A Project report on

VECTOR SPACE MODEL FILE SEARCH

Submitted in partial fulfillment of IV Semester, BCA, Mini project, related to Visual Programming [BCA403T] finalization

Bachelor of Computer Applications

Under

Bangalore North University, Karnataka

By

Pawan Kumar A [R1818835]

&

Natchapon Sirisorn [R1818833]

Under the guidance of

Dr. Krishnan R
Associate Professor
Department of BCA
Krupanidhi Degree College

Krupanidhi Degree College
12/1, Chikka Bellandur,Carmelaram Post Varthur Hobli,
Off Sarjapur Road, Bangalore, Karnataka 560035

August 2019

KRUPANIDHI DEGREE COLLEGE

12/1, Chikka Bellandur,Carmelaram Post Varthur Hobli,
Off Sarjapur Road, Bangalore, Karnataka 560035
AFFILIATED TO BANGALORE NORTH UNIVERSITY, BANGALORE
APPROVED BY GOVT. OF KARNATAKA, ACCREDITED BY NAAC, NEW DELHI

DEPARTMENT OF BACHELOR OF COMPUTER APPLICATIONS

CERTIFICATE

This is to certify that the project entitled 'VECTOR SPACE MODEL FILE SEARCH' submitted in partial fulfillment of IV semester BCA, Mini project, related to the subject Visual Programming [BCA403T] of Bachelor of Computer Applications is a result of the work carried out by Pawan Kumar A [R1818835] & Natchapon Sirisorn [R1818833] during the academic year 2019-20 [Even Semester 2020]

Dr. Krishnan R
Course In- charge,
Department of BCA
Krupanidhi Degree College
Bangalore-35

Prof. Shubhi Srivasthava
Head of Department
Department of BCA
Krupanidhi Degree College
Bangalore-35

Principal
Krupanidhi Degree College

UNDERTAKING BY THE STUDENTS

We the students of IV Semester, **Pawan Kumar A [R1818835] & Natchapon Sirisorn [R1818833]** hereby declare that we have genuinely worked on the project titled **'VECTOR SPACE MODEL FILE SEARCH'**

Pawan Kumar A [R1818835]

Natchapon Sirisorn [R1818833]

ACKNOWLEDGMENT

The satisfaction and euphoric that accompany the success of any work would be incomplete unless we mention the name of the people, who made it possible, whose constant guidance and encouragement served a beacon light and served our effort with success.

We are thankful to the head of the institution, **Dr. Badarunnisa S, Professor & Principal,** for providing an opportunity to do the project.

We are thankful to Prof. Shubhi Srivastava**,** HOD, BCA Department for being source of encouragement.

We express our sincere thanks and credit to our internal guide Dr. **Krishnan R**, Associate Professor, BCA Dept, KDC for his constant encouragement, support and guidance during the project work.

We are also thankful to all faculty of the department for their help and support during the project.

Pawan Kumar A [R1818835]

Natchapon Sirisorn [R1818833]

TABLE OF CONTENTS

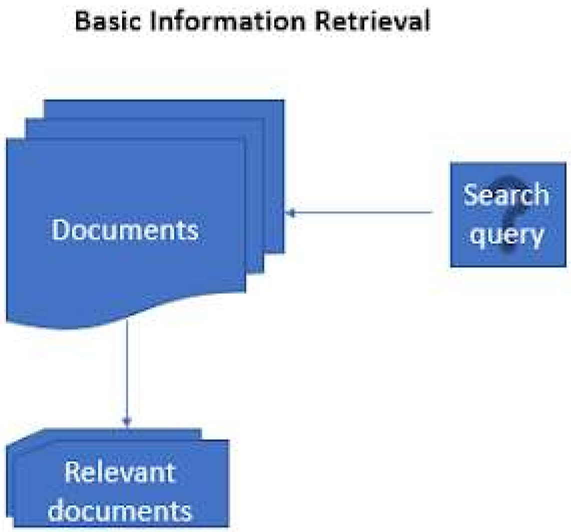
Particulars **Page No**

Chapter 1: Synopsis

1.1 Objective

To implement a basic search engine or document retrieval system using Vector space model. This use case is widely used in information retrieval systems. Given a set of documents and search term(s)/query we need to retrieve relevant documents that are similar to the search query.

1.2 Project Concept



Document retrieval system

Before we get into building the search engine, we will learn briefly about different concepts we use in this post:

Vector Space Model:

A vector space model is an algebraic model, involving two steps, in first step we represent the text documents into vector of words and in second step we transform to numerical format so that we can apply any text mining techniques such as information retrieval, information extraction,information filtering etc.

Let us understand with an example. consider below statements and a query term. The statements are referred as documents hereafter.

Document 1: Cat runs behind rat
Document 2: Dog runs behind cat
Query: rat

Document vectors representation:

In this step includes breaking each document into words, applying preprocessing steps such as removing stopwords, punctuations, special characters etc. After preprocessing the documents we represent them as vectors of words.
Below is a sample representation of the document vectors.

2 *Document 1: (cat, runs, behind, rat)*
Document 2: (Dog, runs, behind, cat)
Query: (rat)

the relevant document to Query = greater of (similarity score between (Document1, Query), similarity score between (Document2, Query)

Next step is to represent the above created vectors of terms to numerical format known as term document matrix.

Term Document Matrix:

A term document matrix is a way of representing documents vectors in a matrix format in which each row represents term vectors across all the documents and columns represent document vectors across all the terms. The cell values frequency counts of each term in corresponding document. If a term is present in a document, then the corresponding cell value contains 1 else if the term is not present in the document then the cell value contains 0.

After creating the term document matrix, we will calculate term weights for all the terms in the matrix across all the documents. It is also important to calculate the term weightings because we need to find out terms which uniquely define a document.

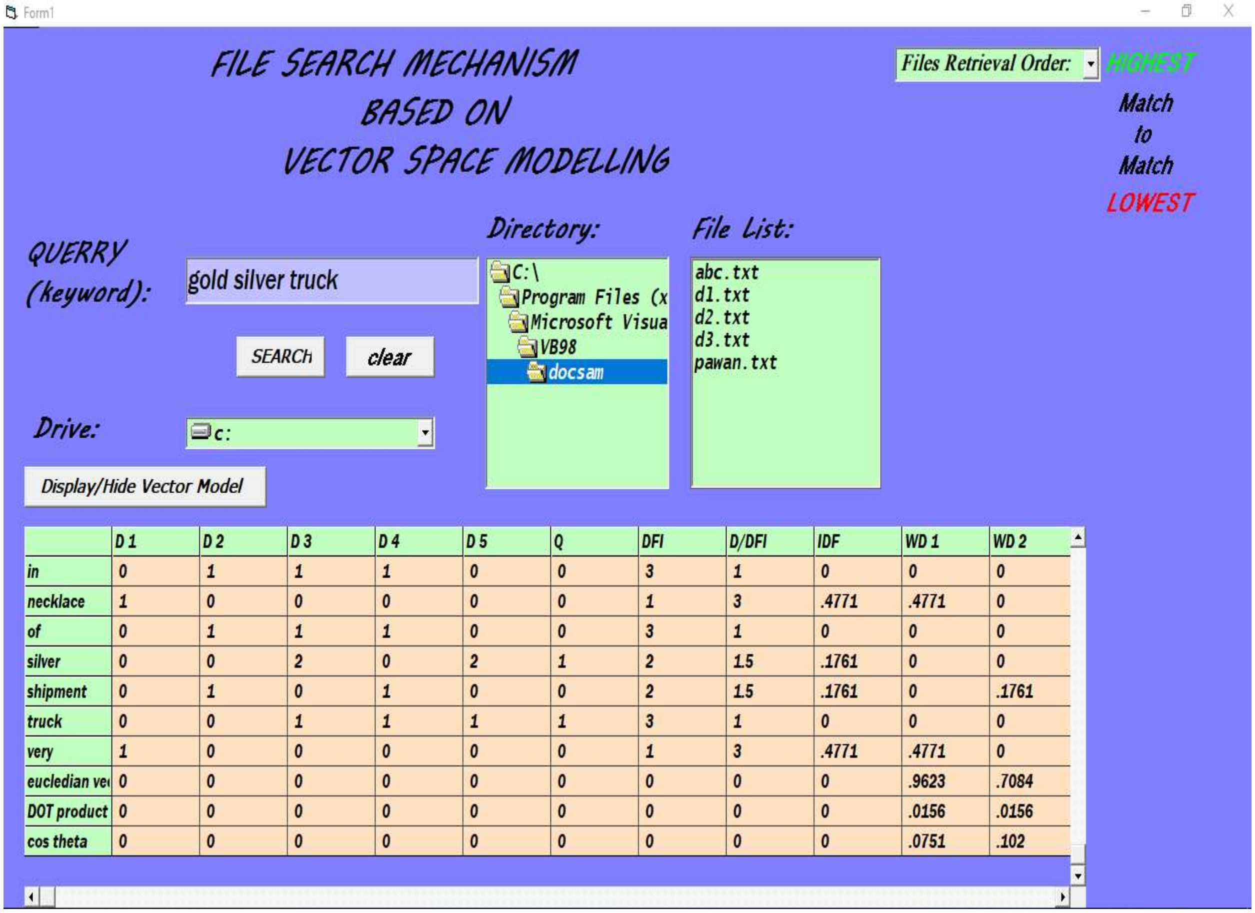
1.3 Tools Used

Visual Studio VB Compiler for Implementation, MS Word for documentation

1.4 Outcome of the Project

Mathematically, closeness between two vectors is calculated by calculating the cosine angle between two vectors. In similar lines, we can calculate cosine angle between each document vector and the query vector to find its closeness. To find relevant document to the query term , we may calculate the similarity score between each document vector and the query term vector by applying cosine similarity . Finally, whichever documents having high similarity scores will be considered as relevant documents to the query term.

When we plot the term document matrix, each document vector represents a point in the vector space. In the below example query, Document 1 and Document 2 represent 3 points in the vector space. We can now compare the query with each of the document by calculating the cosine angle between them.



FILE SEARCH MECHANISM
BASED ON
VECTOR SPACE MODELLING

Files Retrieval Order: HIGHEST Match to Match LOWEST

QUERRY (keyword): gold silver truck

Drive: c:

Directory:
C:\
Program Files (x
Microsoft Visua
VB98
docsam

File List:
abc.txt
d1.txt
d2.txt
d3.txt
pawan.txt

	D 1	D 2	D 3	D 4	D 5	Q	DFI	D/DFI	IDF	WD 1	WD 2
in	0	1	1	1	0	0	3	1	0	0	0
necklace	1	0	0	0	0	0	1	3	.4771	.4771	0
of	0	1	1	1	0	0	3	1	0	0	0
silver	0	0	2	0	2	1	2	1.5	.1761	0	0
shipment	0	1	0	1	0	0	2	1.5	.1761	0	.1761
truck	0	0	1	1	1	1	3	1	0	0	0
very	1	0	0	0	0	0	1	3	.4771	.4771	0
eucledian ve	0	0	0	0	0	0	0	0	0	.9623	.7084
DOT product	0	0	0	0	0	0	0	0	0	.0156	.0156
cos theta	0	0	0	0	0	0	0	0	0	.0751	.102

2. Project Concept

A term document matrix is a way of representing documents vectors in a matrix format in which each row represents term vectors across all the documents and columns represent document vectors across all the terms. The cell values frequency counts of each term in corresponding document. If a term is present in a document, then the corresponding cell value contains 1 else if the term is not present in the document then the cell value contains 0.

After creating the term document matrix, we will calculate term weights for all the terms in the matrix across all the documents. It is also important to calculate the term weightings because we need to find out terms which uniquely define a document.

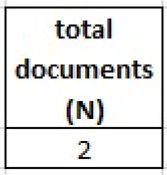
We should note that a word which occurs in most of the documents might not contribute to represent the document relevance whereas less frequently occurred terms might define document relevance. This can be achieved using a method known as term frequency - inverse document frequency (tf-idf), which gives higher weights to the terms which occurs more in a document but rarely occurs in all other documents, lower weights to the terms which commonly occurs within and across all the documents.

Tf-idf = tf X idf

tf = term frequency is the number of times a term occurs in a document

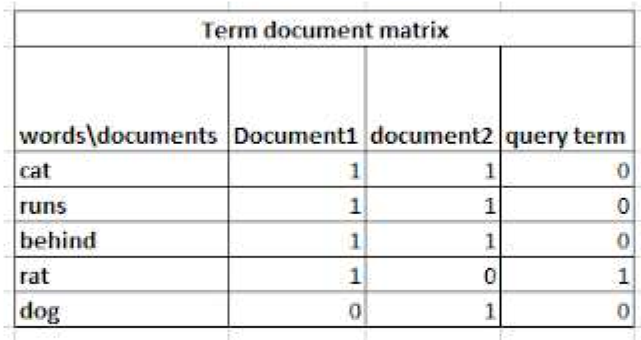
idf = inverse of the document frequency, given as below

idf = log(N/df), where df is the document frequency-number of documents containing a term



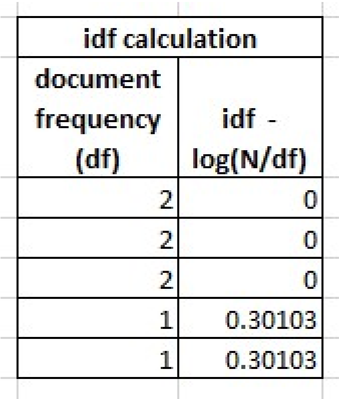
total documents (N)
2

total number of documents



Term document matrix			
words\documents	Document1	document2	query term
cat	1	1	0
runs	1	1	0
behind	1	1	0
rat	1	0	1
dog	0	1	0

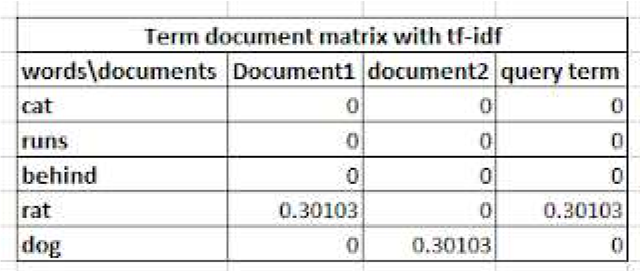
term document matrix



idf calculation	
document frequency (df)	idf - log(N/df)
2	0
2	0
2	0
1	0.30103
1	0.30103

inverse document frequency

Note: idf is calculated using logarithm of inverse fraction between document count and document frequency

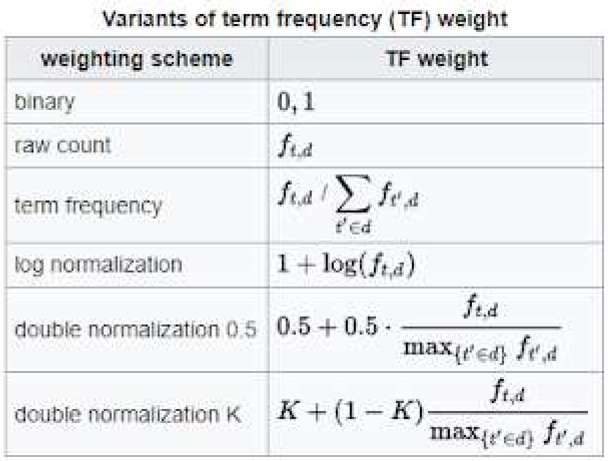


Term document matrix with tf-idf			
words\documents	Document1	document2	query term
cat	0	0	0
runs	0	0	0
behind	0	0	0
rat	0.30103	0	0.30103
dog	0	0.30103	0

tf-idf calculation

Note: Tf-idf weightage is calculated using tf X idf

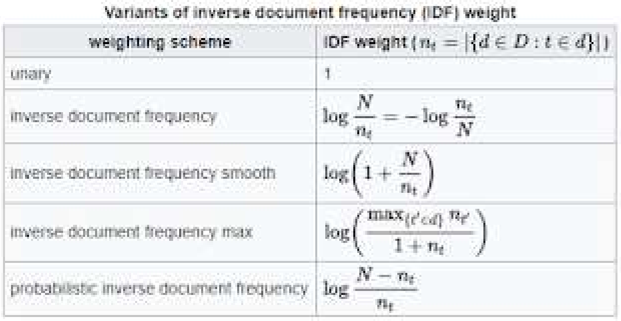
Note, there are many variations in the way we calculate the term-frequency(tf) and inverse document frequency (idf), in this post we have seen one variation. Below images show as the other recommended variations of tf and idf, taken from wiki.



Variants of term frequency (TF) weight

weighting scheme	TF weight
binary	$0, 1$
raw count	$f_{t,d}$
term frequency	$f_{t,d} \Big/ \sum_{t' \in d} f_{t',d}$
log normalization	$1 + \log(f_{t,d})$
double normalization 0.5	$0.5 + 0.5 \cdot \frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$
double normalization K	$K + (1 - K)\frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$

term frequency variations



Variants of inverse document frequency (IDF) weight

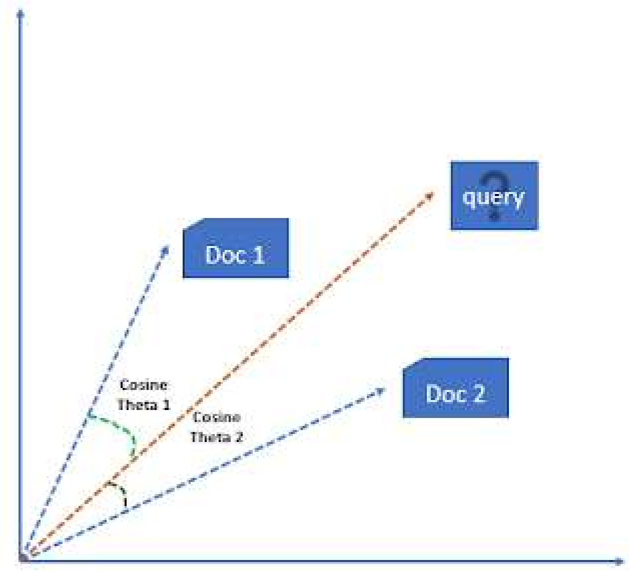
weighting scheme	IDF weight ($n_t = \lvert\{d \in D : t \in d\}\rvert$)
unary	1
inverse document frequency	$\log \frac{N}{n_t} = -\log \frac{n_t}{N}$
inverse document frequency smooth	$\log\left(1 + \frac{N}{n_t}\right)$
inverse document frequency max	$\log\left(\frac{\max_{\{t' \in d\}} n_{t'}}{1 + n_t}\right)$
probabilistic inverse document frequency	$\log \frac{N - n_t}{n_t}$

inverse document frequency variations

Similarity Measures: cosine similarity

Mathematically, closeness between two vectors is calculated by calculating the cosine angle between two vectors. In similar lines, we can calculate cosine angle between each document vector and the query vector to find its closeness. To find relevant document to the query term , we may calculate the similarity score between each document vector and the query term vector by applying cosine similarity . Finally, whichever documents having high similarity scores will be considered as relevant documents to the query term.

When we plot the term document matrix, each document vector represents a point in the vector space. In the below example query, Document 1 and Document 2 represent 3 points in the vector space. We can now compare the query with each of the document by calculating the cosine angle between them.



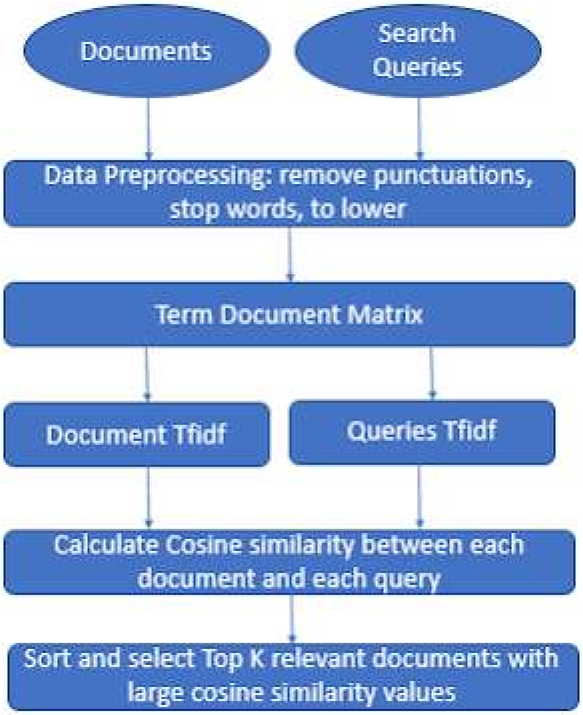
cosine similarity

Apart from cosine similarity, we have other variants for calculating the similarity scores and are shown below:

- Jaccard distance
- Kullback-Leibler divergence
- Euclidean distance

1. Project Logic Representation

- Load documents and search queries into the R programming environment as list objects.
- Preprocess the data by creating a corpus object with all the documents and query terms, removing stop words, punctuations using tm package.


Documents
Search Queries
Data Preprocessing: remove punctuations, stop words, to lower
Term Document Matrix
Document Tfidf
Queries Tfidf
Calculate Cosine similarity between each document and each query
Sort and select Top K relevant documents with large cosine similarity values

high level information retrieval system

- Creating a term document matrix with tf-idf weight setting available in TermDocumentMatrix() method.
- Separate the term document matrix into two parts- one containing all the documents with term weights and other containing all the queries with term weights.
- Now calculate cosine similarity between each document and each query.
- For each query sort the cosine similarity scores for all the documents and take top-3 documents having high scores.

Step 1:

Set up file retrieval using file list box, directory list box, and drive list box

Step 2:

Set up query retrieval text box for user input.

Step 3:

Implement split() function on the query and the selected directory files content to retrieve individual words of each file content.

Step 4:

Implement redundancy removal to filter duplicate words from the above steps.

Step 5:

```
1. Read File Content and Querry

 Invoke Call ReadDataAndQuerry

2. get all the collection of document strings in final text

   Invoke Call ConcatFile

3. initialisations

    donelen = arrlen = qrylen = 0

    4. split(finaltext," ") and get length of splitted string

    Invoke Call wordsintext(finaltext, arr, arrlen)
```

5. split(querry," ") and get length of splitted string

Invoke Call wordsintext(qry, querry, qrylen)

6. sort in alphabetic order A - Z

Invoke Call sort(arr, arrlen)
Call sort(querry, qrylen)

7. Remove redundancy of duplicate words in split array

Invoke Call remduplication(done, donelen, arr, arrlen)

8. quantify terms frequency

Invoke Call termfrequency

9. calculate the document frequency

Invoke Call CalcDFI

10. calculate the document frequency ratio to number of documents

Invoke Call CalcDDFI

11. calculate the IDF Value

Invoke Call CalcIDF

12. calculate the weights of each document Corresponding to the querry

Invoke Call Calcweights

13. Table Orientation of Vector Space Model Matrix

Invoke Call TableOrientation

14. Calculate euclidean length of vector

Invoke Call euclideanvector

15. Calculate Dot vector Values

Invoke Call DotProduct

16. Calculate Cosine Values To determine highest match

Invoke Call CalcCOS

17. Print Vector Model Data into Table Matrix

Invoke Call PrintData

18. File Retrieval Order

Invoke Call FileRetrieve

4. Source Code

```
Dim k As Integer   'local
Dim kp As Integer   'display and hide
Dim qry As String    'querry string
Dim finaltext As String   'collection of all document text
Dim donelen As Integer  'Non repeated words length
Dim arrlen As Integer   'collection of split words in final text
Dim qrylen As Integer   'split words length in querry
Dim terms(1000, 1000) As Double    'table of vector model
Dim arr(10000) As String      'split words of final text
Dim querry(100) As String     'split words of querry
Dim done(10000) As String     'split words of non repeated array
Dim fle(10000, 10000) As String       'file related
Dim flen(10000) As Integer           'file related
Dim filecontent(10000) As String     'file related
Dim rowlen As Integer                'Total columns in matrix

Private Sub Command1_Click()
   Dim i As Integer
   Dim j As Integer

   '1. Read File Content and Querry

   Call ReadDataAndQuerry

   '2. get all the collection of document strings in final text

   Call ConcatFile

   '3. initialisations

   donelen = arrlen = qrylen = 0

   '4. split(finaltext," ") and get length of splitted string

   Call wordsintext(finaltext, arr, arrlen)

   '5. split(querry," ") and get length of splitted string
```

```
Call wordsintext(qry, querry, qrylen)

'6. sort in alphabetic order A - Z

Call sort(arr, arrlen)
Call sort(querry, qrylen)

'7. Remove redundancy of duplicate words in split array

Call remduplication(done, donelen, arr, arrlen)

'8. quantify terms frequency

Call termfrequency

'9. calculate the document frequency

Call CalcDFI

'10. calculate the document frequency ratio to number of documents

Call CalcDDFI

'11. calculate the IDF Value

Call CalcIDF

'12. calculate the weights of each document Corresponding to the querry

Call Calcweights

'13. Table Orientation of Vector Space Model Matrix

Call TableOrientation

'14. Calculate euclidean length of vector

Call euclideanvector

'15. Calculate Dot vector Values
```

```
    Call DotProduct

    '16. Calculate Cosine Values To determine highest match

    Call CalcCOS

    '17. Print Vector Model Data into Table Matrix

    Call PrintData

    '18. File Retrieval Order

    Call FileRetrieve

End Sub

Private Sub Command2_Click()
    Text6.text = ""
End Sub

Private Sub Command3_Click()
    'MsgBox (kp)
    If kp = 0 Then
    table.Visible = True
    kp = 1
    Else
    table.Visible = False
    kp = 0
    End If
End Sub

Private Sub Dir1_Change()
File1.Path = Dir1.Path
End Sub

Private Sub Drive1_Change()
Dir1.Path = Drive1.Drive
End Sub
```

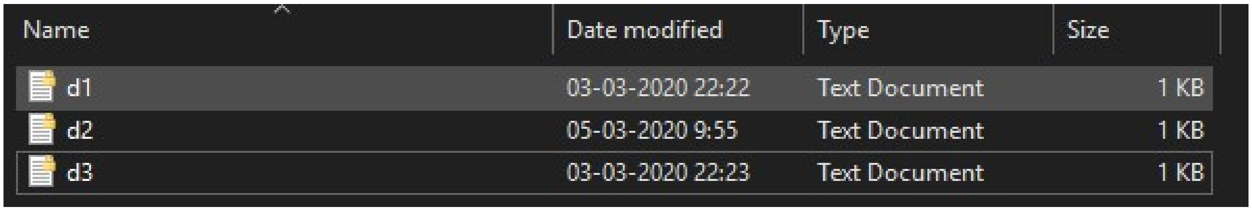
```
Public Sub wordsintext(text As String, ByRef arr() As String, ByRef n As Integer)
    Dim ln As Integer
    Dim i As Integer
    Dim j As Integer
    i = 1
    j = 0
    text = Trim(text)
    ln = Len(text)
    Dim cnt As Integer
    cnt = 0
    Do While i <= ln
        Dim ch As String
        ch = Mid(text, i, 1)
        If ch <> " " And ch <> "." Then
            cnt = cnt + 1
        Else
            If Mid(text, i - cnt, cnt) <> "" And Mid(text, i - cnt, cnt) <> " " And Mid(text, i - cnt, cnt)
<> "  " Then
            arr(j) = Mid(text, i - cnt, cnt)
            j = j + 1
            cnt = 0
            End If
        End If
        i = i + 1
    Loop
    If Mid(text, i - cnt, cnt) <> "" And Mid(text, i - cnt, cnt) <> " " And Mid(text, i - cnt, cnt) <> "
" Then
    arr(j) = Mid(text, i - cnt, cnt)
    End If
    n = j
End Sub


Public Sub wordsintext1(text As String, ByVal k As Integer, ByRef n As Integer)
    Dim ln As Integer
    Dim i As Integer
    Dim j As Integer
    i = 1
    j = 0
    text = Trim(text)
    ln = Len(text)
    Dim cnt As Integer
    cnt = 0
    Do While i <= ln
        Dim ch As String
        ch = Mid(text, i, 1)
        If ch <> " " And ch <> "." Then
            cnt = cnt + 1
        Else
            If Mid(text, i - cnt, cnt) <> "" And Mid(text, i - cnt, cnt) <> " " And Mid(text, i - cnt, cnt)
<> "  " Then
            fle(k, j) = Mid(text, i - cnt, cnt)
            j = j + 1
            cnt = 0
            End If
        End If
        i = i + 1
    Loop
    If Mid(text, i - cnt, cnt) <> "" And Mid(text, i - cnt, cnt) <> " " And Mid(text, i - cnt, cnt) <> "
" Then
    fle(k, j) = Mid(text, i - cnt, cnt)
    End If
    n = j
End Sub

Public Sub sort(ByRef arr() As String, n As Integer)
Dim i As Integer
    Dim j As Integer
    For i = 0 To n - 1
        For j = 0 To n - i - 1
        If arr(j) <> "" And arr(j) <> " " And arr(j) <> "  " And arr(j + 1) <> "" And arr(j + 1) <> " "
And arr(j + 1) <> "  " Then
            If Len(arr(j)) > Len(arr(j + 1)) Then
                Dim temp As String
                temp = arr(j)
                arr(j) = arr(j + 1)
                arr(j + 1) = temp
            End If
        End If
        Next j
    Next i
    For i = 0 To n - 1
        For j = 0 To n - i - 1
        If arr(j) <> "" And arr(j) <> " " And arr(j) <> "  " And arr(j + 1) <> "" And arr(j + 1) <> " "
And arr(j + 1) <> "  " Then
            If Asc(arr(j)) > Asc(arr(j + 1)) Then
                temp = arr(j)
                arr(j) = arr(j + 1)
                arr(j + 1) = temp
            End If
        End If
        Next j
    Next i
End Sub

Public Sub remduplication(ByRef done() As String, ByRef j As Integer, arr() As String, n As
Integer)
    Dim i As Integer
    For i = 0 To n
        If i = 0 Then
            j = 0
            done(j) = arr(i)
            j = j + 1
        End If                              ' a a b c c d e f f
            Dim k As Integer
            For k = 0 To j
                If arr(i) = done(k) Then
                    Exit For
                End If
            Next k
            If k = j + 1 Then
                done(j) = arr(i)
                j = j + 1
            End If
    Next i
End Sub

Public Sub termfrequency()
Dim i As Integer
    Dim j As Integer
    For i = 0 To donelen - 1
        For j = 0 To File1.ListCount - 1
        Dim k As Integer
            For k = 0 To flen(j)
                If done(i) = fle(j, k) Then
                    terms(i, j) = terms(i, j) + 1
                End If
            Next k
        Next j
        If j = File1.ListCount Then
                For k = 0 To qrylen
                    If done(i) = querry(k) Then
                        terms(i, j) = terms(i, j) + 1
                    End If
                Next k
            End If
    Next i
End Sub


Public Sub sortnum(ByRef ar() As Double, n As Integer, ByRef cos() As String)
Dim temp As String
Dim tempk As Double
    For i = 0 To n - 2
        For j = 0 To n - i - 2
            If ar(j) < ar(j + 1) Then
                temp = cos(j)
                cos(j) = cos(j + 1)
                cos(j + 1) = temp
                tempk = ar(j)
                ar(j) = ar(j + 1)
                ar(j + 1) = tempk
            End If
        Next j
    Next i
End Sub

Private Sub Form_Load()
    File1.Path = "C:\Program Files (x86)\Microsoft Visual Studio\VB98\docsam"
    Dir1.Path = "C:\Program Files (x86)\Microsoft Visual Studio\VB98\docsam"
    table.Visible = False
    kp = 0
End Sub

Public Sub CalcDFI()
Dim i As Integer
Dim j As Integer
    For i = 0 To donelen - 1 'DFI
        terms(i, File1.ListCount + 1) = 0
        For j = 0 To File1.ListCount - 1
            If terms(i, j) <> 0 Then
            terms(i, File1.ListCount + 1) = terms(i, File1.ListCount + 1) + 1
            End If
        Next j
    Next i
End Sub

Public Sub CalcDDFI()
Dim i As Integer
    For i = 0 To donelen - 1 'D/DFI D:no of documents 3
        terms(i, File1.ListCount + 2) = Round(3 / terms(i, File1.ListCount + 1), 2)
    Next i
End Sub

Public Sub CalcIDF()
Dim i As Integer
    For i = 0 To donelen - 1 'IDF = Log(D/dfi)
        terms(i, File1.ListCount + 3) = Round(Log(terms(i, File1.ListCount + 2)) / Log(10), 4)
    Next i
End Sub

Public Sub Calcweights()
Dim i As Integer
Dim j As Integer
    rowlen = ((File1.ListCount + 1) * 2) + 3
    For i = 0 To donelen - 1
        For j = File1.ListCount + 4 To rowlen - 1
            terms(i, j) = Round(terms(i, j - 4 - File1.ListCount) * terms(i, File1.ListCount + 3), 4)
        Next j
    Next i
```

```
End Sub

Public Sub TableOrientation()
Dim i As Integer
Dim j As Integer
   table.Rows = donelen + 4
   table.Cols = rowlen + 1
   table.TextMatrix(0, 0) = " "
   For j = 1 To File1.ListCount
      table.TextMatrix(0, j) = "D" + Str(j)
   Next j
   table.TextMatrix(0, File1.ListCount + 1) = "Q"
   table.TextMatrix(0, File1.ListCount + 2) = "DFI"
   table.TextMatrix(0, File1.ListCount + 3) = "D/DFI"
   table.TextMatrix(0, File1.ListCount + 4) = "IDF"
   i = 1
   For j = File1.ListCount + 5 To rowlen - 1
   table.TextMatrix(0, j) = "WD" + Str(i)
   i = i + 1
   Next j
   table.TextMatrix(0, rowlen) = "WDQ"
   For i = 1 To donelen
   table.TextMatrix(i, 0) = done(i - 1)
   Next i
   table.TextMatrix(donelen + 1, 0) = "eucledian vector"
   table.TextMatrix(donelen + 2, 0) = "DOT product"
   table.TextMatrix(donelen + 3, 0) = "cos theta"
End Sub

Public Sub euclideanvector()
Dim i As Integer
Dim j As Integer
   For j = File1.ListCount + 4 To rowlen - 1
   terms(donelen, j) = 0
      For i = 0 To donelen - 1
         terms(donelen, j) = Round(terms(donelen, j) + (terms(i, j) ^ 2), 4)
      Next i
   terms(donelen, j) = Round(Sqr(terms(donelen, j)), 4)
   Next j
End Sub
```

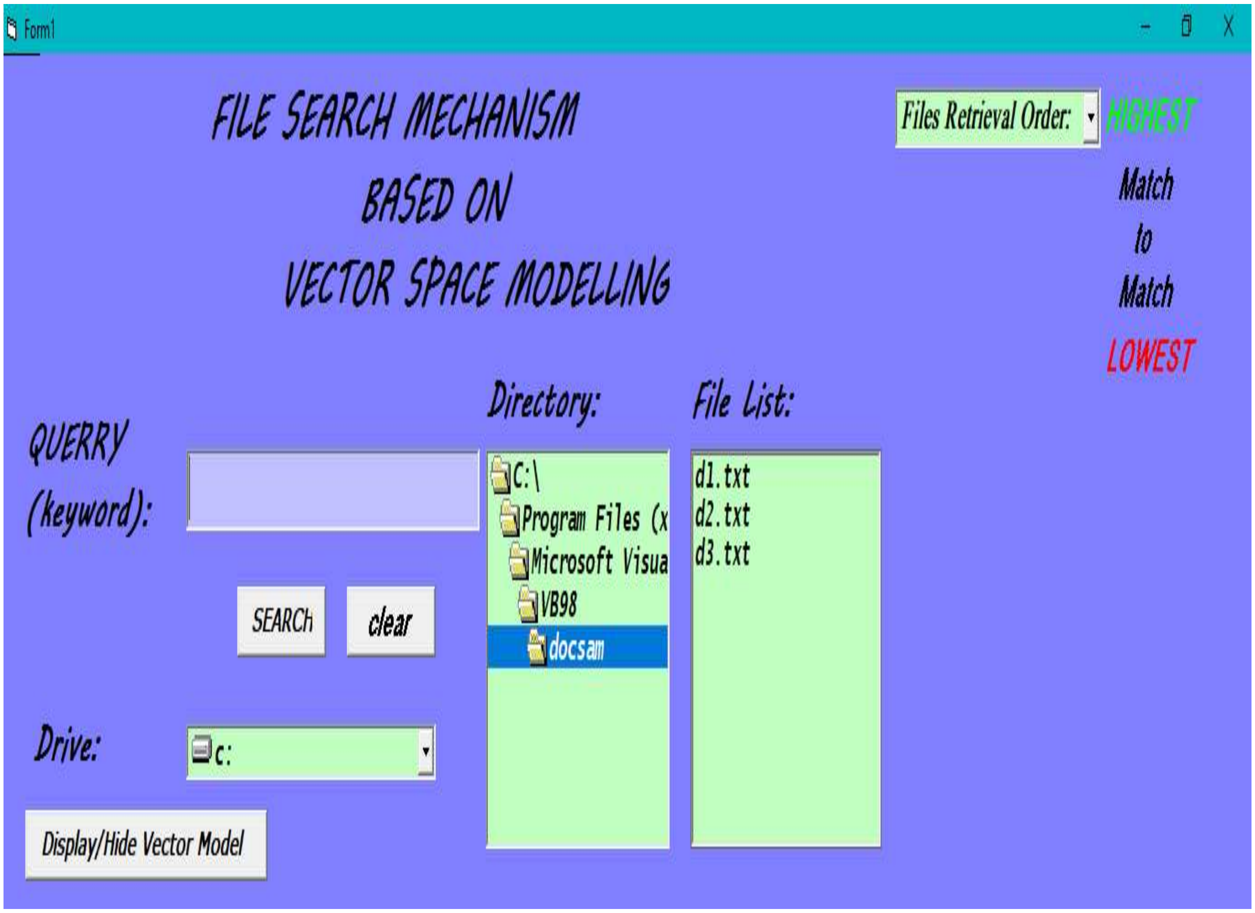
```
Public Sub DotProduct()
Dim i As Integer
Dim j As Integer
   For j = File1.ListCount + 4 To rowlen - 2
      terms(donelen + 1, j) = 0
      For i = 0 To donelen - 1
         terms(donelen + 1, j) = terms(donelen + 1, j) + terms(i, j) * terms(i, rowlen - 1)
      Next i
   terms(donelen + 1, j) = Round(terms(donelen + 1, j), 4)
   Next j
End Sub

Public Sub CalcCOS()
Dim j As Integer
   For j = File1.ListCount + 4 To rowlen - 2
      terms(donelen + 2, j) = Round(terms(donelen + 1, j) / (terms(donelen, rowlen - 1) *
terms(donelen, j)), 4)
   Next j
End Sub

Public Sub PrintData()
Dim i As Integer
Dim j As Integer
   For i = 1 To donelen + 3
      For j = 1 To rowlen
         table.TextMatrix(i, j) = Str(terms(i - 1, j - 1))
      Next j
   Next i
End Sub

Public Sub FileRetrieve()
   Dim i As Integer
   Dim j As Integer
   Dim cosval(10000) As Double
   Dim cosstring(10000) As String
   i = 0
   For j = File1.ListCount + 4 To rowlen - 2
      cosval(i) = terms(donelen + 2, j)
      cosstring(i) = table.TextMatrix(0, j + 1)
```

```
        i = i + 1
    Next j
    Call sortnum(cosval, File1.ListCount, cosstring)
    For i = 0 To File1.ListCount - 1
        Dim y As Integer
        y = Val(Mid(cosstring(i), 4, 1))
        Combo1.AddItem File1.List(y - 1), i
    Next i
End Sub


Public Sub ConcatFile()
Dim i As Integer
    For i = 0 To File1.ListCount - 1
    finaltext = finaltext + filecontent(i) + " "
    Next i
End Sub

Public Sub ReadDataAndQuerry()
    Dim i As Integer
    For i = 0 To File1.ListCount - 1
        Dim strFile As String
        'With CD
        '    .ShowOpen
        '   Text1.Text = .FileName
        'End With
        Open File1.Path + "\" + File1.List(i) For Input As #1
        Do While Not (EOF(1))
            Line Input #1, strFile
            filecontent(i) = filecontent(i) + " " + strFile
        Loop
        Call wordsintext1(filecontent(i), i, flen(i))
        Close #1
    Next i
    qry = Text6.text
End Sub
```

Output



Name	Date modified	Type	Size
d1	03-03-2020 22:22	Text Document	1 KB
d2	05-03-2020 9:55	Text Document	1 KB
d3	03-03-2020 22:23	Text Document	1 KB



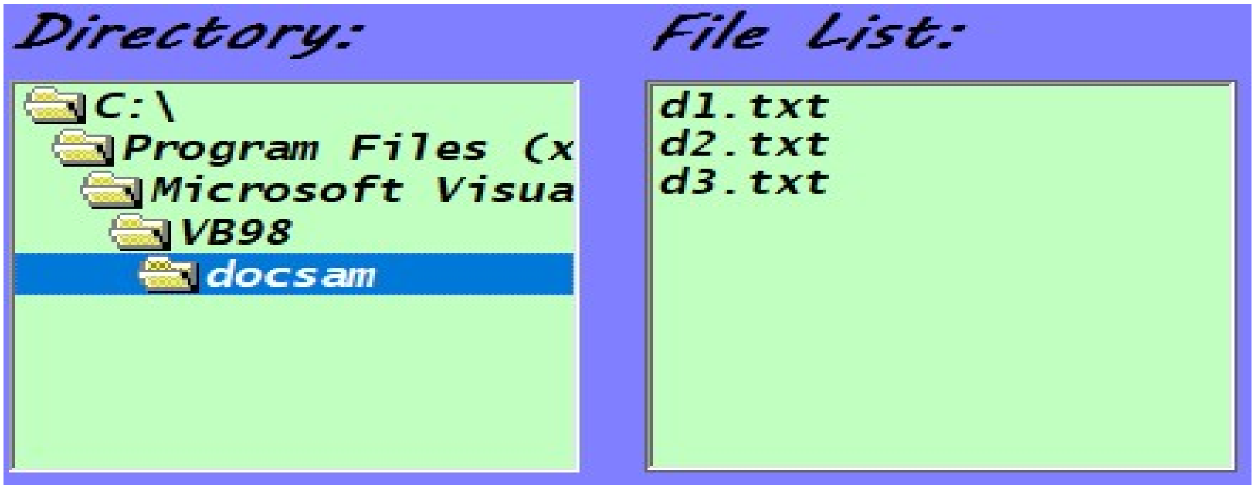
Drive: c:

Directory:

```
C:\
  Program Files (x
    Microsoft Visua
      VB98
        docsam
```

File List:

```
d1.txt
d2.txt
d3.txt
```

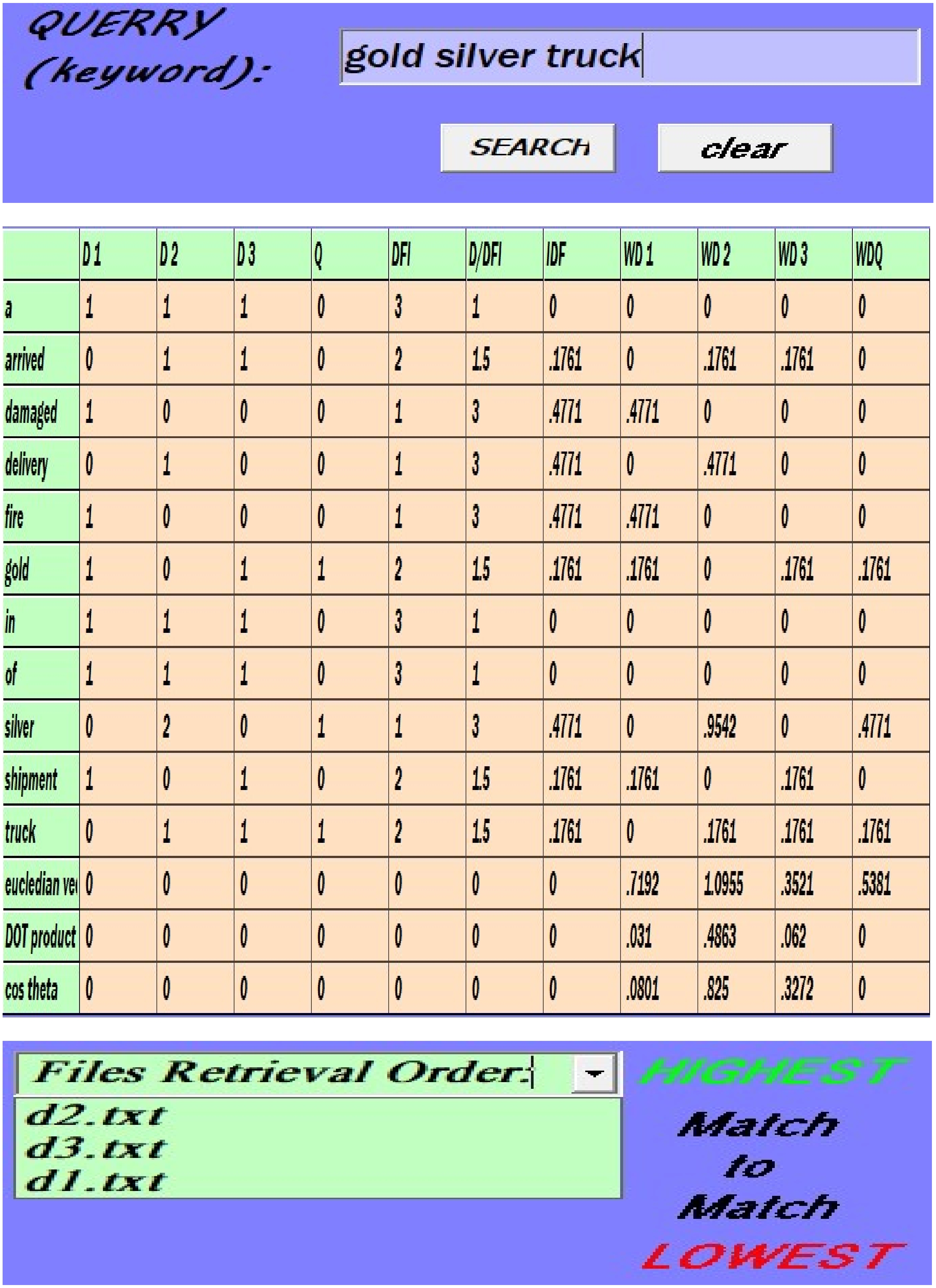


Directory:

```
C:\
  Program Files (x
    Microsoft Visua
      VB98
        docsam
```

File List:

```
d1.txt
d2.txt
d3.txt
```



gold silver truck

	D 1	D 2	D 3	Q	DFI	D/DFI	IDF	WD 1	WD 2	WD 3	WDQ
a	1	1	1	0	3	1	0	0	0	0	0
arrived	0	1	1	0	2	1.5	.1761	0	.1761	.1761	0
damaged	1	0	0	0	1	3	.4771	.4771	0	0	0
delivery	0	1	0	0	1	3	.4771	0	.4771	0	0
fire	1	0	0	0	1	3	.4771	.4771	0	0	0
gold	1	0	1	1	2	1.5	.1761	.1761	0	.1761	.1761
in	1	1	1	0	3	1	0	0	0	0	0
of	1	1	1	0	3	1	0	0	0	0	0
silver	0	2	0	1	1	3	.4771	0	.9542	0	.4771
shipment	1	0	1	0	2	1.5	.1761	.1761	0	.1761	0
truck	0	1	1	1	2	1.5	.1761	0	.1761	.1761	.1761
eucledian ve	0	0	0	0	0	0	0	.7192	1.0955	.3521	.5381
DOT product	0	0	0	0	0	0	0	.031	.4863	.062	0
cos theta	0	0	0	0	0	0	0	.0801	.825	.3272	0

Files Retrieval Order:

d2.txt
d3.txt
d1.txt

Conclusion

The project entitled '**VECTOR SPACE MODEL FILE SEARCH**' is completed successfully with the collective effort of both involved in the project. The project has successfully overcome all the errors pertaining in the existed system. With the help of this project,

- We can retrieve files based on key query
- Efficient search engines for files , items or objects in various fields
- Order of retrieval emanates a sequence of a number of possible files and items to select from.
- Allows documents with partial match to be also included.
- The cosine formula gives a score which can be used to order documents.

The overall goal of the field of Visual Programming is achieved, and same implementation techniques can be used in other programming projects.

Bibliography

Algorithm and heuristics by Dr.David Grossman and Dr.ophir Frieder of the Illinois institute of technology.