Text Summarization for Information Retrieval using Pattern Recognition Techniques

Pritam Singh Negi
Department of Computer Science
H.N.B. Garhwal University
Srinagar Garhwal, India

M.M.S. Rauthan
Department of Computer Science
H.N.B. Garhwal University
Srinagar Garhwal, India

H.S. Dhami
Director, Information
Communication Technology
Kumaun University, Nainital

ABSTRACT
In the present work a model is proposed which is useful for text summarization of the given document by using pattern recognition techniques for improving the retrieval performance of the relevant information. The design and implementation of the proposed systems is concerned with methods for summarizing of the retrieving information from a collection of documents or corpuses. The quality of a system is measured by how useful it is to the typical users of the system. In the basic approach, a query is considered generated from an "ideal" document that satisfies the information need. The system's job is then to estimate the likelihood of each document in the collection being the ideal document and rank them accordingly. The recent development of related techniques stimulates new modeling and estimation methods that are beyond the scope of the traditional approaches.
Keywords: Information retrieval, Pattern Recognition, Text summarization, Mathematical Model.

1. INTRODUCTION
The goal of proposed model is to identify documents relevant to a user's query. In order to do this, system must assume some measure of relevance between a document and a query, i.e., an *operational* definition of a relevant document with respect to a query. A fundamental problem in this model is thus to formalize the concept of relevance; a different formalization of relevance generally leads to a different model. Over the decades, many different retrieval models have been proposed, studied, and tested. Their mathematical basis spans a large spectrum, including algebra, logic, probability and statistics. In this model, we can calculate the probability of the pattern of the document on the basis of part of speech and other conditions which are relevant to the query.

2. LOGIC
For calculating the relevance of the document we will first define and construct identifiers, those are basically based on grammatical aspects of English language pertaining to which we have to retrieve information from the given text. We are interested to retrieve the information from the corpus of English language regarding to their tense form, voice form, speech form, etc. So firstly we have to construct the proper identifiers. As we know a sentence of English language can be found in a particular tense, voice, speech etc. So we can classify all these sentences into particular classes for which we have need to construct some identifiers pertaining to these particular classes.
Example:

1. Class of TENSE
2. Class of VOICE

Given text is firstly divided into the documents; here our mean to a document is a single sentence with the help of sentence boundary detection. After then we construct a matrix whose every column shows a particular sentence and the entries of such column are according to the presence and absence of the identifiers. Thus these all documents (sentences) are arranged in a matrix, whose number of rows and columns are equal to the number of identifiers (t) and documents (d) respectively. This matrix is thus said to be term by document matrix of order t×d.
The presence and absence of a particular identifier in a document is defined as
$T_i D_j = 1$; if i^{th} identifier present in the j^{th} document Otherwise the value of $T_i D_j$ is 0.
Thus in general the sentence of different grammatical aspects can be added together and gives any other sentence of the text comprising all aspects simultaneously, provided that they don't belong to the same class.

2.1 TENSE RECOGNIZING IDENTIFIERS
T1= verb/verb-es/(is/am/are) verb (3^{rd} form)
T2= is/am/are verb+ing/is/am/are being verb (3^{rd} form)
T3= has/have verb (3^{rd} form)/ has/ have been verb (3^{rd} form)
T4= has/ have been ver+ing
T5=verb (2^{nd} form)/ was/ were verb (3^{rd} form)
T6=was/were verb+ing/ was/were being verb (3^{rd} form)
T7=had verb (3^{rd} form)/ had been verb (3^{rd} form)
T8= had been verb+ing
T9= will/ shall verb (1^{st} form)/will/shall be verb (3^{rd} form)
T10= will/shall be verb+ing
T11= will/shall have verb (3^{rd} form)/ will/shall have been verb (3^{rd} form)
T12=will/shall have been verb+ing form

MATHEMATICAL FORMALISM:

These 12 identifiers can be used for dividing entire English language corpus into 12 different tenses. Any sentence exists exactly one of the forms of above given 12 identifiers according to its tense form.
As:
Ram *plays* hockey in the field.
Its equivalent document vector can be written as:
(1,0,0,…,0(up to 12 terms))
Shyam *passed* his exam with good marks.
(0,0,0,1,0,….(up to 12 terms))

2.2 VOICE RECOGNIZING IDENTIFIERS

In this phase the identifiers are defined in such a manner that they will recognize the voice form in a corpus. We know there exist two types of voice forms, one is active and second is passive form. To identify weather a sentence is in an active voice or passive voice, we have to go through the following procedure by taking two following identifiers:

2.2.1: ACTIVE VOICE:

Verb (1st form), Verb(+es form),
Is/ am/ are ing- form,
Has/ have verb(3rd form)
Has been/ Have been verb(ing- form)

Verb (2nd form)
Was/ were verb(ing-form)
Had verb (3rd form)
Had been verb(ing form)

Will/ shall verb (1st form)
Will be/ shall be verb(ing-form)
Will have/ shall have verb (3rd form)
Will have been/ Shall have been verb(ing- form)

2.2.2: PASSIVE VOICE:

Is/ am/ are verb (3rd form)
Is/ am/ are being verb (3rd form)
Has/ have been verb (3rd form)

Was/ were verb (3rd form)
Was/ were being verb (3rd form)
Had been verb (3rd form)

Will/ Shall be verb(3rd form)
Will have been/ Shall have been verb (3rd form)

By using these two identifiers we can separate simple sentences into two broad categories of active and passive voice. This is obvious that any simple sentence can exist in one of the form either active or passive at a time.

MATHEMATICAL FORMALISM:

We can now formalize any sentence into document vector, whose first coordinate is active and the second one represents the class of passive voice.
For example:
"Jack *has played* a match" can be represented as *(1,0)*, since it is in active voice.

"The deer *has been killed* by the lion", can be represented as *(0,1)*, since it is in passive voice.
On the basis of above mention conditions we can categorized the sentences. For example if sentence is in Present indefinite and Active voice then the sentence vector is in form,

$D1= (1, 0,.., 1, 0\ (14^{th}\ term))_{1*14}$

Similarly we can calculate the sentence vector of all other sentences which are present into the document and also create a query vector.
On the basis of above mathematical formalism after creating these sentence vectors we check that which types of sentence vector are mostly used on the whole document and count the number of availability of the sentence vector on the whole document. Then compare the query vector with document vector. In which corpus highest query vector is matched this one is the relevant document of the given query.

3. ALGORITHM

3.1 Algorithm For Main( ) Function

This algorithm is used for opening the file where the sentences are stored which one is calculated from a corpus or document with the help of Sentence Boundary Detection Algorithm.

1. Start.
2. Open the file where sentences are placed which one is created by sentence boundary detection algorithm ("sentence.txt").
3. While EOF Repeat Steps 3.1 to Step 3.4
   Input a single sentence from this file.
   Break this sentence into its constituent words and store them in a global string array all_words[].
   Len = length of this string array.
   While count < Len Repeat Step 3.3.1 to Step 3.3.3
   - 3.1 Pass each constituent word to verb_search( ) function.
   - 3.2 If return-value = 1 then
     Input again a new single sentence from the file containing sentences.
   - 3.3 End if
   - 3.4 End While
4. End While
5. Close this file.
6. Call stats ( ) function.
7. End.

3.2 Algorithm For Verb_Search ( ) Function

This algorithm is used for searching the verb in a sentence.

1. Start.
2. Open the file "verb_db.txt" which contains all forms of verbs.
3. While EOF Repeat Step 3.1 to Step 3.5
   3.1 Input a new single line from this file.
   3.2 Search this line for the constituent word passed as parameter from main ( ) function.
   3.3 If constituent word is found, then
   Call form_search ( ) function and pass this line and constituent word as its parameter
   Return (1)
   3.4 End if.
   3.5 End while
4. Close file "verb_db.txt"
5. Return (0)

3.3 Algorithm For Form_Search( )Function

This algorithm is used for searching the form of the verb which one is present in a sentence which one is useful for calculating the tense of the sentences.

1. Start
2. Store the line passed from verb_search ( ) function in a string array.
3. Calculate the index of the constituent word in this string array.
4. If index = 0 then
   Form-number = 0
   Else If index = 1 then
   Form-number = 1
   Else If index = 2 then
   Form-number = 2
   Else If index = 3 then

Form-number = 3
Else If index = 4 then
Form-number = 4
End if
End if
End if
End if
End if

5. Call tense_search ( ) function and pass this form_number & constituent word as its parameter.
6. Stop.

3.4 Algorithm For Tense_Search( ) Function

This algorithm is used for searching the tense and voice of the sentences on the basis of the form number of the verb which one is present in a sentence and calculates the sentence vector of the each sentence.

1. Start.
2. Declare an integer array doc_vec[14] and initialize array elements with a value of 0.
3. For each constituent word , Repeat Step 3.1 to Step 3.8

3.1 If ((form-number = 0) AND (all_words[i-1] != "will" AND all_words[i-1] != "shall"))
Print “Simple present tense in active voice"
End if

3.2 If ((form-number = 0) AND (all_words[i-1] = "will" OR all_words[i-1] = "shall"))
Print “Simple future tense in active voice "
End if

3.3 If(form-number = 1) then
Print "simple past tense in active voice"
doc_vec[4] = 1
doc_vec[12] = 1
End if

3.4 If(form-number = 2) then

3.4.1 If(all_words[i-1] = "is" OR all_words[i-1] = "am" OR all_words[i-1] = "are")
Print " simple present tense in passive voice "
doc_vec[0] = 1
doc_vec[13] = 1
End if

3.4.2 If((all_words[i-1] = "being") AND (all_words[i-2] = "is" OR all_words[i-2] = "am" OR all_words[i-2] = "are")) then
Print "Present continous tense in passive voice"
doc_vec[1] = 1
doc_vec[13] = 1
End if

3.4.3 If(all_words[i-1] = "was" OR all_words[i-1] = "were") then
Print " simple past tense in passive voice "
doc_vec[4] = 1
doc_vec[13] = 1
End if

3.4.4 If((all_words[i-1] = "being") AND (all_words[i-2] = "was" OR all_words[i-2] = "were")) then
Print "Past continous in passive voice”
doc_vec[5] = 1
doc_vec[13] = 1
End if

3.4.5 If(all_words[i-1] = "had") then
Print "past perfect tense in active voice.”
doc_vec[6] = 1
doc_vec[12] = 1
End if

3.4.6 If((all_words[i-1] = "been") AND (all_words[i-2] = "had")) then
Print "past perfect tense in passive voice "
doc_vec[6] = 1
doc_vec[13] = 1

3.4.7 If((all_words[i-1] = "be") AND (all_words[i-2] = "will" OR all_words[i-2] = "shall")) then
Print " simple future tense in passive voice “
doc_vec[8] = 1
doc_vec[13] = 1
End if

3.4.8 If((all_words[i-1] = "have") AND (all_words[i-2] = "will" OR all_words[i-2] = "shall")) then
Print "Future perfect tense in active voice “
doc_vec[10] = 1
doc_vec[12] = 1
Else
if(all_words[i-1] = "have" OR all_words[i-1] = "has") then
Print "present perfect tense in active voice”
doc_vec[2] = 1
doc_vec[12] = 1
End if
End if

3.4.9 If((all_words[i-1] = "been" AND all_words[i-2] = "have") AND (all_words[i-3] = "will" OR all_words[i-3] = "shall")) then
Print" future perfect tense in passive voice \n"
doc_vec[10] = 1
doc_vec[13] = 1
Else
If((all_words[i-1] = "been") AND (all_words[i-2] = "has" OR all_words[i-2] = "have")) then
Print “ present perfect tense in passive voice “
doc_vec[2] = 1
doc_vec[13] = 1
End if
End if
End if

3.5 If(form-number = 3) then

3.5.1 If(all_words[i-1] = "is" OR all_words[i-1] = "am" OR all_words[i-1] = "are")
Print "present continuous tense in active voice“
doc_vec[1] = 1
doc_vec[12] = 1
End if

3.5.2 If(all_words[i-1] = "was" OR all_words[i-1] = "were") then
Print “past continous in active voice”
doc_vec[5] = 1
doc_vec[12] = 1
End if

3.5.3 If((all_words[i-1] = "been") AND (all_words[i-2] = "had")) then
Print "past perfect continuous tense in active voice”
doc_vec[7] = 1
doc_vec[12] = 1
End if

3.5.4 if((all_words[i-1] = "be") AND (all_words[i-2] = "will" OR all_words[i-2] = "shall")) then
Print “future continuous tense in active voice”
doc_vec[9] = 1
doc_vec[12] = 1
End if

3.5.5 If((all_words[i-1] = "been" AND all_words[i-2] = "have") AND (all_words[i-3] = "will" OR all_words[i-3] = "shall")) then
Print " future perfect continuous tense in active voice “
doc_vec[11] = 1
doc_vec[12] = 1
Else If((all_words[i-1] = "been") AND (all_words[i-2] = "has" OR all_words[i-2] = "have")) then
Print " present perfect continuous tense in active voice”
doc_vec[3] = 1
doc_vec[12] = 1
End if
End if
End if

3.6 If(form-number = 4) then
Print " simple present tense in active voice”
doc_vec[0] = 1
doc_vec[12] = 1

3.7 End if

3.8 End For

4. Open file “query_vector.txt”
5. For i = 0 to 14
Write to file doc_vec[i].
i = i + 1
6. End For
7. Close file
8. Stop

On the basis of above mention algorithm we find the result of the text summarization of every sentence in the following manner:

- If the sentence is in Simple present tense and in active voice then the identifier is
  1 0 0 0 0 0 0 0 0 0 0 0 1 0
- If the sentence is in Simple present tense and in passive voice then the identifier is
  1 0 0 0 0 0 0 0 0 0 0 0 0 1
- If the sentence is in Present Continuous tense and in active voice then the identifier is
  0 1 0 0 0 0 0 0 0 0 0 0 1 0
- If the sentence is in Present Continuous tense and in passive voice then the identifier is
  0 1 0 0 0 0 0 0 0 0 0 0 0 1
- If the sentence is in Present Perfect tense and in active voice then the identifier is
  0 0 1 0 0 0 0 0 0 0 0 0 1 0
- If the sentence is in Present perfect tense and in passive voice then the identifier is
  0 0 1 0 0 0 0 0 0 0 0 0 0 1
- If the sentence is in Present Perfect Continuous tense and in active voice then the identifier is
  0 0 0 1 0 0 0 0 0 0 0 0 1 0
- If the sentence is in Present Perfect Continuous and in passive voice then the identifier is
  0 0 0 1 0 0 0 0 0 0 0 0 0 1
- If the sentence is in Simple Past tense and in active voice then the identifier is
  0 0 0 0 1 0 0 0 0 0 0 0 1 0
- If the sentence is in Simple present tense and in passive voice then the identifier is
  0 0 0 0 1 0 0 0 0 0 0 0 0 1
- If the sentence is in Past Continuous tense and in active voice then the identifier is
  0 0 0 0 0 1 0 0 0 0 0 0 1 0
- If the sentence is in Past Continuous tense and in passive voice then the identifier is
  0 0 0 0 0 1 0 0 0 0 0 0 0 1
- If the sentence is in Past Perfect tense and in active voice then the identifier is
  0 0 0 0 0 0 1 0 0 0 0 0 1 0
- If the sentence is in Past perfect tense and in passive voice then the identifier is
  0 0 0 0 0 0 1 0 0 0 0 0 0 1
- If the sentence is in Past Perfect Continuous tense and in active voice then the identifier is
  0 0 0 0 0 0 0 1 0 0 0 0 1 0
- If the sentence is in Past Perfect Continuous and in passive voice then the identifier is
  0 0 0 0 0 0 0 1 0 0 0 0 0 1
- If the sentence is in Simple Future tense and in active voice then the identifier is
  0 0 0 0 0 0 0 0 1 0 0 0 1 0
- If the sentence is in Simple Future tense and in passive voice then the identifier is
  0 0 0 0 0 0 0 0 1 0 0 0 0 1
- If the sentence is in Future Continuous tense and in active voice then the identifier is
  0 0 0 0 0 0 0 0 0 1 0 0 1 0
- If the sentence is in Future Continuous tense and in passive voice then the identifier is
  0 0 0 0 0 0 0 0 0 1 0 0 0 1
- If the sentence is in Future Perfect tense and in active voice then the identifier is
  0 0 0 0 0 0 0 0 0 0 1 0 1 0
- If the sentence is in Future perfect tense and in passive voice then the identifier is
  0 0 0 0 0 0 0 0 0 0 1 0 0 1
- If the sentence is in Future Perfect Continuous tense and in active voice then the identifier is
  0 0 0 0 0 0 0 0 0 0 0 1 1 0
- If the sentence is in Future Perfect Continuous and in passive voice then the identifier is
  0 0 0 0 0 0 0 0 0 0 0 1 0 1

3.5 Algorithm For Stats( ) Function

This algorithm is used for calculating the number of the same type of the sentences present in the document file.

1. Start
2. Declare array s_arr[12] , av[14] , pv[12] , f_arr[14]
3. Open file “query_vector.txt”
4. While EOF repeat Step 4.1 to Step 4.4

4.1 Input a new single line from this file.

4.2 Insert this line into array s_arr[ ]

4.3 For I = 0 to 11
If s_arr[I] = 1
f_arr[I] = f_arr[I] + 1
If s_arr [12] = 1
av [I] = av [I] + 1
Else
pv [I] = pv [I] + 1
End if

End if
Next I
4.4 End for
5. End While
6. Close file “query_vector.txt”
7. Declare a string array s2[14] = { "simple present tense" , "present continuous tense" , "present perfect tense" , "present perfect continuous tense" , "simple past tense" , "past continuous tense" , "past perfect tense" , "past perfect continuous tense" , "simple future tense" , "future continuous tense" , "future perfect tense" , "future perfect continuous tense" , "active voice" , "passive voice"}
8. Use Bubble Sort technique to sort s2 [14] and f_arr [14] accordingly in descending order.
9. Open a file “results.txt”
10. For k = 0 to 13
Write the sorted array f_arr [k] onto file.
Next k
11. End for
12. Close file “results.txt”
13. Stop

4. CONCLUSION

We have presented a system for text summarization of a given document. With the help of this system identifiers are defined and constructed, which are basically based on grammatical aspects of English language and these identifiers are useful for retrieving the relevant information from the given text. We created an open source software tool and the experimental results show that the approach can achieve a high accuracy. With the help of this system anyone can understand the writing style of any author. This is an elementary task for calculating the writing style of the author and the summarization of the document. On the basis of proposed system we can also calculate the relevance of the document.

5. REFERENCES

[1] Borlund, P. (2003). The concept of relevance in IR. Journal of the American Society for Information Science and Technology, 54(10), 913–925.

[2] Budd, J.M. (2004) Relevance: Language, semantics, philosophy. Library Trends, 52(3), 447–462.

[3] Mares, E. (1998). Relevance logic. In Stanford Encyclopedia of Philosophy. Retrieved October 17, 2005, from http://plato.stanford. edu/entries/logic-relevance/#Bib

[4] Negi Pritam Singh, Rauthan M. M. S. & Dhami H. S., (2010), Sentence Boundary Disambiguation: A User Friendly Approach, International Journal of Computer Applications (0975 – 8887), Volume 7– No.8, October 2010

[5] Rieh, S.Y., & Xie, H.I. (2006). Analysis of multiple query reformulations on the Web: The interactive information retrieval context. Information Processing & Management, 42(3), 751–768.

[6] Ruthven, I. (2005). Integrating approaches to relevance. In A. Spink & C. Cole (Eds.), New directions in cognitive information retrieval (pp. 61–80). Amsterdam: Springer.

[7] Ruthven, I. (2005). Integrating approaches to relevance. In A. Spink & C. Cole (Eds.), New directions in cognitive information retrieval (pp. 61–80). Amsterdam: Springer.

[8] Saracevic, T. (2006). Relevance: A review of and a framework for the thinking on the notion of information science. Part II. In D.A. Nitecki & E.G. Abels (Eds.), Advances in Librarianship (Vol. 30, pp. 3–71). San Diego: Academic Press.

[9] Saracevic, T. (2007). Relevance: A review of the literature and a framework for thinking on the notion in information science. Part III: Behavior and effects of relevance. Journal of the American Society for Information Science and Technology, 58, 2126–2144.

[10] Zipf, G. (1949). Human behavior and the principle of least effort. Cambridge, MA: Addison-Wesley.