DOCUMENT SUMMARIZATION

PROJECT

Submitted in partial fulfillment

of the requirements for the degree of

BACHELOR OF TECHNOLOGY
(CSE & / IT)

by

Chandan Sharma– 071420
Pradeep Kumar- 071316
Shirish Kumar- 071311
Tashi Bodh- 071308

Under the Supervision of

Mr. Ravikant Verma

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY
WAKNAGHAT
SOLAN, HIMACHAL PRADESH
INDIA
2011

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY
WAKNAGHAT
SOLAN, HIMACHAL PRADESH

Date:

CERTIFICATE

This is to certify that the thesis entitled **DOCUMENT SUMMARIZATION** is submitted in the partial fulfillment of the award of degree of Bachelor of Technology (C.S.E.) by **JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT.**

Signature in full of Supervisor: ______________________________

Name in Capital block letters: Mr. RAVIKANT VERMA___________

Designation: LECTURER__________________________________

ACKNOWLEDGEMENT

The project entitled as **"DOCUMENT SUMMARIZATION"** is a software which is used for converting a html file into a text file and summarizing it by finding noun, pronoun and verb from any selected line of that particular page. This software can also change the font color, font size and font style. This project is our combined effort and realizes the potential of team and gives us a chance to work in co-ordination.

Again, I would like to express our sincere thanks and gratitude to the project guide **'Mr. Ravikant Verma'** under whose guidance we are going to complete this project. He provides us the kind of strategies required for the completion of a task.

ABSTRACT

The project '**DOCUMENT SUMMARIZATION**' which simply summarizes HTML page. First step is to take any HTML page and then convert it into a text file by removing all the tags of HTML page. It also has the feature to change in font size, font style and font color. Next step is to summarize the whole text file into a single paragraph. Document Summarization, is based on taking into account a cognitive psycho-logical model, and the roles and importance of sentences and their syntax in document understanding. The software involves entering some information from the converted html page and the software will provide us the details about nouns, verbs and pronoun in the line entered. After thorough implementation and objective evaluations, the algorithm has now shown good performance in HTML document summarization.

TABLE OF CONTENTS

LIST OF FIGURES

1.)INTRODUCTION

The task of Automatic Document Summarization is to take an information source, extract content from it, and present the most important content to the user in a condensed form and in a manner sensitive to the user's or application's needs . This area is highly interdisciplinary and related with natural language processing, artificial intelligence, information retrieval, information extraction, statistics and cognitive psychology. Over the last few years, much of the research has been concentrated in finding effective algorithms and building up efficient systems As Automatic Document Summarization is closely related with cognitive psychology as well, much attention and study have been paid in psychological models for text understanding and representations, To summarize a document or documents, a reader has to understand the document(s) and integrate information and make connections across sentences to form a coherent discourse representation . Syntax is there in order to allow the construction of formal structures by means of which complex meanings can be expressed. Phrases form not only syntactic units but also semantic units . Considering the importance of sentences, syntax and phrases, and being inspired from the Event-Indexing model, we designed and developed a new generic algorithm for automatic document summarization based on the analysis of human cognition and intelligence.

2. CONVERSION

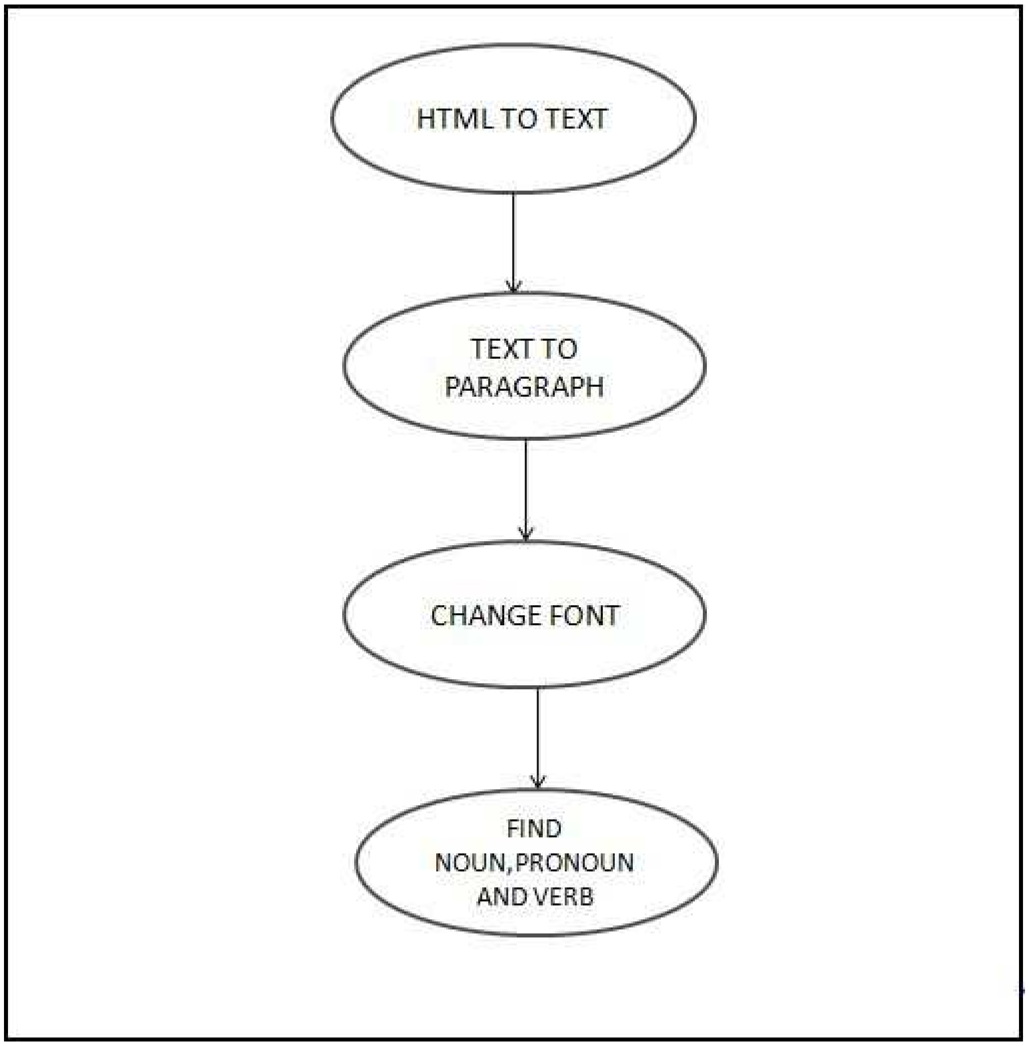
HTML file is processed through HTML parser for conversion into text file. The text file obtained after parsing is without any tags. The parser takes input document and remove all the links and hyperlinks from the HTML file.

3. TEXT FILE TO PARAGRAPH

This module uses some regular expressions to split the text section extracted from document(s) into sentences. As it is known that some abbreviations (e.g. 'Mr.', 'Dr.' and 'i.e.') may cause incorrect segmentation, a collection of such abbreviations has
been made to reduce segmentation errors. This module breaks the original structure of each document but records the original information of the position of each sentence in the document.

In this part the HTML file which is converted into paragraph is taken into account. In this we select the required line for the comparison and finding out the noun, verb and pronoun. For this process we use our database for comparison and finding out the required result. The database is in Microsoft sql server.

- This is the final process of our document summarization project.


HTML TO TEXT
TEXT TO PARAGRAPH
CHANGE FONT
FIND NOUN,PRONOUN AND VERB

4.PROBLEMS OF EXISTING SYSTEM

The existing system of Steganography poses more restrictions on the choosing of media files. User can select only wav files to encode. It supports water marking method to encode .It complexity arises when more message to be encoded. The message length is restricted to 500 characters. It doesn’t shows the variations occurred after encoding the message. The LSB algorithm in the existing system is not efficient because it hides the message in consecutive bytes received from media files.

The disadvantages of existing system are:-

1. Length of the string to be entered is limited to 300.
2. Lack in good user interface.
3. Only compatible with Microsoft SQL server .
4. User needs to understand better to know the operations.

These are the disadvantages in the existing system which can be overcome by the proposed system.

5.<u>SOFTWARE REQUIREMENTS SPECIFICATION</u>

Ultimately the requirement phase translates the ideas whatever is in the mind of client (the input) into a formal document (the output of the requirement phase.). In a more general way the SRS is a document that completely describes “What” the proposed system should do without describing “How” the software will do it.

FEASIBILITY STUDY

The feasibility study concerns with the consideration made to verify whether the system fit to be developed in all terms. Once an idea to develop software is put forward the question that arises first will pertain to the feasibility aspects.

There are different aspects in the feasibility study:
- Operational Feasibility.
- Technical Feasibility.
- Economical Feasibility.

OPERATIONAL FEASIBILITY:

There in no difficulty in implementing the system, if the user has the knowledge in internal working of the system. Therefore, it is assumed that he will not face any problem in running the system. The main problem faced during development of a new system is getting acceptance from the users. As users are responsible for initiating the development of a new system this is rooted out.

TECHNICAL FEASIBILITY:

Technical feasibility deals with the study of function, performance, and constraints like resources availability, technology, development risk that may affect the ability to achieve an acceptable system.

ECONOMICAL FEASIBILITY:

One of the factors, which affect the development of a new system, is the cost it would incur. The existing resources available in the company are sufficient for implementing the proposed and hence no extra cost has to be incurred to run the system developed. Thus, the system is financially feasible.

6. SYSTEM REQUIREMENTS

HARDWARE REQUIREMENTS (MININUM)

CPU	: 1.6 GHz
RAM	: 384 MB
DISPLAY	: 1024*768 Monitor
Hard Disk	: 2 GB

SOFTWARE REQUIREMENTS

OS	: Windows XP/Vista/7
Software	: Microsoft Visual Studio 2008
Database	: Microsoft SQL Server 2005

WATERFALL MODEL

The waterfall model derives its name due to the cascading effect from one phase to the other as is illustrated in Figure1.1. In this model each phase well defined starting and ending point, with identifiable deliveries to the next phase.

Note that this model is sometimes referred to as the linear sequential model or the software life cycle. The model consists of six distinct stages, namely:

1. REQUIREMENTS ANALYSIS

In the requirements analysis phase:
(a) The problem is specified along with the desired service objectives (goals).
(b) The constraints are identified.

2. SPECIFICATION PHASE

In the specification phase the system specification is produced from the detailed definitions of (a) and (b) above. This document should clearly define the product function.
Note that in some text, the requirements analysis and specifications phases are combined and represented as a single phase.

3. SYSTEM AND SOFTWARE DESIGN PHASE

In the system and software design phase, the system specifications are translated into a software representation. The software engineer at this stage is concerned with:

a. Data structure
b. Software architecture
c. Database connectivity

4. IMPLEMENTATION AND TESTING PHASE

In the implementation and testing phase stage the designs are translated into the software domain:

- ➢ Detailed documentation from the design phase can significantly reduce the coding effort.

- ➢ Testing at this stage focuses on making sure that any errors are identified and that the software meets its required specification.

5. INTEGRATION AND SYSTEM TESTING PHASE

In the integration and system testing phase all the program units are integrated and tested to ensure that the complete system meets the software requirements. After this stage the software is delivered to the customer [Deliverable – The software product is delivered to the client for acceptance testing.]

6. MAINTENANCE PHASE

The maintenance phase the usually the longest stage of the software. In this phase the software is updated to:

- ➢ Meet the changing customer needs
- ➢ Adapted to accommodate changes in the external environment
- ➢ Correct errors and oversights previously undetected in the testing phases
- ➢ Enhancing the efficiency of the software

Observe that feed back loops allow for corrections to be incorporated into the model. For example a problem/update in the design phase requires a 'revisit' to the specifications phase. When changes are made at any phase, the relevant documentation should be updated to reflect that change.

7. DESIGNING PHASE

PROGRAM FLOW DIAGRAM:

The diagram model

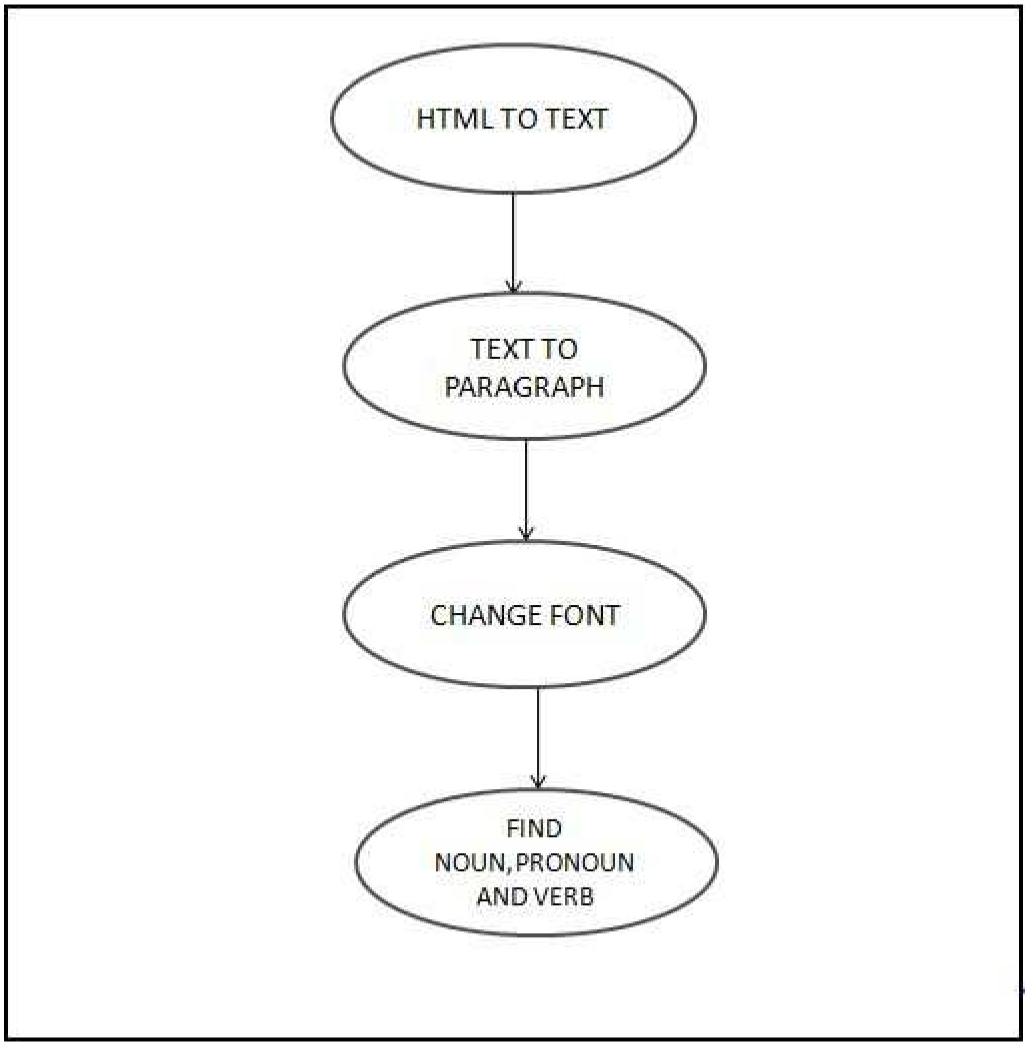


Fig. 1

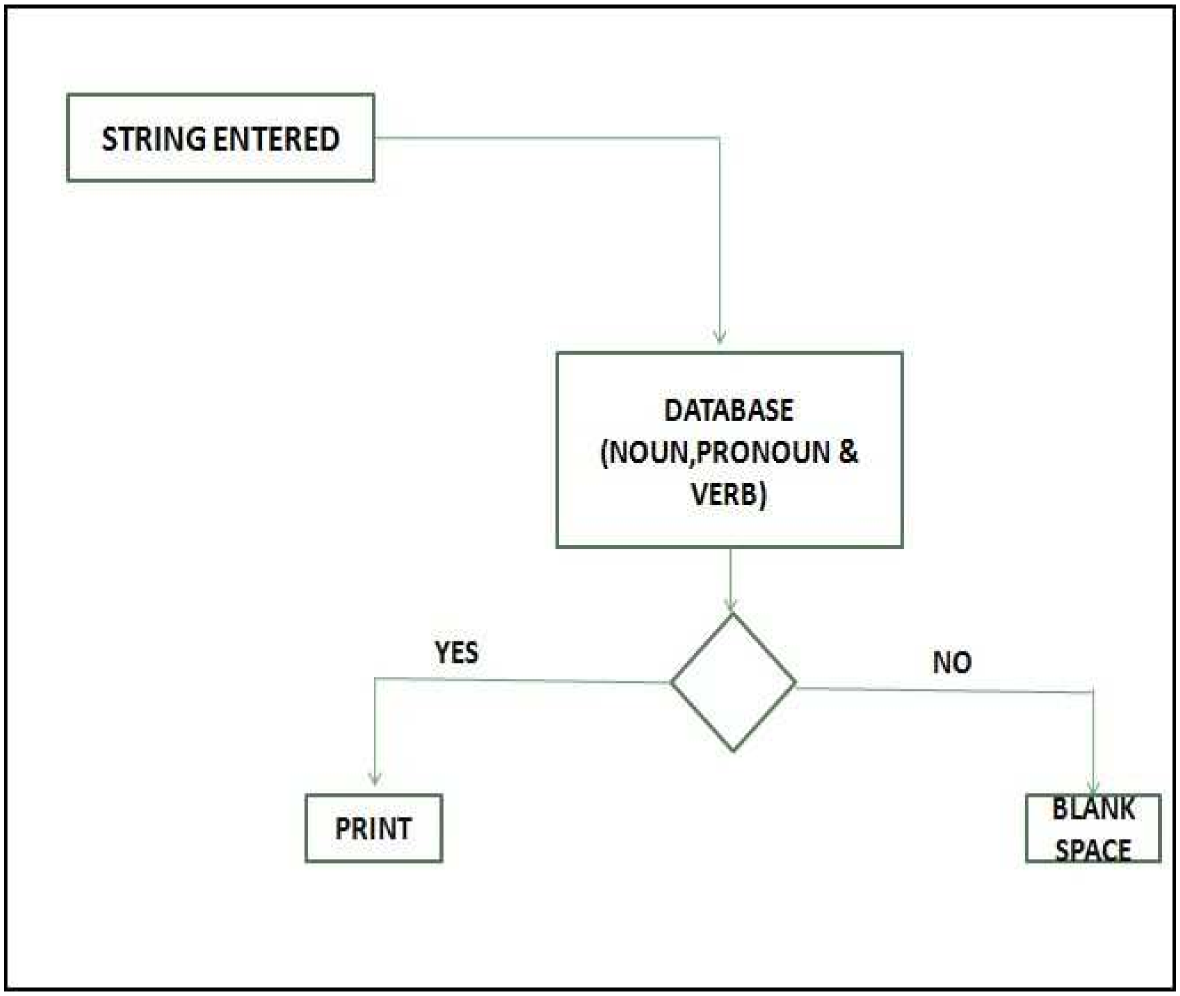

STRING ENTERED
DATABASE (NOUN,PRONOUN & VERB)
YES
NO
PRINT
BLANK SPACE

Fig. 2

0 LEVEL DFD

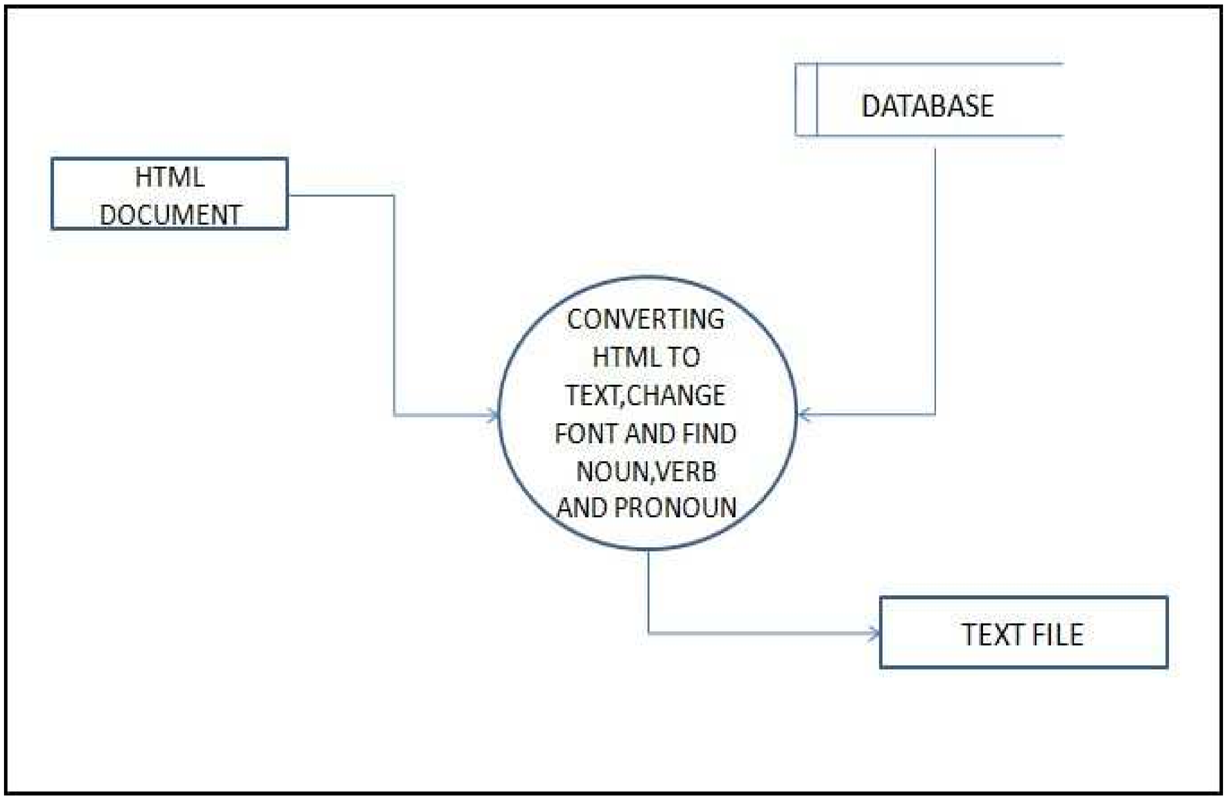

DATABASE
HTML DOCUMENT
CONVERTING HTML TO TEXT,CHANGE FONT AND FIND NOUN,VERB AND PRONOUN
TEXT FILE

Fig. 3

1.) Inputs: HTML file
2.) This Project summarizes html document and find out nouns, pronouns and verbs.

1ST LEVEL DFD (DECODE)

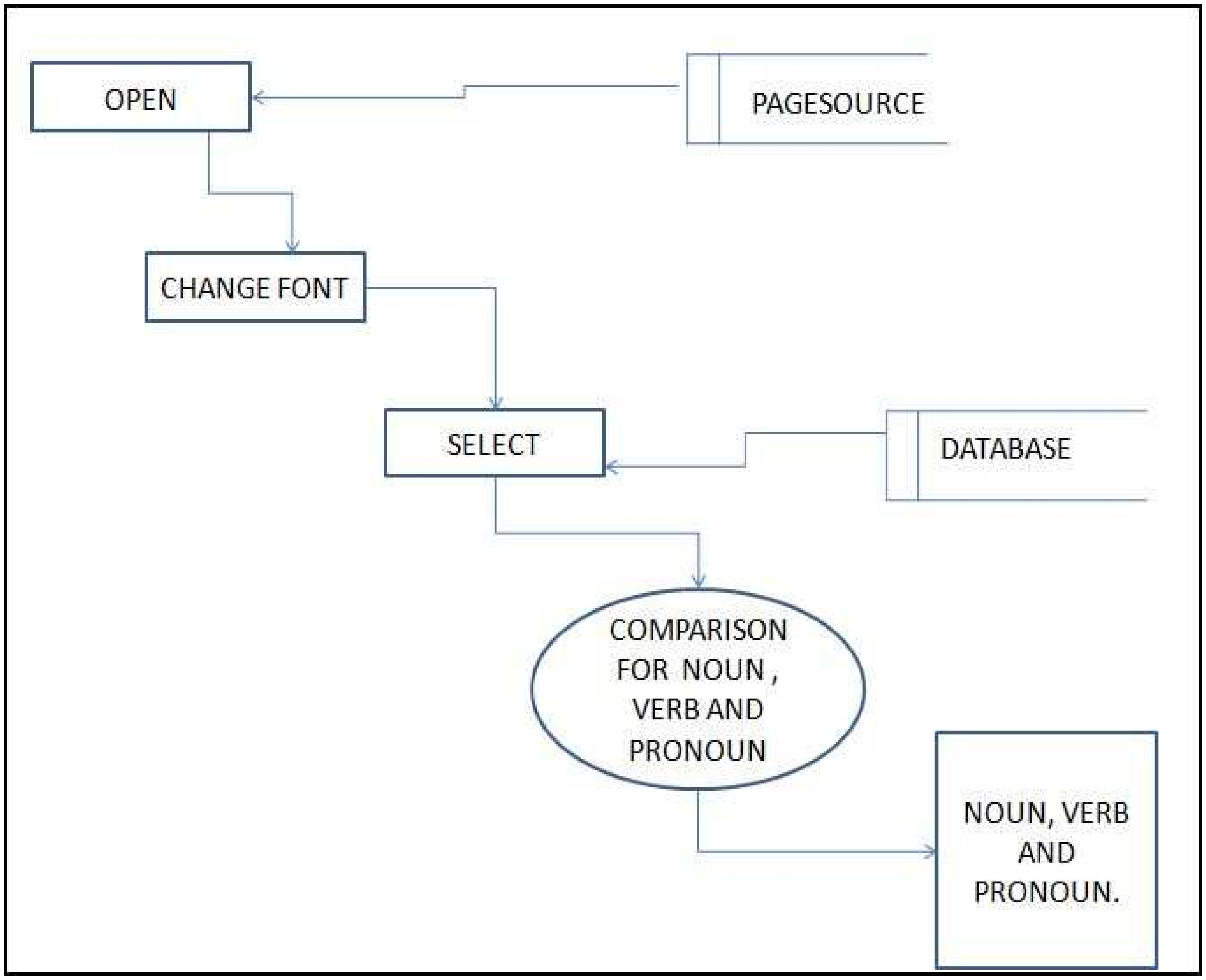


Fig. 4

1) Open HTML file.
2) Convert to text and change font.
3) Comparison from the database
4) Message is displayed and file can be saved

CONCLUSION

After carefully studying cognitive psychology models associated with document summarization based on the parsing techniques. The algorithm involves syntactic parsing and using Word Net for lexical referencing to get accurate segmentation and analysis on phrase and clause levels, by mimicking the procedure of human cognition and intelligence in document summarization. This software algorithm showed its effectiveness in summarization.

FUTURE WORK

Future enhancements recommended are :

1. Compatibility with other database system.
2. Applying clustering techniques.
3. The limit of message length to be increased.
4. User Interface to be improved.

APPENDIX – A

Features of Microsoft Visual Studio 2008 (.NET Framework 3.5)

Microsoft Visual Studio 2008 delivers on the Microsoft vision of smart client applications by enabling developers to rapidly create connected applications that deliver the highest quality, rich user experiences. With Visual Studio 2008, organizations will find it easier than ever before to capture and analyze information to help them make effective business decisions. Visual Studio 2008 enables organizations of every size to rapidly create more secure, manageable, and reliable applications that take advantage of Windows Vista and the 2007 Office system.

Visual Studio 2008 delivers key advances for developers in three primary pillars:

- Rapid application development
- Effective team collaboration
- Break through user experiences

Visual Studio 2008 provides advanced development tools, debugging features, database functionality, and innovative features for quickly creating tomorrow's cutting-edge applications across a variety of platforms.

Visual Studio 2008 includes enhancements such as visual designers for faster development with the .NET Framework 3.5, substantial improvements to Web development tools and language enhancements that speed development with all types of data. Visual Studio 2008 provides developers with all the tools and framework support required to create compelling, expressive, AJAX-enabled Web applications.

Developers will be able to take advantage of these rich client-side and server-side, frameworks to easily build client-centric Web applications that integrate with any back-end data provider, run within any modern browser, and have complete access to ASP.NET application services and the Microsoft platform.

RAPID APPLICATION DEVELOPMENT

To help developers rapidly create modern software, Visual Studio 2008 delivers improved language and data features, such as Language Integrated Query (LINQ), that make it easier for individual programmers to build solutions that analyze and act on information.

Visual Studio 2008 also provides developers with the ability to target multiple versions of the .NET Framework from within the same development environment. Developers will be able to build applications that target the .NET Framework 2.0, 3.0 or 3.5, meaning that they can support a wide variety of projects in the same environment.

BREAK THROUGH USER EXPERIENCE

Visual Studio 2008 offers developers new tools that speed creation of connected applications on the latest platforms including the Web, Windows Vista, Office 2007, SQL Server 2008, and Windows Server 2008. For the Web, ASP.NET AJAX and other new technologies will enable developers to quickly create a new generation of more efficient, interactive, and personalized Web experiences.

EFFECTIVE TEAM COLLABORATION

Visual Studio 2008 delivers expanded and improved offerings that help improve collaboration in development teams, including tools that help integrate database professionals and graphic designers into the development process.

APPENDIX – B

CODING

DocumentSummarizer.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Text.RegularExpressions;
using System.IO;

namespace HTML_TO_TEXT
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object
sender, EventArgs e)
        {

        }
        private string StripHTML(string
source)
```

```
{
    try
    {
        string result;

        // Remove HTML Development
formatting
        // Replace line breaks with space
        // because browsers inserts space
        result = source.Replace("\r", " ");
        // Replace line breaks with space
        // because browsers inserts space
        result = result.Replace("\n", " ");
        // Remove step-formatting
        result = result.Replace("\t",
string.Empty);
        // Remove repeating spaces because
browsers ignore them
        result =
System.Text.RegularExpressions.Regex.Replace
(result,

@"( )+", " ");

        // Remove the header (prepare first
by clearing attributes)
        result =
System.Text.RegularExpressions.Regex.Replace
(result,
                 @"<(
)*head([^>])*>","<head>",

System.Text.RegularExpressions.RegexOptions.
IgnoreCase);
```

```
        result =
System.Text.RegularExpressions.Regex.Replace
(result,
                 @"(<( )*(/)( )*head(
)*>)","</head>",

System.Text.RegularExpressions.RegexOptions.
IgnoreCase);
        result =
System.Text.RegularExpressions.Regex.Replace
(result,

"(<head>).*(</head>)",string.Empty,

System.Text.RegularExpressions.RegexOptions.
IgnoreCase);

        // remove all scripts (prepare first
by clearing attributes)
        result =
System.Text.RegularExpressions.Regex.Replace
(result,
                 @"<(
)*script([^>])*>","<script>",

System.Text.RegularExpressions.RegexOptions.
IgnoreCase);
        result =
System.Text.RegularExpressions.Regex.Replace
(result,
                 @"(<( )*(/)( )*script(
)*>)","</script>",

System.Text.RegularExpressions.RegexOptions.
IgnoreCase);
```

```
        //result = 
System.Text.RegularExpressions.Regex.Replace
(result,
        //
@"(<script>)([^(<script>\.</script>)])*(</sc
ript>)",
        //          string.Empty,
        //
System.Text.RegularExpressions.RegexOptions.
IgnoreCase);
        result = 
System.Text.RegularExpressions.Regex.Replace
(result,

@"(<script>).*(</script>)",string.Empty,

System.Text.RegularExpressions.RegexOptions.
IgnoreCase);

        // remove all styles (prepare first
by clearing attributes)
        result = 
System.Text.RegularExpressions.Regex.Replace
(result,
                 @"<(
)*style([^>])*>","<style>",

System.Text.RegularExpressions.RegexOptions.
IgnoreCase);
        result = 
System.Text.RegularExpressions.Regex.Replace
(result,
                 @"(<( )*(/)( )*style(
)*>)","</style>",
System.Text.RegularExpressions.RegexOptions.
IgnoreCase);
        result =
System.Text.RegularExpressions.Regex.Replace
(result,

"(<style>).*(</style>)",string.Empty,

System.Text.RegularExpressions.RegexOptions.
IgnoreCase);

        // insert tabs in spaces of <td>
tags
        result =
System.Text.RegularExpressions.Regex.Replace
(result,
                 @"<( )*td([^>])*>","\t",

System.Text.RegularExpressions.RegexOptions.
IgnoreCase);

        // insert line breaks in places of
<BR> and <LI> tags
        result =
System.Text.RegularExpressions.Regex.Replace
(result,
                 @"<( )*br( )*>","\r",

System.Text.RegularExpressions.RegexOptions.
IgnoreCase);
        result =
System.Text.RegularExpressions.Regex.Replace
(result,
                 @"<( )*li( )*>","\r",
System.Text.RegularExpressions.RegexOptions.
IgnoreCase);

        // insert line paragraphs (double
line breaks) in place
        // if <P>, <DIV> and <TR> tags
        result =
System.Text.RegularExpressions.Regex.Replace
(result,
                 @"<( )*div([^>])*>","\r\r",

System.Text.RegularExpressions.RegexOptions.
IgnoreCase);
        result =
System.Text.RegularExpressions.Regex.Replace
(result,
                 @"<( )*tr([^>])*>","\r\r",

System.Text.RegularExpressions.RegexOptions.
IgnoreCase);
        result =
System.Text.RegularExpressions.Regex.Replace
(result,
                 @"<( )*p([^>])*>","\r\r",

System.Text.RegularExpressions.RegexOptions.
IgnoreCase);

        // Remove remaining tags like <a>,
links, images,
        // comments etc - anything that's
enclosed inside < >
```

```
        result =
System.Text.RegularExpressions.Regex.Replace
(result,
                 @"<[^>]*>",string.Empty,

System.Text.RegularExpressions.RegexOptions.
IgnoreCase);

        // replace special characters:
        result =
System.Text.RegularExpressions.Regex.Replace
(result,
                 @" "," ",

System.Text.RegularExpressions.RegexOptions.
IgnoreCase);

        result =
System.Text.RegularExpressions.Regex.Replace
(result,
                 @"&bull;"," * ",

System.Text.RegularExpressions.RegexOptions.
IgnoreCase);
        result =
System.Text.RegularExpressions.Regex.Replace
(result,
                 @"&lsaquo;","<",

System.Text.RegularExpressions.RegexOptions.
IgnoreCase);
        result =
System.Text.RegularExpressions.Regex.Replace
(result,
                 @"&rsaquo;",">",
System.Text.RegularExpressions.RegexOptions.
IgnoreCase);
        result =
System.Text.RegularExpressions.Regex.Replace
(result,
                 @"&trade;","(tm)",

System.Text.RegularExpressions.RegexOptions.
IgnoreCase);
        result =
System.Text.RegularExpressions.Regex.Replace
(result,
                 @"&frasl;","/",

System.Text.RegularExpressions.RegexOptions.
IgnoreCase);
        result =
System.Text.RegularExpressions.Regex.Replace
(result,
                 @"&lt;","<",

System.Text.RegularExpressions.RegexOptions.
IgnoreCase);
        result =
System.Text.RegularExpressions.Regex.Replace
(result,
                 @"&gt;",">",

System.Text.RegularExpressions.RegexOptions.
IgnoreCase);
        result =
System.Text.RegularExpressions.Regex.Replace
(result,
                 @"&copy;","(c)",
System.Text.RegularExpressions.RegexOptions.
IgnoreCase);
        result =
System.Text.RegularExpressions.Regex.Replace
(result,
                 @"&reg;","(r)",

System.Text.RegularExpressions.RegexOptions.
IgnoreCase);
        // Remove all others. More can be
added, see
        //
http://hotwired.lycos.com/webmonkey/referenc
e/special_characters/
        result =
System.Text.RegularExpressions.Regex.Replace
(result,
                 @"&(.{2,6});",
string.Empty,

System.Text.RegularExpressions.RegexOptions.
IgnoreCase);

        // for testing

//System.Text.RegularExpressions.Regex.Repla
ce(result,
        //
this.txtRegex.Text,string.Empty,
        //
System.Text.RegularExpressions.RegexOptions.
IgnoreCase);

        // make line breaking consistent
        result = result.Replace("\n", "\r");

        // Remove extra line breaks and
tabs:
        // replace over 2 breaks with 2 and
over 4 tabs with 4.
        // Prepare first to remove any
whitespaces in between
        // the escaped characters and remove
redundant tabs in between line breaks
        result =
System.Text.RegularExpressions.Regex.Replace
(result,
                 "(\r)( )+(\r)","\r\r",

System.Text.RegularExpressions.RegexOptions.
IgnoreCase);
        result =
System.Text.RegularExpressions.Regex.Replace
(result,
                 "(\t)( )+(\t)","\t\t",

System.Text.RegularExpressions.RegexOptions.
IgnoreCase);
        result =
System.Text.RegularExpressions.Regex.Replace
(result,
                 "(\t)( )+(\r)","\t\r",

System.Text.RegularExpressions.RegexOptions.
IgnoreCase);
        result =
System.Text.RegularExpressions.Regex.Replace
(result,
                 "(\r)( )+(\t)","\r\t",
System.Text.RegularExpressions.RegexOptions.
IgnoreCase);
        // Remove redundant tabs
        result =
System.Text.RegularExpressions.Regex.Replace
(result,
                 "(\r)(\t)+(\r)","\r\r",

System.Text.RegularExpressions.RegexOptions.
IgnoreCase);
        // Remove multiple tabs following a
line break with just one tab
        result =
System.Text.RegularExpressions.Regex.Replace
(result,
                 "(\r)(\t)+","\r\t",

System.Text.RegularExpressions.RegexOptions.
IgnoreCase);
        // Initial replacement target string
for line breaks
        string breaks = "\r\r\r";
        // Initial replacement target string
for tabs
        string tabs = "\t\t\t\t\t";
        for (int index=0;
index<result.Length; index++)
        {
            result = result.Replace(breaks,
"\r\r");
            result = result.Replace(tabs,
"\t\t\t\t");
            breaks = breaks + "\r";
            tabs = tabs + "\t";
        }

        // That's it.
        return result;
    }
    catch
    {
        MessageBox.Show("Error");
        return source;
    }
}

        private void
convertToToolStripMenuItem_Click(object
sender, EventArgs e)
        {
            String d=StripHTML(source.Text
);
           // source .Text =d.ToString ();
            this.source.Lines =

StripHTML(this.source.Text).Split("\r".ToCha
rArray());

        }

        private void
uploadToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            OpenFileDialog ofd = new
OpenFileDialog();
            DialogResult dr =
ofd.ShowDialog();
            if (dr == DialogResult.OK)
            {
                StreamReader sr = new
StreamReader(ofd.FileName);
                source.Text =
sr.ReadToEnd();
                sr.Close();
            }

        }

        private void
saveToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            SaveFileDialog sfd = new
SaveFileDialog();
            DialogResult dr = new
DialogResult();
            dr = sfd.ShowDialog();
            if (dr == DialogResult.OK)
            {
                String filename =
sfd.FileName + ".doc";
                StreamWriter sw = new
StreamWriter(filename);
                sw.Write(source.Text);
                sw.Close();
            }
        }

        private void
fontToolStripMenuItem_Click(object sender,
EventArgs e)
        {
        }

        private void
fontSizeAndStyleToolStripMenuItem_Click(obje
ct sender, EventArgs e)
        {
            FontDialog fntdlg = new
FontDialog();

            DialogResult dr =
fntdlg.ShowDialog();
            if(dr==DialogResult .OK )
            {
                source.Font = fntdlg.Font;
        }
        }

        private void
fontColorToolStripMenuItem_Click(object
sender, EventArgs e)
        {
            ColorDialog cldlg = new
ColorDialog();

            DialogResult dr =
cldlg.ShowDialog();
            if (dr == DialogResult.OK)
            {
                source.ForeColor =
cldlg.Color;
            }

        }
        private void
openPrinterToolStripMenuItem_Click(object
sender, EventArgs e)
        {
            printDialog1.AllowPrintToFile =
true;

              }

        private void
checkNVPToolStripMenuItem_Click(object
sender, EventArgs e)
        {
            Form2 frm = new Form2();
            frm.Show();
        }

        private void
saveAsToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            this.Close();
        }
    }
}
```

Databaseconnectivity.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace HTML_TO_TEXT
{
    public partial class Form2 : Form
    {
        String message = "";
        String MESSAGE2 = "";
        String MESSAGE3 = "";
        SqlConnection con = new
SqlConnection();
        Boolean signal = false;

        public Form2()
        {
            InitializeComponent();
        }

        private void button1_Click(object
sender, EventArgs e)
        {
            String noun = "";
            String pronoun = "";
            String verb = "";
            String word = "";
            String s = textBox1.Text;
            s += " ";
            int i = 0;
            int length = 0;
            int j = 0;
            Boolean signal2 = true;
            length = s.Length;
            while (i < length)
            {
                signal2 = true;
                word += s[i].ToString();
                if (s[i] == 32)
                {//here we will have to
fetch from the database
                    word = word.Replace(" ",
"");

                    SqlCommand cmdfetch =
new SqlCommand("select * from tbcheck where
noun=@noun", con);

cmdfetch.Parameters.Add("@noun",
SqlDbType.VarChar, 50).Value =
word.ToString();
                    SqlDataReader dr =
cmdfetch.ExecuteReader();
                    if (dr.HasRows)
                    {
                        noun += "  " +
word.ToString();
                    }
                    dr.Close();
                    cmdfetch.Dispose();
//noun fetched
                    SqlCommand cmdfetch1 =
new SqlCommand("select * from tbcheck where
pronoun=@noun1", con);

cmdfetch1.Parameters.Add("@noun1",
SqlDbType.VarChar, 50).Value =
word.ToString();
```

```
                    SqlDataReader dr1 =
cmdfetch1.ExecuteReader();
                    if (dr1.HasRows)
                    {
                        pronoun += "  " +
word.ToString();
                    }
                    dr1.Close();
                    cmdfetch1.Dispose();//
                    //pronoun fetched
                    SqlCommand cmdfetch2 =
new SqlCommand("select * from tbcheck where
verb=@noun2", con);

cmdfetch2.Parameters.Add("@noun2",
SqlDbType.VarChar, 50).Value =
word.ToString();
                    SqlDataReader dr2 =
cmdfetch2.ExecuteReader();
                    if (dr2.HasRows)
                    {
                        verb += "  " +
word.ToString();
                    }
                    dr2.Close();
                    cmdfetch2.Dispose();
                    //HERE I HAVE ADDED ONE
MORE STRING....17/05/2011
                    word = word + " ";




                   //
MessageBox.Show(word.ToString());
```

```
                        s = s.Replace(word, "");
                        length = s.Length;
                        word = "";

                        if (i == length)
                        {
                            SqlCommand
cmdfetch88 = new SqlCommand("select * from
tbcheck where noun=@noun", con);

cmdfetch88.Parameters.Add("@noun",
SqlDbType.VarChar, 50).Value =
word.ToString();
                            SqlDataReader dr88 =
cmdfetch88.ExecuteReader();
                            if (dr88.HasRows)
                            {
                                noun += "  " +
word.ToString();
                            }
                            dr88.Close();

cmdfetch88.Dispose();
                            //noun fetched
                            SqlCommand
cmdfetch111 = new SqlCommand("select * from
tbcheck where pronoun=@noun1", con);

cmdfetch111.Parameters.Add("@noun1",
SqlDbType.VarChar, 50).Value =
word.ToString();
                            SqlDataReader dr111
= cmdfetch111.ExecuteReader();
                            if (dr111.HasRows)
                            {
```

```
                                        pronoun += "  "
+ word.ToString();
                                    }
                                    dr111.Close();

cmdfetch111.Dispose();//
                                    //pronoun fetched
                                    SqlCommand
cmdfetch22 = new SqlCommand("select * from
tbcheck where verb=@noun2", con);

cmdfetch22.Parameters.Add("@noun2",
SqlDbType.VarChar, 50).Value =
word.ToString();
                                    SqlDataReader dr22 =
cmdfetch22.ExecuteReader();
                                    if (dr22.HasRows)
                                    {
                                        verb += "  " +
word.ToString();
                                    }
                                    dr22.Close();

cmdfetch22.Dispose();
                                   // s = s.Trim();

                                   // word = "";

                               }
                               i = 0;
                               signal2 = false;
                           }
                           i++;
                           if (signal2 == false)
                           {
                            i = 0;
                        }

                    }

//length = length + 2;
                    //while (i <(length-1))
                    //{

                    //    if (s[i] != 32 && s[i] !=
44)
                    //    {
                    //        signal = false;
                    //       // s = s.Remove(1);



                    //            word =
s.Substring(0, i);
                    //            if (i == (length-
1))
                    //            {
                    //
MessageBox.Show(word.ToString());
                    //            }

                    //       // word = s.Remove(0,
i);
                    //
//MessageBox.Show(word.ToString());
                    //    }


                    //    if (s[i] == 32 || s[i] ==
44)
```

```
            //     {
            //         signal = true;
            //         i++;
            //         word = s.Substring(0,
i);
            //         s = s.Replace(word,
"");
            //
//;s.Replace(word.ToString(), "");
            //
MessageBox.Show(word.ToString());
            //        // word = "";
            //         i--;
            //     }
            //    i++;



            //}
            //i=0;
            message = message + "NOUN :" +
noun;
            MESSAGE2 = MESSAGE2 + "PRONOUN :
" + pronoun;
            MESSAGE3 = MESSAGE3 + "VERB :  "
+ verb;

            MessageBox.Show(message);

          MessageBox.Show(MESSAGE2);

              MessageBox.Show(MESSAGE3);
              message = "";
              MESSAGE2 = "";
              MESSAGE3 = "";
```

```
        }

        private void Form2_Load(object
sender, EventArgs e)
        {
            try
            {
                con.ConnectionString =
"server=napster-
pc;database=httext;uid=sa;pwd=123";
                if (con.State ==
ConnectionState.Closed)
                {
                    con.Open();
                }
            }
            catch
            {
                MessageBox.Show("PROBLEM IN
CONNECTIVITY,CHECK YOUR SQL SERVER
SETTINGS");
            }
        }
    }
}
```

REFERENCES

1. **AN INTELLIGENT ALGORITHM FOR AUTOMATIC DOCUMENT SUMMARIZATION**
   Yi Guo and George Stylios
   RiFlex, Heriot-Watt University
   Scotland, TD1 3HF, U.K.
   {y.guo, g.stylios}@hw.ac.uk

1. **www.wikipedia.com**

2. **www.google.com**

3. **www.sourcefourge.com**

4. **www.w3schools.com**