUpdate();
      con2.close();
    }
    catch(Exception e)
    {
      System.out.println("Error" + e);
    }
}
public void findmax()
{
  int c=0;
  try
  {
    con1=DriverManager.getConnection("jdbc:odbc:MyDSN2");
    String str1="SELECT * FROM sentencescore order by Score desc";
    PreparedStatement ps=con1.prepareStatement(str1);
    ResultSet rs=ps.executeQuery();
    while(rs.next())
    {
      if(c<percent)
      {
          store(rs.getInt("No"));
          c++;
     }
      else
          break;
    }
    con1.close();
  }
  catch(Exception e)
  {
    System.out.println("Error"+e);
  }
}
```

**Main Frame:**

**Summarizer.java:**

```
import java.util.List;
import java.io.*;
import edu.stanford.nlp.ling.Sentence;
import edu.stanford.nlp.ling.TaggedWord;
import edu.stanford.nlp.ling.HasWord;
import edu.stanford.nlp.tagger.maxent.MaxentTagger;
import java.sql.*;
import java.lang.*;
import TWD.*;
import Stemmer.*;
import Scoring.*;
import summarize.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.*;
import javax.swing.event.*;
import java.util.Date;

class dlgbox extends JDialog implements ActionListener
{
public dlgbox(JFrame parent,String title,String message)
{
super(parent,title,true);
JPanel messagepane=new JPanel();
messagepane.add(new JLabel(message));
getContentPane().add(messagepane);
JPanel buttonpane=new JPanel();
JButton button=new JButton("OK");
buttonpane.add(button);
button.addActionListener(this);
getContentPane().add(buttonpane,BorderLayout.SOUTH);
setDefaultCloseOperation(DISPOSE_ON_CLOSE);
pack();
setVisible(true);
}
```

```
public void actionPerformed(ActionEvent et)
{
setVisible(false);
dispose();
}
}

public class TaggerDemo {
static String filename;
static JTextField txtfilename,tf2;
static String content;
static GridBagConstraints c;
static Container pane,pane1;
static JFrame
frame,dlgframe,progressframe,frameMain,stemframe,tagframe;
static JTextArea tasummary,tastat;
static JComboBox comboSummary;
static Dimension d1,d,d2;
static int propor;
static JProgressBar pb;
static String bri,message;
static Date starttime,endtime;
static JButton butopendir;
static int flag;
static JMenu homemenu,viewmenu;
static JMenuItem
summarizeitem,aboutitem,exititem,inputfileitem,summaryitem,taggeditem,st
emmeditem;

private static void butopendirActionPerformed(ActionEvent et)
{
      JFileChooser jfc = new JFileChooser(".");
      jfc.setDialogTitle("Select directory ...");
      jfc.setFileSelectionMode(JFileChooser.FILES_ONLY);
      if (jfc.showOpenDialog(butopendir) ==
JFileChooser.APPROVE_OPTION) {
        try {
          txtfilename.setText(jfc.getSelectedFile().getCanonicalPath());
        }
       }
```

```
}

public static void main(String[] args) throws Exception {
        frame=new JFrame("Auto-summarization");
        frameMain=new JFrame("Auto-summarization");
        pane=frame.getContentPane();
        pane1=frameMain.getContentPane();
        d=new Dimension(250,30);
        pane.setLayout(new GridBagLayout());
        pane1.setLayout(new GridBagLayout());
         c = new GridBagConstraints();
        JMenuBar mbar=new JMenuBar();
        mbar.setBorder(new BevelBorder(BevelBorder.RAISED));
        viewmenu.add(summaryitem);
        taggeditem=new JMenuItem("Tagged file");
        viewmenu.add(taggeditem);
        stemmeditem=new JMenuItem("Stemmed file");
        viewmenu.add(stemmeditem);
        mbar.add(homemenu);
        mbar.add(viewmenu);
        c.fill =        GridBagConstraints.HORIZONTAL;
        c.weightx = 0.5;
        c.gridx = 0;
        c.insets=new Insets(10,10,10,10);
        c.gridy = 0;
        inputfileitem.setEnabled(false);
        summaryitem.setEnabled(false);
        taggeditem.setEnabled(false);
        stemmeditem.setEnabled(false);
        pane1.add(mbar,c);
        pane1.setSize(600,400);

        JLabel lb=new JLabel("");
        lb.setPreferredSize(d2);
        pane1.add(lb,c);

        exititem.addActionListener(new ActionListener()
        {
          public void actionPerformed(ActionEvent e)
          {
```

```
            System.exit(0);
        }
      });
      inputfileitem.addActionListener(new ActionListener()
      {
        public void actionPerformed(ActionEvent e)
        {
            try
            {

      Runtime.getRuntime().exec("C:/WINDOWS/system32/notepad.exe
      "+filename);
            }
            catch(Exception exc)
            {
               System.out.println(exc);
            }
        }
      });


      summarizeitem.addActionListener(new ActionListener()
      {
        public void actionPerformed(ActionEvent e)
        {
            JLabel lb2 = new JLabel("Summary Proportion");
            c.fill = GridBagConstraints.HORIZONTAL;
            c.insets=new Insets(10,10,10,10);
            c.weightx = 0.5;
            c.gridx = 0;
            c.gridy = 1;
            pane.add(lb2, c);
            comboSummary = new JComboBox();
            c.fill = GridBagConstraints.HORIZONTAL;
            c.insets=new Insets(10,10,10,10);
            comboSummary.setToolTipText("Select the desired summary
proportion");
            comboSummary.addItem("25%");
            c.weightx = 0.5;
            c.gridx = 1;
```

```
c.gridy = 1;
pane.add(comboSummary, c);
comboSummary.addActionListener(new ActionListener()
{
        public void actionPerformed(ActionEvent e)
        {
                propor=comboSummary.getSelectedIndex();
        }
});

butopendir=new JButton("Browse");
pane.add(butopendir, c);
butopendir.addActionListener(new ActionListener()
{
     public void actionPerformed(ActionEvent evt)
    {
        butopendirActionPerformed(evt);

    }
 });
JButton b1 = new JButton("Summarize");
pane.add(b1, c);
b1.addActionListener(new ActionListener()
{
   public void actionPerformed(ActionEvent e)
   {
     try{
        content=" ";
        bri=" ";
        starttime=new Date();
        filename=txtfilename.getText();
        flag=1;
        if(filename.compareTo("")==0)
        {
        dlgbox dlg=new dlgbox(new JFrame(),"Error","Enter a
valid filename");
        flag=0;
        }
        FileInputStream fin=new FileInputStream(filename);
```

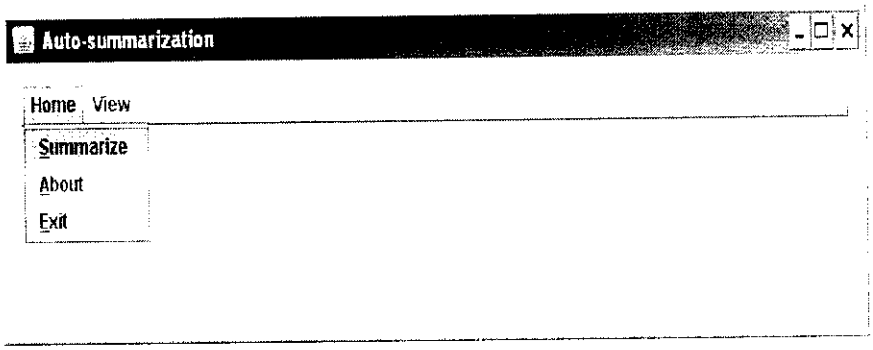
```
                    System.out.println("\nPreprocessing ...\n\nLoading POS
tagger");
                    String str;
                    FileOutputStream fout=new
FileOutputStream("tagged.txt");
                    System.out.println("\nReading POS tagger from models
... \n");
                     MaxentTagger tagger = new MaxentTagger("left3words-
            wsj-0-18.tagger");
                      @SuppressWarnings("unchecked")
                     List<Sentence<? extends HasWord>> sentences =
MaxentTagger.tokenizeText(new BufferedReader(new
FileReader(filename)));
                     for (Sentence<? extends HasWord> sentence :
sentences) {
                     Sentence<TaggedWord> tSentence =
MaxentTagger.tagSentence(sentence);
                      str=tSentence.toString(false);
                   for(int i=0;i<str.length();i++)
                     fout.write(str.charAt(i));
                     fout.write('\n');
                     System.out.println(str);
            }
                     System.out.println("Tagging successful\n");
                     message="Tagging the document..";
                     refresh();
                     System.out.println("Stemming in process...");
                     char[] w = new char[501];
                      Stemmer s = new Stemmer();
                 try
                 {
                   FileInputStream in = new FileInputStream("tagged.txt");
                   FileOutputStream out1=new
 FileOutputStream("stemmed.txt");
                   try
                   {
                    while(true)

                    {  int ch = in.read();
                      if (Character.isLetter((char) ch))
```

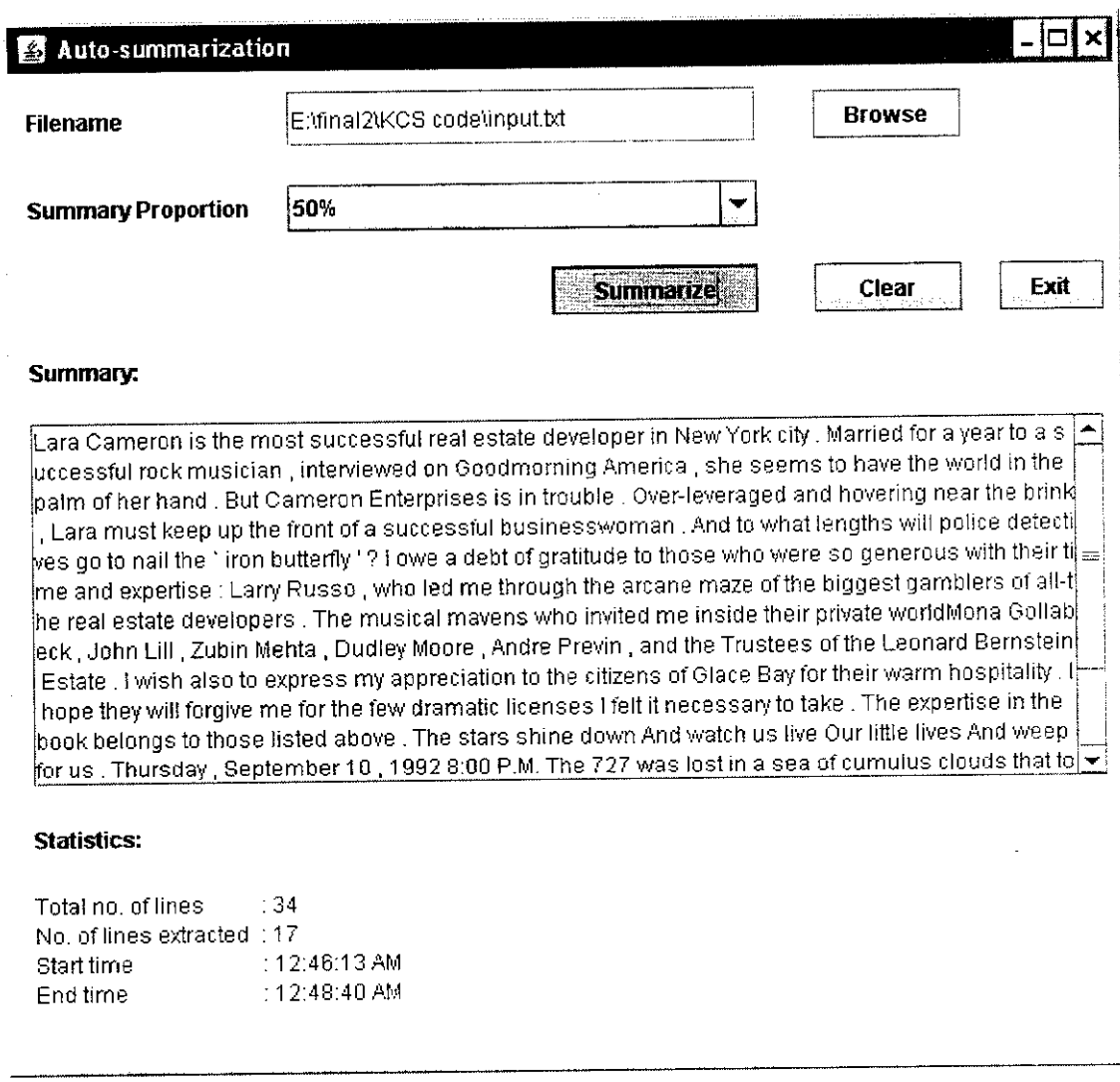
```
            {
              int j = 0;
              while(true)
              {  ch = Character.toLowerCase((char) ch);
                w[j] = (char) ch;
                if (j < 500) j++;
                ch = in.read();
                if (!Character.isLetter((char) ch))
                {
                  for (int c = 0; c < j; c++) s.add(w[c]);
                        s.stem();
                  {  String u;
                  char smp[]=new char[30];
                  int z;
                  u = s.toString();
                 smp=u.toCharArray();
                 for(z=0;z<u.length();z++)
                 out1.write(smp[z]);
                 z=0;                    }
                  break;   }  }
              }
             if (ch < 0) break;
            out1.write((char)ch);

          }
          System.out.println("Stemming successful");
          message="Stemming in progress..";
          refresh();
      }          }
  try
    {
          DataStore db=new DataStore();
          ScanFile sf=new ScanFile();
          TermWeight tt=new TermWeight();
          db.droptable();
          db.createtable();
          int i=0;
                sf.readfile("stemmed.txt","stemmed.txt");
                tt.findtw(1);
                Scorer sco=new Scorer();
```

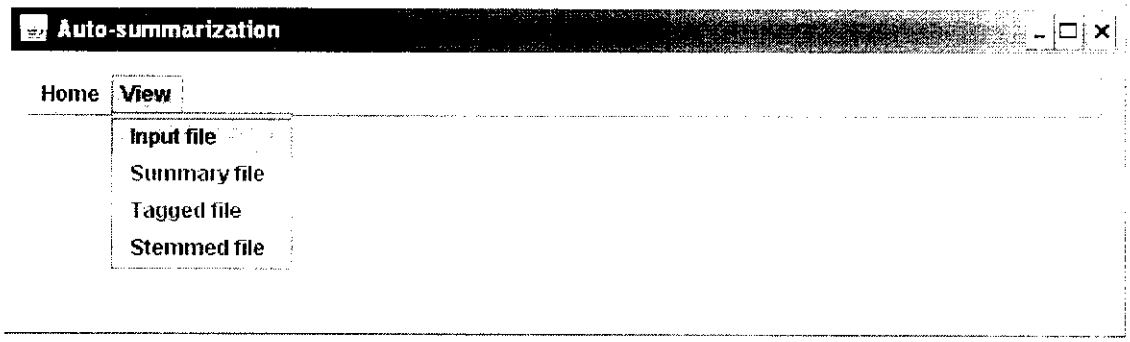
```
                sco.score("stemmed.txt");
                gen_summ g=new gen_summ();
            if(propor==0)
                g.generate(25);
            else if(propor==1)
                g.generate(50);
            else
                g.generate(75);
            int ch=0;
            FileInputStream fin2=new
        FileInputStream("summary.txt");
            while(ch!=-1)
            {
                ch=fin2.read();
                content+=(char)ch;
            }}}
         endtime=new Date();
        try{
        DataInputStream dis=new DataInputStream(new
FileInputStream("sen_len.dat"));
        int filesize=dis.readInt();
        int per;
        if(propor==0)
            per=25;
        else if(propor==1)
            per=50;
        else
            per=75;
            int percent=(per*filesize)/100;
        long tim1=starttime.getTime();
        long tim2=endtime.getTime();
        long tim=(tim2-tim1)/1000;
        long min=0,se=0;
        if(tim>=60)
        {
            min=tim/60;
            se=tim%60;
        }
        if(flag!=0)
```

```
        bri="Total no. of lines : " + filesize + "\nNo. of lines extracted :
        " + percent + "\nStart time : " + starttime + "\nEnd time : " +
        endtime ;//+ "\nElapsed time : " +  min  + "min"  + se +"sec";
        }
        catch(Exception e10){}
        refresh();
    }
});
        c.gridwidth=4;
        c.gridheight=2;
        pane.add(sp1,c);

        JLabel lb4=new JLabel("Statistics:");
        pane.add(lb4, c);

        tastat=new JTextArea(bri,6,4);
        tastat.setLineWrap(true);
        pane.add(tastat, c);
        inputfileitem.setEnabled(true);
        summaryitem.setEnabled(true);
        taggeditem.setEnabled(true);
        stemmeditem.setEnabled(true);
        frame.setLocation(100,100);
        frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOS
 E);
        frame.pack();
        frame.setVisible(true);
    }
});
        frameMain.setDefaultCloseOperation(JFrame.EXIT_ON_CLO
 SE);
        frameMain.pack();
        frameMain.setVisible(true);
    }
}
```

## 7.2 Screen Shots:





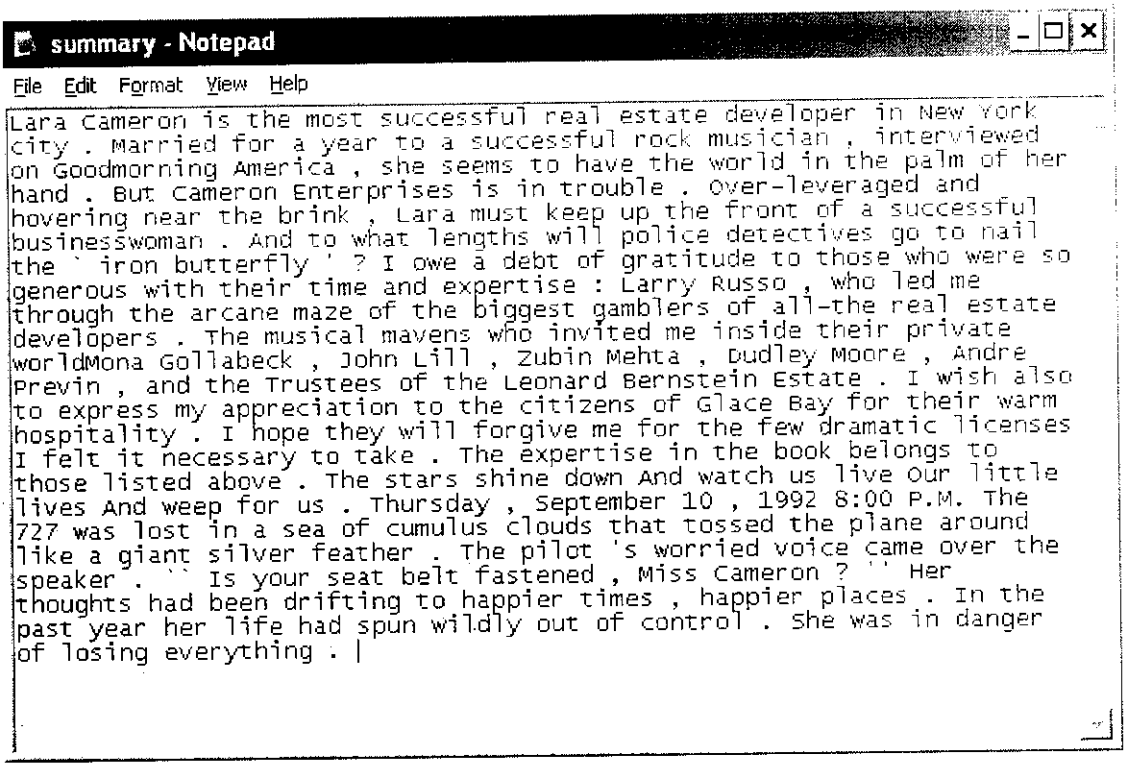

input.txt

input - Notepad

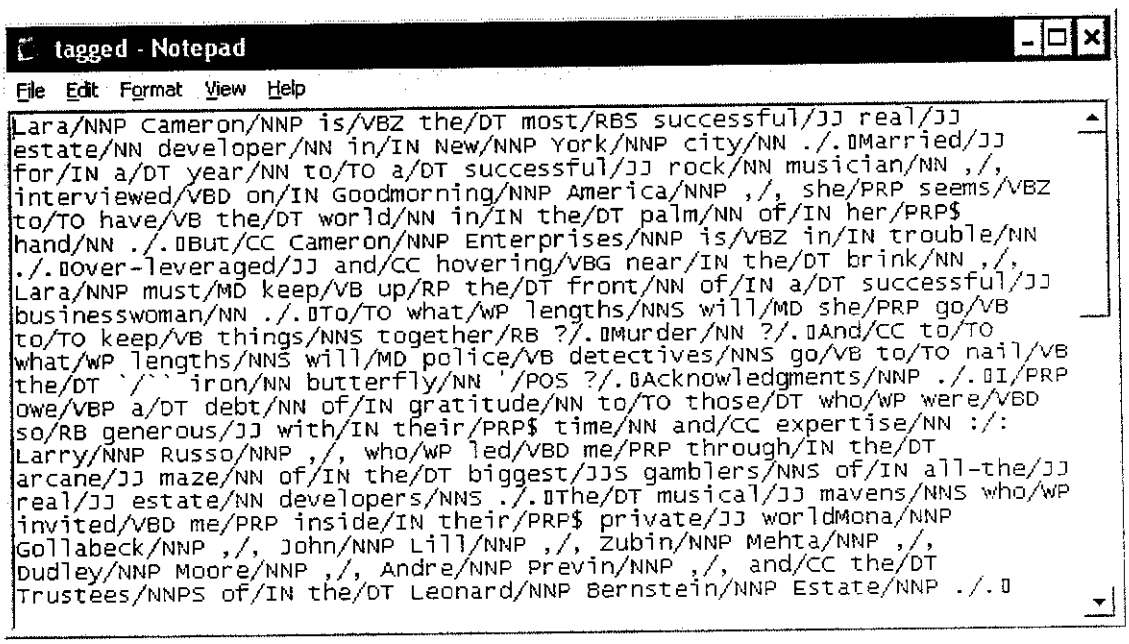
File Edit Format View Help

Lara Cameron is the most successful real estate developer in New York city. Married for a year to a successful rock musician,interviewed on Goodmorning America, she seems to have the world in the palm of her hand. But Cameron Enterprises is in trouble. Over-leveraged and hovering near the brink, Lara must keep up the front of a successful businesswoman. To what lengths will she go to keep things together? Murder? And to what lengths will police detectives go to nail the 'iron butterfly'? Acknowledgments. I owe a debt of gratitude to those who were so generous with their time and expertise: Larry Russo, who led me through the arcane maze of the biggest gamblers of all-the real estate developers. The musical mavens who invited me inside their private worldMonaGollabeck, John Lill, Zubin Mehta, Dudley Moore, Andre Previn, and the Trustees of the Leonard Bernstein Estate. I wish also to express my appreciation to the citizens of Glace Bay for their warm hospitality. I hope they will forgive me for the few dramatic licenses I felt it necessary to take. The expertise in the book belongs to those listed above. Any errors are mine.The stars shine down And watch us live Our little lives And weep for us. Chapter One. Thursday, September 10, 1992 8:00 P.M. The 727 was lost in a sea of cumulus clouds that tossed the plane around like a giant silver feather. The pilot's worried voice came over the speaker. "Is your seat belt fastened, Miss Cameron?" There was no response.
"Miss Cameron... Miss Cameron She was| shaken out of a deep reverie.
"Yes." Her thoughts had been drifting to happier times, happier
places.
"Are you all right? We should be out of this storm soon."
"I'm fine, Roger."
Maybe we'll get lucky and crash, Lara Cameron thought. It would be a
fitting end. Somewhere, somehow, it had all gone wrong.
It's the

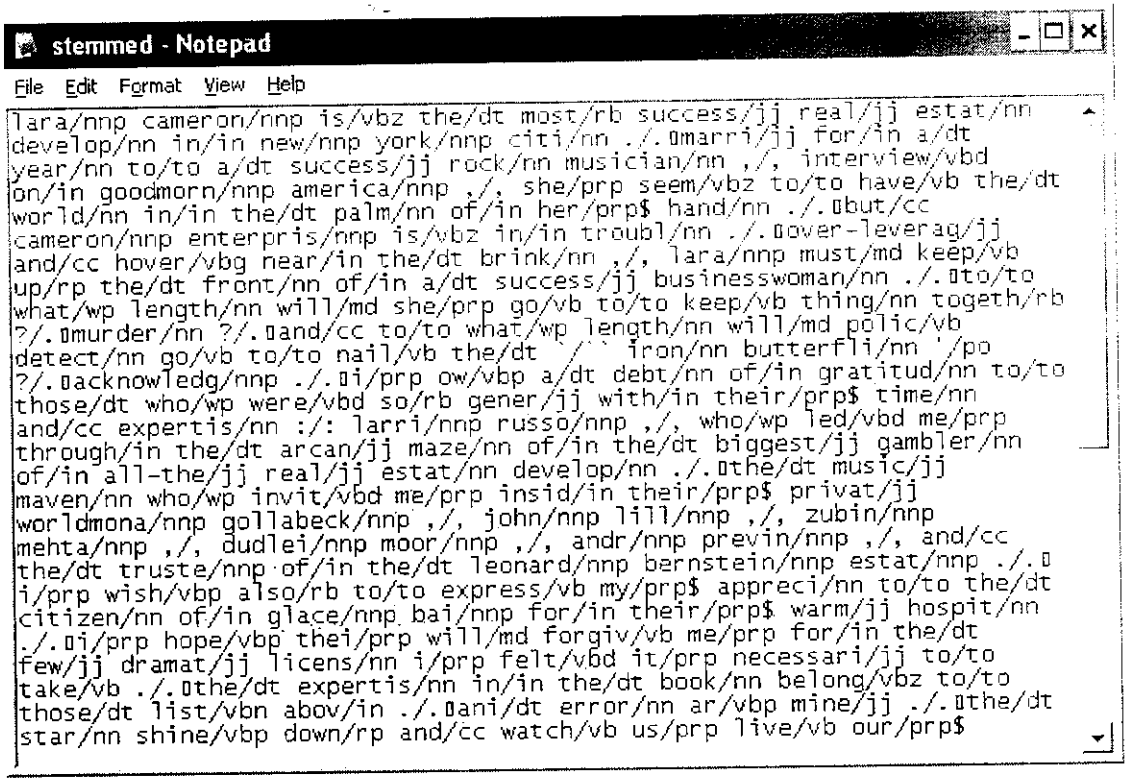
summary.txt




tagged.txt



stemmed.txt


stemmed - Notepad
File Edit Format View Help
lara/nnp cameron/nnp is/vbz the/dt most/rb success/jj real/jj estat/nn
develop/nn in/in new/nnp york/nnp citi/nn ./. marri/jj for/in a/dt
year/nn to/to a/dt success/jj rock/nn musician/nn ,/, interview/vbd
on/in goodmorn/nnp america/nnp ,/, she/prp seem/vbz to/to have/vb the/dt
world/nn in/in the/dt palm/nn of/in her/prp$ hand/nn ./. but/cc
cameron/nnp enterpris/nnp is/vbz in/in troubl/nn ./. over-leverag/jj
and/cc hover/vbg near/in the/dt brink/nn ,/, lara/nnp must/md keep/vb
up/rp the/dt front/nn of/in a/dt success/jj businesswoman/nn ./. to/to
what/wp length/nn will/md she/prp go/vb to/to keep/vb thing/nn togeth/rb
?/. murder/nn ?/. and/cc to/to what/wp length/nn will/md polic/vb
detect/nn go/vb to/to nail/vb the/dt `/`` iron/nn butterfli/nn '/po
?/. acknowledg/nnp ./. i/prp ow/vbp a/dt debt/nn of/in gratitud/nn to/to
those/dt who/wp were/vbd so/rb gener/jj with/in their/prp$ time/nn
and/cc expertis/nn :/: larri/nnp russo/nnp ,/, who/wp led/vbd me/prp
through/in the/dt arcan/jj maze/nn of/in the/dt biggest/jj gambler/nn
of/in all-the/jj real/jj estat/nn develop/nn ./. the/dt music/jj
maven/nn who/wp invit/vbd me/prp insid/in their/prp$ privat/jj
worldmona/nnp gollabeck/nnp ,/, john/nnp lill/nnp ,/, zubin/nnp
mehta/nnp ,/, dudlei/nnp moor/nnp ,/, andr/nnp previn/nnp ,/, and/cc
the/dt truste/nnp of/in the/dt leonard/nnp bernstein/nnp estat/nnp ./.
i/prp wish/vbp also/rb to/to express/vb my/prp$ appreci/nn to/to the/dt
citizen/nn of/in glace/nnp bai/nnp for/in their/prp$ warm/jj hospit/nn
./. i/prp hope/vbp thei/prp will/md forgiv/vb me/prp for/in the/dt
few/jj dramat/jj licens/nn i/prp felt/vbd it/prp necessari/jj to/to
take/vb ./. the/dt expertis/nn in/in the/dt book/nn belong/vbz to/to
those/dt list/vbn abov/in ./. ani/dt error/nn ar/vbp mine/jj ./. the/dt
star/nn shine/vbp down/rp and/cc watch/vb us/prp live/vb our/prp$

# *REFERENCES*

## 8. REFERENCES:


[1] Te-Min Chang Wen-Feng Hsiao, "A hybrid approach to automatic text summarization", Proceedings of 8th IEEE International Conference, Computer and Information Technology, July 2008, (CIT 2008), pp. 65-70

[2] H. Zhuge, The Knowledge Grid. Singapore: World Scientific Publishing Co., 2004.

[3] Toutanova K., Klein D., Manning C., and Singer Y. (2003), "Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network," Proceedings of Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2003), pp. 252-259.

[4] Krovetz R (1993). "Viewing Morphology as an Inference Process," Proceedings of the 16th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval(SIGIR'93), Pittsburgh, PA, USA, pp.191-202

[5] http://nlp.stanford/tagger/shtml

[6] http://tartarus.org/~martin/PorterStemmer/

[7] Katz S. M. (1995). Distribution of content words and phrases in text and language modeling. Natural Language Engineering 2(1): 15–59.

[8] Chen K. H. and Chen H. H. (1995), "A corpus-based approach to text partition," In Proceedings of the Workshop of Recent Advances in Natural Language Processing, , pp. 152-161