International Journal of
Emerging Research in Management &Technology
ISSN: 2278-9359 (Volume-4, Issue-3)

Research Article | **March 2015**

Automatic Query Expansion

C. R Anjali Krishna, M. B Arya, P. N Neeraja, A Nikhila, K. M Sreevidya, M. Aparna[#]
(#Asst. Professor)
Dept. of Computer Science & Engineering,
Sreepathy Institute of Management and Technology,
Palakkad, Kerala, India

Abstract—

Automatic query expansion system allows users to impel and order search results in an instinctive manner by focusing a query on a particular geographic region. Information retrieval system purposes to find the useful documents wanted by users. The major problem of information retrieval is ambiguity of language. In most of the cases user query will be in natural language, and the information need will be specified vaguely. AQE try to fill this gap by adding new terms (identified by local analysis of the document collection) into the original query terms. This modified query is used to retrieve the documents. In this paper, we include the major works that we have done on the design of an Automatic query expansion system starting from the design of a simple search engine. We are using vector space model to represent the documents and query. Automatic local cluster analysis is used for query expansion.

Keywords— AQE, Index file, Inverted File, Document, Query.

I. INTRODUCTION

Information retrieval (IR) is the retrieval of information (not data) from a collection of documents. The retrieved documents aim at satisfying a user information need usually expressed in natural language. There are several ways to improve the search effectiveness. Query expansion is one such approach that improves effectiveness of the search. Query expansion is the process of adding some new terms to the original query to improve the retrieval performance. AQE is the process of automatically boosting additional terms or phrases to the original query and is considered an extremely promising technique to improve the retrieval effectiveness. We are using vector space model to represent the documents and query. In vector space model both documents and query are considered as a vector of weight values. Weight value is considered as the product of term frequency and inverse document frequency. The vector model takes into consideration documents which match the query terms partially.

II. SYSTEM OVERVIEW

Automatic query expansion system is based on the domain "geography". It deals with the geographic features of Kerala. It allows local search of documents. A mini search engine is created. Documents for the search engine are collected from Wikipedia and other web pages. An offline approach of searching is done. Designed system is dynamic i.e. it support the addition of new documents also. In AQE documents are collected and stored in a folder.
This system consists of mainly 4modules:

1. A Document Pre-processor.
2. Index Generator.
3. Query Processor and Ranker.
4. Automatic Query Expansion.

The first two modules are used for the index creation of documents which are stored in the document set. The main purpose of this module is to generate the index file or the inverted file for each document in the document set. The inverted file will be a global array. We can view the inverted file as a "term X document" matrix, where each row contains the terms in documents. And the columns takes the ID's of each documents in the document set. The entries of the matrix will be the weight value of a particular term in a particular document. Weight value is calculated by using the formulae of term frequency * inverse document frequency. The term frequency describes the term weight in a document itself. Term frequency is the number of occurrences of a term "t" in a document "d", denoted as tf (t, d). Inverse document frequency represents the weight of a term in the entire document collection. It will be the logarithmic ratio of total number of documents in the document set to that of number of documents where the term appears.

The elements for “term X document" matrix is determined by a series of steps and it includes splitting the documents in to words, removing the punctuations and stop words. And finally perform stemming on the obtained results. For each stemmed word, its term frequency and inverse document frequency is calculated and add the term along with its weight value to the global array. The resulted array will be called the inverted file. Weight of each term in all documents is calculated. Usually this task is dynamic it means that when a new document came it automatically create the index file by including that document.

The third module finds the vocabulary corresponding for the user inputted query. Vocabulary is formed by a series of steps such as tokenization, stop word removal, and stemming. For each term in the vocabulary systems checks in what all documents the term appears and that documents are retrieved and displayed to the user.

Fourth module is used to improve the retrieval performance. After completing the third step certain documents corresponding to the user query are retrieved and displayed to the user. These retrieved documents are known as local document set. From these local documents another "term X document" matrix is constructed. Inverse of that matrix is calculated it will be a "document X term" matrix. Both these matrixes are multiplied. The resulting will be a "term X term" matrix, which represents the co-relation of a word with that of another word. Based on that matrix, query expansion is done. It works by adding the column term to that of the row term whose cell value in the matrix is greater than the threshold. And this modified query is again given as the input to the search engine to retrieve the documents.

III. DOCUMENT PRE-PROCESSING

A. Document Set Details

"Kerala Geography" is chosen as the project domain. A document set containing 500 text files is formed. Transportation, Forest, Water bodies, Places, Hills and valleys are the major geographical features that we consider for AQE. Only textual documents are considering.

B. Steps in Document Pre-Processing

This module generates the global vocabulary array for the documents stored in the document set. The major steps involves in the vocabulary array construction are:

1. Tokenize each document.
2. Remove stop words.
3. Perform stemming.
4. Generate the global vocabulary array.

This mainly deals with how the token is determined, how stop word removal and stemming is done.

C. Tokenization

This is the first step in indexing. Tokenization module identifies the keywords corresponding to each document in the document set. Keywords are usually referred to as tokens or vocabulary terms. Given a character sequence and a defined document unit, tokenization is the task of chopping it up into pieces, called tokens, perhaps at the same time throwing away certain characters, such as punctuation. In AQE first of all the raw text is split to form sentences and each sentence is further divided in to words. Splitting of the raw text into sentence is based on sentence segmenter. We maintain a punctuation list that contains the punctuations like comma, semicolon, colon, dash, hyphen, brackets, braces, parentheses, apostrophe, quotation marks etc. While splitting the sentence in to words we check whether the currently considering word belongs to the punctuation list, if it belongs then that word is not further considered. In AQE chopping the document is based on whitespace and punctuation.

D. Dropping Common Terms: Stop Words

Sometimes, some extremely common words which would appear to be of little value in helping select documents matching a user need are excluded from the vocabulary entirely. These words are called stop words. In our project we maintain a stop word file which contains around 200 commonly seen stop words. The token obtained from the previous step is compared with the stop word file. When a match occurs corresponding token is removed otherwise it is given as the input to the stemmer. This technique helps us to improve the system resource. Our stop word list includes commonly used adjectives, connectives, verbs and certain other words.

Stopwords		
a	it	these
about	its	they
again	itself	this
all	just	those
almost	kg	through
also	km	thus
although	made	to
always	mainly	upon
among	make	use
an	may	used
and	mg	using
another	might	various
any	ml	very
are	mm	was
as	most	we
at	mostly	were

Fig.1 Stopword list

E. Stemming

Usually the same word with different suffix form (ing, es, ies,) are seen in same document or different document. So storing each word separately is not an efficient method. To avoid this the words are stemmed to its root value. This task is done by the stemmer. We are using porter stemmer algorithm. The below shown example shows the output of a sample text which undergoes stemming:

Sample text: Such an analysis can reveal features that are not easily visible from the variations in the individual genes and can lead to a picture of expression that is more biologically transparent and accessible to interpretation.

Porter stemmer: Such an analyi can reveal featur that ar not easili visibl from th variat in the individu gene and can lead to a pictur of express that is more biology transpar and access to interpres.

IV. INDEX GENERATION

This is the second part of automatic query expansion (AQE). The first module of AQE is document pre-processing, combining first and second module we can construct an index file for the documents. The last step of document pre-processor is stemming.The stemmed words are added to the global vocabulary array. For each stemmed words in global array the term frequency and inverse document frequency is calculated. The weight will be equal to the product of term frequency and inverse document frequency. Usually the term is considered as a vector where each value in vector represent the weight of that term in a particular document. The weight value will be in between 0 and 1. All terms in global array are added along with their weight vector to the index file. Usually this task is dynamic it means that when a new document came it automatically create the index file including that document. Also we can view the inverted file as a two dimensional array where each row represent the vocabulary terms and the columns represent the weight value of a particular term in a particular document.

Consider 6 documents for this process.

D1) rose and Jasmin are beautiful flowers in the world.

D2) rose is seen in kerala.

D3) Jasmin are usually used in marriage occasions.

D4) Jasmin is white in colour.

D5) Rose is seen in several colours including white red and blue

D6) I want a Lilly flower in kerala.

A. Term Frequency and Weighting (tf)

In the vector model intra clustering similarity is defined as measuring the raw frequency of a term " t " inside in a document "d". It simply depict the occurrence of a term in a document. When we give these 6 documents to the document pre-processor we get 'blue' 'flower' 'marraig' 'rose' 'colour' 'lilli' 'kerala' 'includ', 'beauti' 'Occas' 'world' 'white' 'jasmin' 'red' as index terms. Next we need to calculate the tf of each index term in 6 documents. For example in D1 the term rose is present only one time hence its count equals 1 whereas in D2 and D3 the term white is not present hence its count equals zero. Similarly for each term tf is calculated.

B. Inverse Document Frequency (idf) And Weighting

The inter-clustering dissimilarity is quantified by measuring the inverse of the frequency of a term "t" inside the documents in the collection. It is also known as "idf" factor. The idf of a term is the logarithmic ratio between the numbers of documents in the document set to that of number of documents the term appears. I.e. Idf= logN/n, Where N is the number of documents in a database and n is the number of documents where the index term "t" appear. The log value is calculated at base 2.

C. Weight Calculation

After finding the tf and idf of each term in each document weight value can be calculated. Weight value will be equal to the product of tf-idf value .So after completing the above 3 steps the weight value of each ti in each dj will be as:

Table .1 Term X Document matrix

	D1	D2	D3	D4	D5	D6
blue	0.	0.	0.	0.	0. .40824829	0.
flower	0. 4472136	0.	0.	0.	0.	0. 57735027
marraig	0.	0.	0. 57735027	0.	0.	0.
Rose	0. 4472136	0.70710678	0.	0.	0. 40824829	0.
colour	0.	0.	0.	0. 57735027	0. 40824829	0.
Lilli	0.	0.	0.	0.	0.	0. 57735027
kerala	0.	0.70710678	0.	0.	0.	0. 57735027
includ	0.	0.	0.	0.	0. 40824829	0.
beauti	0. 4472136	0.	0.	0.	0.	0.

occas	0.	0.	0. 57735027	0.	0.	0.
World	0. 4472136	0.	0.	0.	0.	0.
white	0.	0.	0.	0. 57735027	0. 40824829	0.
jasmin	0.4472136	0.	0.57735027	0. 57735027	0.	0.
red	0.	0.	0.	0.	0.40824829	0.

V. QUERY PROCESSOR AND RANKER

A. Query Processor

This is the third part of Automatic query expansion. The first and second part are used to create the index file for the documents. Now, we will use the index file to answer actual search queries. A Query Processor is something which generate the vocabulary terms corresponding to the user inputted query and perform a searching and matching function over the index file of documents obtained in the first two steps and retrieve the top most relevant documents for the inputted query.

The steps in query pre-processing are as follows:

1) Tokenize query terms.
2) Delete stop words.
3) Stem words.

The three steps are similar to the steps in the index creation for documents. After completing the first three steps the inputted query input*: lilly and rose "*changes to ['rose', 'lilli'].

B. Ranker

Ranker is one which performs the ranking operation. The ranking process can be explained as follow:
Keys - A list which stores terms in global vocabulary array of documents.
Voc - A list which stores the vocabulary terms in user inputted query.

- For each term in voc perform the following operations:
- Let ti <- Current term in user vocabulary.
- If ti is present in keys:

  Then its position in list "keys" is retrieved.
  Row corresponding to the position value is sorted.
  If cell value is > threshold value:
  File corresponding to that cell value is retrieved and displayed to the user.

After completing this step a set of files will be retrieved corresponding to the query Q.
So considering the query *Q: lilly and rose*
The retrieved files are "D1, D2, D5, D6".Since we are considering only a single word match the retrieved documents may or may not irrelevant. So to improve the relevancy we are using AQE system.

VI. AUTOMATIC QUERY EXPANSION AND RANKER

A. Automatic Query Expansion

Automatic query expansion is the process of adding additional key terms automatically to the original query. Automatic query expansion can be done in two ways:-
1) Automatic local analysis.
2) Automatic global analysis.
In our project we choose association cluster method which is a sub-type of local clustering for automatic query expansion. In local clustering the retrieved local documents are considered for expanding the query.

B. Automatic Query Expansion Working

The working of an automatic query expansion is explained using the following example:
Consider six documents D1, D2, D3, D4, D5, and D6
D1) rose and Jasmin are beautiful flowers in the world.
D2) rose is seen in Kerala.
D3) jasmines are usually used in marriage occasions.
D4) Jasmin is white in colour.
D5) rose is seen in several colours including white red and blue.
D6) I want a lily flower in Kerala.
Initially the search engine retrieves a set of documents corresponding to the initial query and these documents are considered as the local document set.

Automatic query expansion takes these local document set as the input and construct a "term x document" matrix. For the query "Lilly and Rose" initially retrieved documents include: "D1, D2, D5, D6" All documents which contain any of the vocabulary terms i.e. either Lilly or rose are retrieved by the search engine.

These documents D1, D2, D5, and D6 are considered as the local document set. Corresponding to the local document set the "term x term" matrix are constructed.

"Term x Term" Construction Steps:

- Term x Document" Construction Method:

1) For each document in the local document set below operations are performed.
– Tokenize the documents.
– Removal of stop words.
– Perform stemming.
– Calculate tf-idf (weight vector) value of each word in the vocabulary array of local document set.
– Each word and corresponding tf-idf values are combined to form a single matrix and this matrix will be a "term X document" matrix. Each cell in the tf-idf matrix represents the weight of a particular term ti in a particular document dj.

Table 2 Term x document matrix

	D1	D2	D3	D4
BLUE	0.	0.	0.40824829	0.
FLOWER	0.4472136	0.	0.	0.57735027
ROSE	0.4472136	0.70710678	0.40824829	0.
COLOR	0.	0.	0.40824829	0.
LILLI	0.	0.	0.	0.57735027
KERALA	0.	0.	0.	0.57735027
INCLUDE	0.	0.	0.40824829	0.
BEAUTI	0.4472136	0.	0.	0.
WORLD	0.4472136	0.	0.	0.
WHITE	0.	0.	0.40824829	0.
JASMIN	0.4472136	0.	0.	0.
RED	0.	0.	0.40824829	0.

The transpose of the above matrix ("term x document") is calculated and the result will be a "Document x Term" matrix which is given below:

```
[[ 0.         0.4472136  0.4472136  0.         0.         0.         0.
   0.4472136  0.4472136  0.         0.4472136  0.       ]

 [ 0.         0.         0.70710678 0.         0.         0.70710678
   0.         0.         0.         0.         0.         0.       ]

 [ 0.40824829 0.         0.40824829 0.40824829 0.         0.
   0.40824829 0.         0.         0.40824829 0.         0.40824829]

 [ 0.         0.57735027 0.         0.         0.57735027 0.57735027
   0.         0.         0.         0.         0.         0.       ]]
```

Fig .2 Document x term

The product of "term x document" and "document x term" is calculated. It will be a "term x term" matrix this matrix shows the co-relation of a word with another word.

Next step is to expand the original user query. The expansion steps include:-

Keys1<- Vocabulary array of local document set.

Voc<- Vocabulary of user inputted query.

For each word in Voc:

• Its position in keys1 is calculated. That position value will be equal to the row value in "term x term" matrix where the term appear.

• Sort that row in decreasing order.

• Choose the first term which passes the threshold.

If that term doesn't matches with the current user vocabulary term replace the current word in user vocabulary with this high threshold word.

Otherwise choose the next word which passes the threshold value replace the current checking word in the user vocabulary with this highest threshold one.


```
[[ 0.16666667  0.          0.16666667  0.16666667  0.          0.
   0.16666667  0.          0.          0.16666667  0.          0.16666667]
 [ 0.          0.53333333  0.2         0.          0.33333333  0.33333333
   0.          0.2         0.2         0.          0.2         0.        ]
 [ 0.16666667  0.2         0.86666667  0.16666667  0.          0.5
   0.16666667  0.2         0.2         0.16666667  0.2         0.16666667]
 [ 0.16666667  0.          0.16666667  0.16666667  0.          0.
   0.16666667  0.          0.          0.16666667  0.          0.16666667]
 [ 0.          0.33333333  0.          0.          0.33333333  0.33333333
   0.          0.          0.          0.          0.          0.        ]
 [ 0.          0.33333333  0.5         0.          0.33333333  0.83333333
   0.          0.          0.          0.          0.          0.        ]
 [ 0.16666667  0.          0.16666667  0.16666667  0.          0.
   0.16666667  0.          0.          0.16666667  0.          0.16666667]
 [ 0.          0.2         0.2         0.          0.          0.          0.
   0.2         0.2         0.          0.2         0.        ]
 [ 0.          0.2         0.2         0.          0.          0.          0.
   0.2         0.2         0.          0.2         0.        ]
 [ 0.16666667  0.          0.16666667  0.16666667  0.          0.
   0.16666667  0.          0.          0.16666667  0.          0.16666667]
 [ 0.          0.2         0.2         0.          0.          0.          0.
   0.2         0.2         0.          0.2         0.        ]
 [ 0.16666667  0.          0.16666667  0.16666667  0.          0.
   0.16666667  0.          0.          0.16666667  0.          0.16666667]]
```

Fig .3 term x term

• Repeat this procedure until all the words in the user vocabulary are taken. The expanded query for the above user query "Lilly and rose" is as follows:
Expanded query: [’Kerala’]
After getting this expanded query the next step is to retrieve the relevant documents corresponding to the terms in the expanded query from the local document set.

C. Advance Searching (Ranking)

Ranker is one which performs the ranking operation. The ranking process can be explained as follows.
Keys1- A list which stores terms in global vocabulary array of local documents set.
Voc1 - A newly created vocabulary list corresponding to the user inputted query (expanded vocabulary).
For each term in voc1 perform the following operations:

- Let ti<-current term in expanded vocabulary.
- If ti is present in keys1:
  Then its position in list "keys1" is retrieved.
- Row corresponding to the position value is sorted.
- If cell value is > threshold value:

File corresponding to that cell value is retrieved and displayed to the user. After completing this step a set of files will be retrieved corresponding to the query Q. So considering the query Q: lilly and rose. The retrieved files are "D2, D6".This is the advanced search result. In initial retrieval 4 documents (D1, D2, D5 and D6) are there but this search result retrieves only two of the above results which improves the precision and relevancy of retrieval taken from the paper [1].

VII. TESTING AND EVALUATIONS

In this phase we test our system’s performance by evaluating the output that we get for sample testing queries. The retrieval performance evaluation for information retrieval system is usually based on the test reference collection and on an evaluation measure. The test reference collection consist of a collection of documents, set of example information request (sample test queries), and a set of relevant documents (provided by the specialist) for each example information request. A retrieval strategy S, the evaluation measure quantifies the similarity between the set of documents retrieved by S and the set of relevant documents provided by the specialists. And the most used retrieval evaluation measures are Recall and Precision.

A. Document Analysis

We have a document set containing 500 documents.
The average number of lines per document = 36
The average number of words per document = 373

B. Precision-Recall Analysis

Let information request (query) be I (test reference collection) and are be the corresponding relevant document set. Let |R| be the number of relevant documents in this set. And we give query to the system, system process the query and generate a document answer set A. Let |A| be the number of documents in this set. Let |Ra| be the number of documents in the intersection of R and A.

Recall is the fraction of the relevant documents (the set R) which has been retrieved i.e.

$$Recall = |Ra|/|R|$$

Precision is the fraction of the retrieved documents (the set A) which is relevant i.e.

$$Precision = |Ra|/|A|$$

For finding performance of our system we find the precision and recall of each sample query. And also the average of precision and recall is taken. Initially we select some test queries (example information request) for evaluating our IR systems performance and give them to our system. And the queries that we select for testing the system are given below:

Features of theyyam
About river bharathapuzha
Gavi
Thekkady forest
Features of vembanad kayal

And we get the corresponding expected answers from our search engine and also get improved answer set after the automatic query expansion.

Table .3 Recall and Precision for Search Engine

Query	Recall	Precision
q1:About river bharathapuzha	12%	33%
	25%	50%
	37%	33%
	50%	40%
	62%	31%
	75%	33%
	87%	36%
	100%	40%
q2:Features of vembanad kayal	16%	20%
	33%	33%
	50%	42%
	83%	55%
q3:Features of theyyam	12%	20%
	25%	33%
	37%	42%
	50%	50%
	62%	55%
	75%	60%
	87%	63%
	100%	66%
q4:Thekkady forest	20%	100%
	40%	66%
	60%	75%
	80%	66%
q5:Gavi	20%	100%
	40%	100%
	60%	100%
	80%	100%

Table .4 Recall and Precision for Automatic Query Expansion

Query	Recall	Precision
q1:About river bharathapuzha	12%	33%
	25%	50%
	37%	60%
	50%	66%
	62%	55%
	75%	54%
	87%	50%

	100%	40%
q2:Features of vembanad kayal	16% 33% 50% 83%	100% 100% 100% 66%
q3:Features of theyyam	12% 25% 37% 50% 62% 75% 87% 100%	100% 66% 60% 66% 71% 66% 70% 72%
q4:Thekkady forest	20% 40% 60% 80%	100% 100% 100% 66%
q5:Gavi	20% 40% 60% 80%	100% 100% 100% 100%

VIII. RESULT AND OBSERVATION

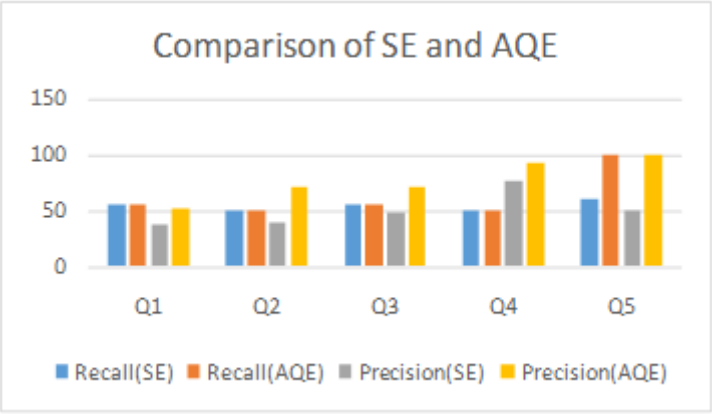
The result obtained by comparing average precision – recall of the system is tabulated in Table 5.

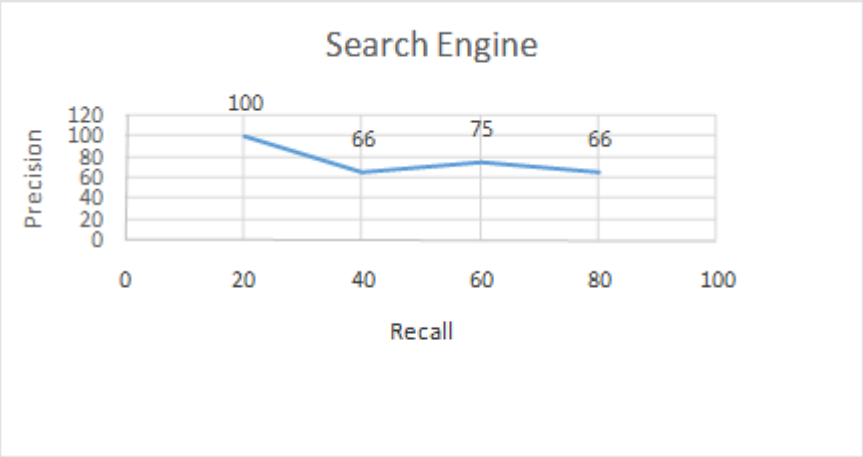
Table .5 Average Precision –Recall comparison

Query	Search Engine		Automatic Query Expansion	
	Recall	Precision	Recall	Precision
Q1	56	37	56	51
Q2	50	40	50	71
Q3	56	48	56	71
Q4	50	76	50	92
Q5	60	100	50	100

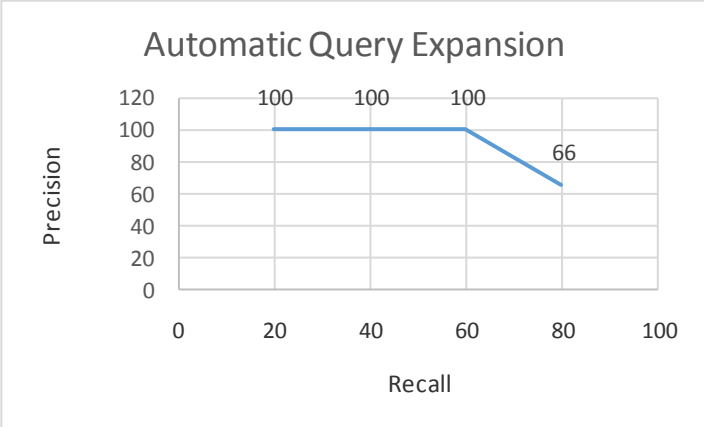
So the above table is the base for our result and observation. In majority cases that around for 98\% queries recall will remain constant for both the search engine and for the automatic query expansion. So we understood that no of the retrieval of relevant document is either increasing or decreasing .But in case of the precision the changes are remarkable. When the user give a very short query that mean a query consist of minimum no of key terms those kind of queries precision are improved drastically. And also for the query which contain the highest idf valued vocabulary those kinds of queries outputs an increase when we perform automatic query expansion. And for some large query their precision is slightly decreased when we do automatic query expansion. And from the above result we came to the conclusion that the precision is increased for more than 50% of the document and for the remaining documents more than 25% have no change and for the others the precision is decreased slightly.

So overall our system has a very good performance after the query expansion task.





Graph. 1 Precision and Recall for a particular query "Thekkady forest"



Graph. 2 Precision and Recall for a particular query "Thekkady forest"

XI. CONCLUSION

As per the goal of this project an attempt made to show how an Automatic query expansion retrieve relevant document .The present system built based on Kerala "geography". User input the query based on kerala geography AQE system retrieve relevant documents. The major problem of information retrieval is ambiguity of language. So to avoid this problem we developed a system. In the current system Precision and Recall of AQE is increased compared to search engine. The system now considers only kerala geography. We would like to expand our domain in future. Query expansion is done by using single words. In future we would like to do phrase level approach for query expansion and also reduce time, space complexity for better performance.

ACKNOWLEDGMENT

We are extremely thankful to our Principal Dr. S.P. Subramanian for giving us his consent for this project. We are thankful to Mr. P. Ganesh, Asst. Professor and Head of the Department of Computer Science, for his valuable suggestions and support. We are indebted to our project Coordinators Ms. Nisha S, Mr. Manu Madhavan and our guide Mrs. M. Aparna, Asst. Professors, Dept. of Computer Science and Engineering for their constant help and support throughout the presentation of the project by providing timely advices and guidance. We thank God almighty for all the blessing received during this endeavor. Last, but not least we thank all our friends for the support and encouragement they have given us during the course of our work.

REFERENCES

[1] Ricardo Baeza-Yates, *Introduction and Modelling, Query languages*: *IR Modern Information Retrieval Addison*: Wesley Longman Publishing Co. Inc. Boston, MA, USA 1999.
[2] (2015) The Python website. [Online].Available: https://www.python.org/
[3] (2015) The NLTK website. [Online].Available: http://www.nltk.org/

