# Language Model for Information Retrieval

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 an attempt has been made to discuss the applicability of language model as an approach to calculate the relevance of the document by utilizing user-supplied information of those documents that are relevant to the query items. This method shall have the advantage of improving retrieval performance as we have utilized user-supplied information of those documents that are relevant to the query in question. The design and implementation of information retrieval systems is concerned with methods for storing, organizing and retrieving information from a collection of documents. The quality of a system is measured by how useful it is to the typical users of the system. In this approach, a query shall be considered generated from an "ideal" document that shall satisfy the information need. The system's job has been to calculate the frequency of the word in the given document and rank them accordingly.
**Keywords:** Statistical language, Information retrieval, estimation methods, traditional approach.

## 1. INTRODUCTION
The goal of language 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. Fifty years ago Maron and Kuhns published ''On Relevance, Probabilistic Indexing and Information Retrieval" (1960). This was the first paper to present a probabilistic approach to information retrieval, and perhaps the first paper on ranked retrieval. To retrieve a ranked, or sorted, list of documents in response to the user's search request, an IR system must use evidence of similarity between the query and each document. Different types of models have been proposed for text retrieval. The most widely used include vector-space models have been proposed by G. Salton et al (1975) and probabilistic models propagated by Sparck Jones et al (2000). In the category of other prominent workers of this area we can cite the works of Bong-Hyun Cho et al (2003),Croft and Lafferty (2003), Bodoff & Robertson (2004), Nick Crasswell (2005), Jhai and Lafferty (2006), Liqi Gao et al (2006), Manning et al (2008), Victor P. Lavrenko (2010), Jun Wang and Jianhan Zhu (2010) etc.to mention only a few. A 'language model' is created for each document to represent its content; usually calculating the frequency of the words occurring in the document. Since language models became popular for use in information retrieval in the late 90s, many variant models have been proposed. However, reported evaluations of the language modeling approach for adhoc search tasks use different query sets and collections. With no formal definition, but an approximate model of relevance, most retrieval systems adopt the Probability Ranking Principle [59] for identifying relevant documents. The principle suggests that the ranking of documents be based on their probability of relevance. IR systems compute a relevance value for each document and rank documents in the decreasing order of their relevance value. Relevance value estimation is tied to the representation adopted for documents and queries. Words and/or phrases that occur in documents and queries are typically used as elements, also referred to as terms or features, in their representation. A statistical retrieval system attempts to answer the basic question that what is the probability of any word, which is relevant to the query? Now, let consider a random variable Q represent a query word and random variable I represent an word from a collection of document I, and event r denote the relevance. Then we can calculate the answer of the above mention question and calculate the probability of relevance P(r|I,Q). This probability can be estimated indirectly using the rule: $P(r|I,Q) = P(I,Q|r)P(r)/P(I,Q)$. Previously probabilistic models of document retrieval have been studied extensively. In general, these approaches can be characterized as methods of estimating the probability of relevance of documents to user queries. One component of a probabilistic retrieval model is the indexing model, i.e., a model of the assignment of indexing terms to documents.

In this paper, we can describe how to do the indexing of the document and calculate the frequency of the query items in the document on the basis of occurrence of the query items in to the document and then after calculate the probability of the relevance.

## 2. LOGIC AND ALGORITHM
For calculating the relevance of the document we will first remove the stop words from those documents in which we want to calculate the frequency of the words, which are present in the query, and produces a file in which words are reduced to their grammatical roots (called stems). Such as compute, computer, computing, computation all derived from word comput. This module is important so that it supports the partial matching concept.

### 2.1 STEMMING
Stemming is a technique for improving IR performance. It is provide searcher with ways of finding morphological variants of search terms. We use the term conflation or stemming, meaning the act of fusing or combining, as the general term for the process of matching morphological term variants. Conflation can be either manual as well as via programs like stemmers. Stemming is also used in IR to reduce the size of index file since a single stem typically corresponds to several full terms, by storing stems instead of terms, compression factors of over 50 percent can be achieved. For this purpose our program implements Porter's

stemming algorithm, which is most widely, used algorithm for stemming.

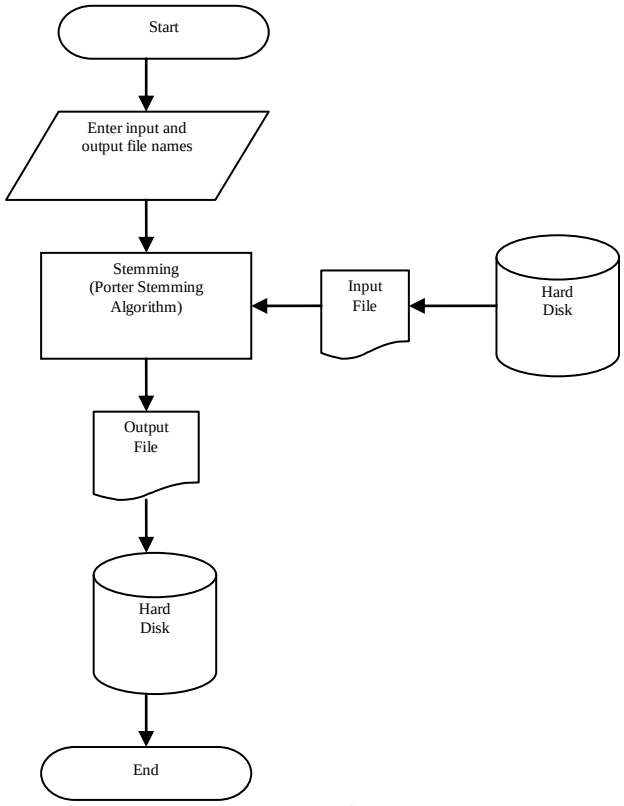


***Fig. 2.1(a) Flow chart of Stemming Module***

By using Porter algorithm we are doing stemming on the file. For this first convert all the letters in lower case. Then remove the prefixes like "kilo", "micro", "milli", "intra", "ultra", "mega", "nano", "pico", "pseudo"
For example megabyte, kilobyte all converted as "byte".
Now doing these five steps one by one

Step 1

- Remove "es" from words that end in "sses" or "ies"
  For example passes converted as pass and cries converted as converted as cri
- Remove "s" from words whose next to last letter is not an "s"
  For example runs converted as run and fuss converted as fuss
- If word has a vowel and ends with "eed" remove the "ed"
  For example agreed converted as agre
- Remove "ed" and "ing" from words that have no other vowel
  For example red converted as red, bothering converted as bother, bring converted as bring
- Add "e" is word has a vowel and ends with "ated" or "bled"
  For example enabled converted as enable, generated converted as generate
- Replace trailing "y" with an "I" if word has a vowel
  For example satisfy converted as satisfi, fly converted as fly

Step 2

- With what is left, replace any suffix on the left with suffix on the right
  For example
  conditional converted as condition
  nationalization converted as nationalize
  effectiveness converted as effective
  usefulness converted as useful
  nervousness converted as nervous
  nervously converted as nervous
  fervently converted as fervent
  inventiveness converted as inventive
  sensibility converted as sensible

Step 3

- With what is left, replace any suffix on the left with suffix on the right
  For example
  fabricate converted as fabric
  combativ converted as comb
  nationalize converted as national
  tropical converted as tropic
  faithful converted as faith
  inventiveness converted as inventive
  harness converted as har

Step 4

- Remove remaining standard suffixes
  *al, ance, ence, er, ic, able, ible, ant, ement, ment, ent, sion, tion, ou, ism, ate, iti, ous, ive, ize, ise*

Step 5

- Remove trailing "e" if word does not end in a vowel
  For example hinge converted as hing
  free converted as free

After doing these five steps on the file store all the word in to the output file.

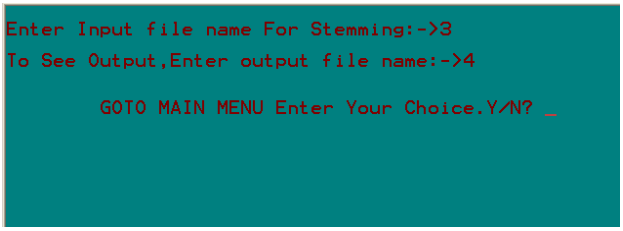


***Fig. 2.1(b) Stemming Module***

## 2.2 TOKANIZATION

Lexical analysis or tokenizing is the process of converting an input stream of characters into a stream of words or tokens. Tokens are groups of characters with collective significance. Lexical analysis is the first stage of automatic indexing, and of query processing. Lexical analysis of a query produces tokens that are parsed and turned into an internal representation suitable for comparison with indexes. In this module performs lexical analysis on the text and produces a file which contains unique words that are used for automatic Indexing and query processing which in turn can be matched with the index file to return the desired result. This is important module since all the documents need to be indexed to be retrieved. This module tokenizes documents so that they can be used for producing index terms.
Algorithm:

1. Enter the file Name getting from performing the stemming operation.
2. Enter Output File Name where you want to store the tokens.
3. Read the input file
4. If not found ()
   Cannot Open File
   Otherwise

Open the output file in write mode
Now check if null character is found
Store the first token into the outputfile
Doing this process till end of the input file and store every token in the output file.

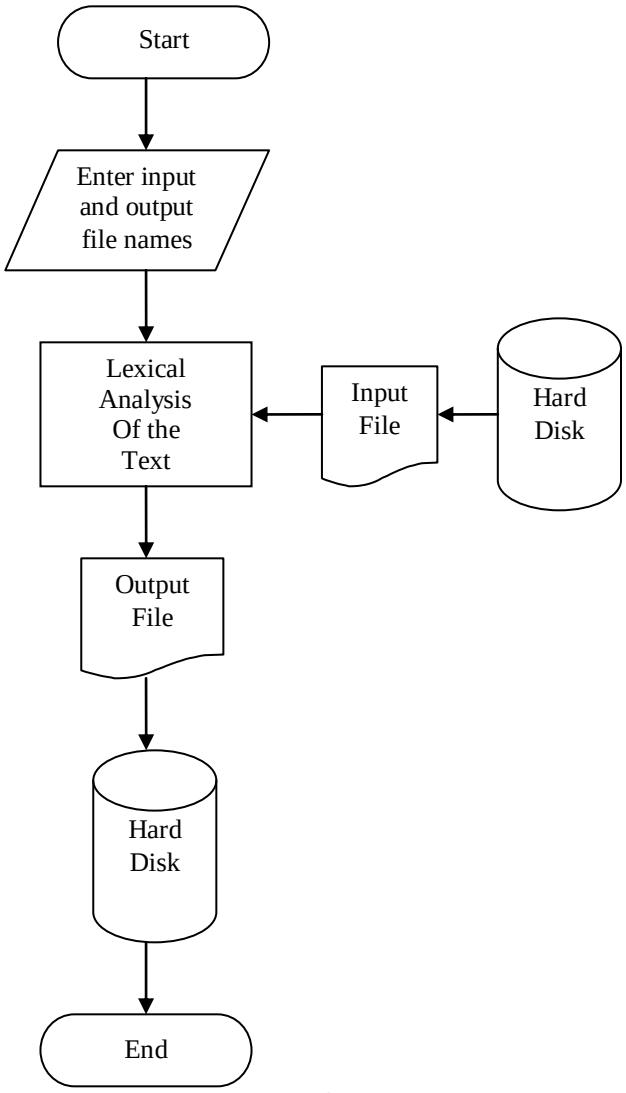


***Fig. 2.2(a) Flow chart of Tokenizing Module***

Program:

```
void WORDTOK()
{
   char*infile,*outfile,*buf,c;
   FILE*fread,*fwrite;
   printf("\nEnter Input FIle Name:->");
   scanf("%s",infile);
   printf("\nEnter Output File Name:->");
   scanf("%s",outfile);
   fread=fopen(strcat(infile,".txt"),"r");
   if(fread==NULL)
   {
      printf("\nCannot Open FIle.");
   }
   else
   {
       fwrite=fopen(strcat(outfile,".txt"),"w");
      if(fwrite==NULL)
      {
         printf("\nCannot Open FIle.");
      }
      while((fscanf(fread,"%s",buf)!=EOF))
      {
         if(c!=' ')
         {
            buf[i]=c;
            i++;
         }//endif
         else if(c==' ')
         {
            buf[i]='\0';
            fprintf(fwrite,"%s\n",buf);
            i=0;
         }//endelse
         fprintf(fwrite,"%s\n",buf);
      }//endwhile
    }//endelse
  getch();
}
```

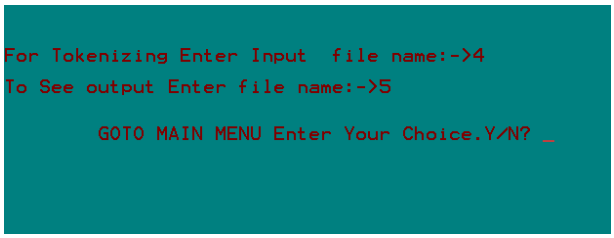


***Fig. 2.2(b) Tokenization Module***

Output: It can be generated in the form of the text file.

```
automat
care
 manual
tweak
artist
expert
thoma
marian
fine
salari
lac
float
number
baker
brush
acryl
canva
www
jwbart
sharma
automat
bush
pandel
titl
code
lexic
analyst
simul
descript
accept
```

***Fig. 2.2(c) Tokenization of the given document***

## 2.3 INDEX TERM FREQUENCY

Index terms are usually known as keywords. These are unique words that are obtained from lexical analysis process. This module finds the frequency and their relative position in the document so that they can be fetched efficiently for the user query retrieval. In this module we can take two text file one stemmed file and one tokenized file and match each word of tokenized file in the stemmed file. The result is put in the third file. This is important module since the query must retrieve efficiently the documents of user need.

1. Enter Tokenized filename.
2. Enter filename get after stemming operation.

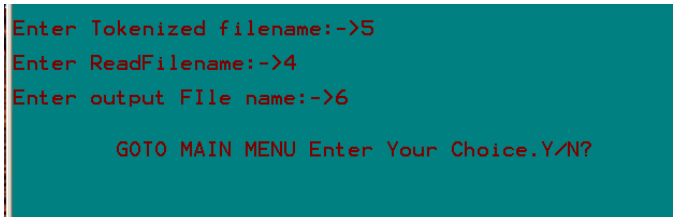


***Fig. 2.3(a) Index term frequencies and position Module***

3. Take each and every word from tokenized file and compare with the file get after stemming operation.
4. If token is found
   Then store the token name, frequency and position of the token in the output file.

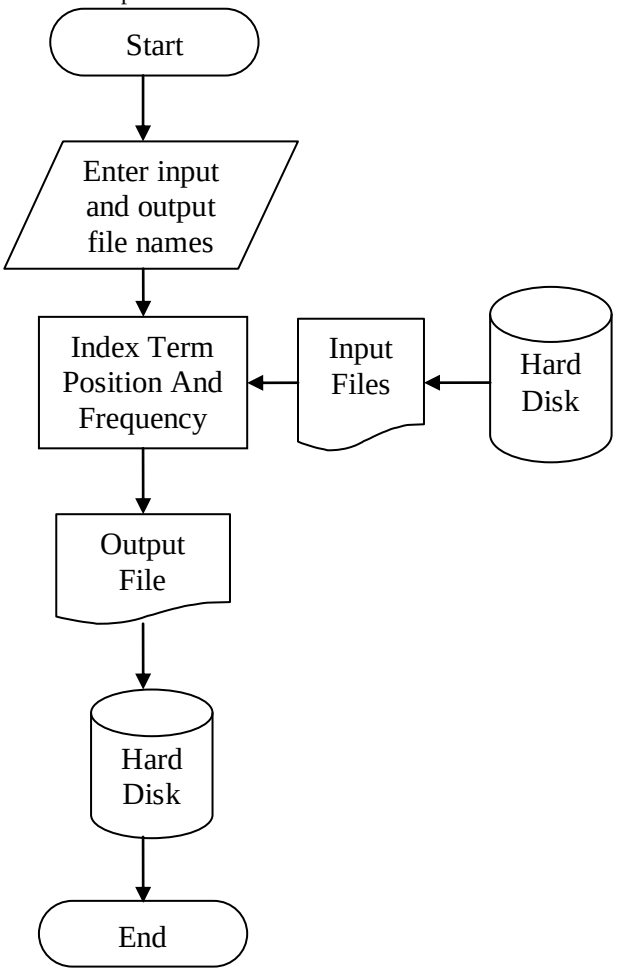


***Fig. 2.3(b) Flow chart of Index term frequencies and position Module***

Program:

```
int INDEX()
{
  char *infile,*outfile,*buf,*rbuf,*tokfile,*temp;
  FILE*read,*write,*tok;
  int count=0,pos=0,arr[MAX],tempcount,i;
  printf("\nEnter Tokenized filename:->");
  scanf("%s",tokfile);
  if((tok=fopen(strcat(tokfile,".txt"),"r"))==NULL)
  {
    printf("\ncannot open file");
  }
  printf("\nEnter ReadFilename:->");
  scanf("%s",infile);
  read=fopen(strcat(infile,".txt"),"r");
  if(read==NULL)
  {
    printf("\nCannot Open File.");
  }
  else
  {
    printf("\nEnter output FIle name:->");
    scanf("%s",outfile);
    write=fopen(strcat(outfile,".txt"),"w");
    if(write==NULL)
    {
      printf("\nCannot Open file.");
    }
    else
    {
        fprintf(write,"%s","WORD(s)\t\t
        FREQUENCY\t\t POSITION\n");
        fprintf(write,"_____\n");
        while(fscanf(tok,"%s",rbuf)!=EOF)
        {
          count=0;
          pos=0;
          tempcount=0;
          while(fscanf(read,"%s",buf)!=EOF)
          {
            tempcount++;
            if(strcmp(buf,rbuf)==0)
          {
            count++;
            arr[pos]=tempcount;
            pos++;
          }
        }
        fseek(read,0,SEEK_SET);
        fprintf(write,"%s\t\t\t%d\t\t\t",rbuf,count);
        for(i=0;i<pos;i++)
        {
          fprintf(write," %d ",arr[i]);
        }
        fprintf(write,"\n");
      }
    }
  }
  fclose(read);
  fclose(write);
  getch();
  return 0;
}
```

File Edit Format View Help

| WORD(s) | FREQUENCY | POSITION |
|---|---|---|
| automat | 2 | 1 23 |
| care | 2 | 2 174 |
| manual | 1 | 3 |
| tweak | 1 | 4 |
| artist | 1 | 5 |
| expert | 1 | 6 |
| thoma | 1 | 7 |
| marian | 1 | 8 |
| fine | 2 | 9 65 |
| salari | 1 | 10 |
| lac | 1 | 11 |
| float | 1 | 12 |
| number | 2 | 13 212 |
| baker | 1 | 14 |
| brush | 2 | 15 17 |
| air | 1 | 16 |
| brush | 2 | 15 17 |
| acryl | 1 | 18 |
| canva | 1 | 19 |
| www | 2 | 20 39 |
| jwbart | 1 | 21 |
| sharma | 1 | 22 |
| bush | 1 | 24 |
| pandei | 1 | 25 |
| titl | 1 | 26 |
| code | 7 | 27 34 42 47 50 55 77 |
| lexic | 2 | 28 36 |
| analysi | 2 | 29 37 |
| simul | 1 | 30 |
| descript | 1 | 31 |
| accept | 1 | 32 |

***Fig. 2.3(c) Index Term Frequencies and their Relative Positions***

## 3. CONCLUSIONS AND FUTURE WORK

The main objective of our system is to make the stemmed document without stop words and tokens of that final document. With the help of this system we can search the document easily. By calculating the index term frequency we can get the relevance of the document with the logic that a document in which more query items are present that document shall get a top ranking. We can also retrieve and estimate relevant (according to query) information from the document. In this paper we have provided the initial theory and empirical research for ranked retrieval on the basis of frequency of the indexed items and have presented a best way of looking at the problem of text retrieval based on probabilistic language modeling that is both conceptually simple and explanatory. We feel that our model will provide effective retrieval and provide accurate representations of the data and any one can understand our approach to retrieval.

For the future work, we are planning to add a number of extensions to our language model for information retrieval. First of all, we shall categorize the index term based on their parts of speech and calculate the frequency of the individual word on the basis of their parts of speech. We can also identify the collocation of the word pairs within different boundaries of a document such as paragraphs or sentences.

## 4. REFERENCES

[1] Bodoff, D., & Robertson, S. E. (2004). New unified probabilistic model. Journal of the American Society for Information Science and Technology, 55(6), 471–487.

[2] Bong-Hyun Cho, Changki Lee and Gary Geunbae Lee (2003) Exploring term dependences in probabilistic information retrieval model, Information processing and Management, 39(4), pp.505-519.

[3] C.D. Manning, P. Raghavan, H. Schütze (2008) Classical and web information retrieval systems: algorithms, mathematical foundations and practical issues in .Introduction to information retrieval, Cambridge.

[4] Croft, W. B., & Lafferty, J. (Eds.). (2003). Language modeling for information retrieval. Boston: Kluwer Academic.

[5] G.Salton, A. Wong, and C. S. Yang (1975), "A Vector Space Model for Automatic Indexing," Communications of the ACM, vol. 18, nr. 11, pages 613–620.

[6] Jun Wang and Jianhan Jhu (2010) On statistical analysis and optimization of Information retrieval effectiveness Metrics, In Proceedings of the 33rd International ACM SIGIR conference on Research and Development in Information Retrieval edited by Hsin-His Chen, Efthismis N.Efthimiadis, Jacques Savoy, Fabio Crestani Lugano and Stephane Marehand-Maillet, Association for Computing Machinery, New York. pp. 226-233.

[7] K. Sparck Jones, S. Walker and S.E. Robertson (2000) A probabilistic model of information retrieval: development and comparative experiments. Information Processing and Management 36, Part 1 779-808.

[8] Liqi Gao, Yu Zhang, Ting Liu & Guiping Liu (2006) Word sense Language model for Information retrieval, Lecture notes in Computer Science, Volume 4182/2006, 158-171.

[9] Maron, M. E., & Kuhns, J. (1960). On relevance, probabilistic indexing and information retrieval. Journal of the Association for Computing Machinery, 7(3), 216–244.

[10] Nick Craswell, Stephen Robertson, Hugo Zaragoza, and Michael Taylor. Relevance weighting for query independent evidence. In Proceedings of ACM SIGIR'2005, Salvador, Brazil, 2005.

[11] Victor P. Lavrenko (2010) Introduction to probabilistic models in Information retrieval. In proceedings of the 33rd International ACM SIGIR conference on Research and Development in Information Retrieval edited by Hsin-His Chen, Efthismis N.Efthimiadis, Jacques Savoy, Fabio Crestani Lugano and Stephane Marehand-Maillet, Association for Computing Machinery, New York. pp. 905.

[12] Zhai C. and Lafferty J. (2006) A risk minimization framework for information retrieval. Information processing and Management, 42(1), 31-55.