Improving Performance of Hindi-English based Cross Language Information Retrieval using Selective Documents Technique and Query Expansion

Aditi Agrawal[1], Dr. A. J. Agrawal[2]

[1, 2]Shri Ramdeobaba College of Engineering and Management, Nagpur, India

Abstract: ***This paper introduces a system for cross language information retrieval where selective term technique and query expansion will be combined to improve the retrieval of more relevant documents. The document and query processing is done using the techniques of tf-idf and cosine similarity for retrieval. Threshold value is decided to retrieve the documents relevant to query. Later a pseudo Relevance technique will be applied to further expand the query and improve relevance in future.***

Keywords: Cross Language Information Retrieval, Hindi-English Dictionary, cosine similarity, tf-idf

1. Introduction

Cross language information retrieval has become more important in recent years. The idea behind the cross-lingual IR is to retrieve documents in a language different from the query language. This may be useful even when the user is not able to understand the language used in the retrieved documents. Once the user knows that the information that is required is available and is relevant, the retrieved documents can be translated in the language known to the user. Cross-Language Information Retrieval (CLIR) has been an important field for research. Three different techniques are available in CLIR Based on different translation resources, Dictionary based CLIR, Machine translator based CLIR and Corpora based CLIR. A common approach is to translate queries using dictionaries because of the simplicity and the availability. Ambiguity is a major problem in dictionary-based CLIR systems. Given a query in the source language, the translated query in the target language is built by selecting the correct translations from a list of candidate translations for each term in the initial query. There are two techniques to address this problem. Single selection technique and multiple selections technique. Our proposal is to solve this problem by using Query Expansion (QE). QE consists of adding more words to the initial query. Thus, the query can retrieve documents that do not contain terms from the initial query. The languages involved are Hindi and English. The main structure of this article is as follows. The second section introduces vector space model, tf-idf and cosine similarity used to retrieve documents third section discusses about inverted index .The purpose of an inverted index is to allow fast full text searches, at a cost of increased processing when a document is added to the database. It is the most popular data structure used in document retrieval systems, used on a large scale for example in search engines. Fourth section describes the pseudo relevance technique combined with co-occurring terms for selecting query terms for expansion which will be implemented later and the fifth section describes the complete methodology. The last part provides a brief conclusion and further work.

2. Vector Space Model

Documents are represented as vectors by vector space model. It is used in relevancy rankings, and information retrieval. Documents and queries are represented as vectors.

$$d_j = (\omega_{1,j}, \omega_{2,j}, \dots, \omega_{t,j})$$

$$q = (\omega_{1,q}, \omega_{2,q}, \dots, \omega_{n,q})$$

The classic vector space model has the term-specific weights in the document vectors as the products of local and global parameters. It is known as term frequency-inverse document frequency model. Tf–idf weighting scheme are often used as a central tool in scoring and ranking by search engines. It is a numerical statistic that reflects the importance of the word for the document in a collection or corpus

TF (t) = (Frequency of term t in a document) / (Total number of terms in the document).

Idf: Inverse document frequency
IDF (t) = 1+log e (Total number of documents in corpus / Number of documents with term t in it).

Using the cosine similarity between document d and query q the documents which are more similar to the query can be calculated. Cosine Similarity is used to calculate the similarity between document and query or two documents. Using the formula given below we can find out the similarity between a query and documents.

Cosine Similarity (Query, Document) =Dot product (q, d)/||q||*||d||

Dot product (Query, Document) = q [0] * d [0] + q [1] * d [1] * … * q[n] * d[n]
||Query||
= square root (q [0] ^2 + q [1] ^2 + … + q[n] ^2)

||Document||
= square root (d [0] ^2 + d [1] ^2 + … + d[n] ^2)

3. Inverted Index

Nearly all retrieval engines for full-text search today are dependent on a data structure called an inverted index, which provides access to the list of documents that contain the term in the query. An inverted index has postings lists, one associated with each term that appears in the collection. Inverted index is a data structure that we build while processing the documents that we are going to provide while answering the search queries. For a query, we use the index to return the list of documents which are relevant to the query. The inverted index contains mappings from terms to the documents that those terms appear in. Each term in the vocabulary is a key in the index whose value is its postings list. List of those documents in which the term appears is its postings list. if we have the following documents the information stored by the inverted index will be as follows:

Doc1: Cross Language Information Retrieval and Ranking
Doc2: Issues in Cross Language
Doc3: Information Retrieval Issues

Then the postings list of the term ‘Cross’ would be the list [1, 2], meaning the term ‘Cross’ appears in documents with IDs 1 and 2. Similarly the postings list of the term ‘Issues’ would be [2, 3], and for the term ‘Ranking’ the postings list would be [1]. We may also want to keep extra information in the index such as the number of different documents that the term appears in or the number of occurrences of the term in the whole collection etc. The amount of data to be maintained in index will directly depend on the functionality that we want in our search engine. For a search engine indexing algorithm the inverted index data structure is a central component. The search engine implementation goal is to optimize the speed of the query. With the inverted index created, the query can be resolved by directly going to the word id in the inverted index.

4. Pseudo Relevance Technique

Relevance feedback (RF) was introduced in Rocchio's work, where a formula was introduced for forming a new query vector by minimizing its similarity to non-relevant Documents and maximizing its similarity to relevant documents in the collection. Initially, this technique was applied for vector space model and uses feedbacks given by users on the relevance of documents retrieved from the initial ranking and tries to automatically refine the query. The purpose of query expansion is to introduce new terms that are closely related to original query and restructure the query. PRF assumes the top n documents from initial retrieval as being relevant and uses these pseudo-relevant documents to refine the query for the next retrieval. PRF has been widely applied in different Information Retrieval frameworks like vector space models Due to its automatic manner and effective performance, Traditional PRF approaches recalculate query term weights based on statistics from retrieved documents and the collection such as term frequency tf, document frequency df, or term frequency-inverse document frequency tf-idf. For example, each term in retrieved documents is assigned an expansion weight w (t, Dr) as the mean of term weights in each retrieved document

$$w(t, DR) = \frac{\sum_{d \epsilon D_r} W(t,d)}{R}$$

Where R is the number of retrieved documents, w (t, Dr) is the Frequency of a term t in document d. These term weights then are used to define new query by Rocchio's formula

$$Q_{new} = \alpha .Q + \beta .\sum_{r \epsilon D_r} \frac{r}{R}$$

Here, Q and Qnew represent original and new queries, Dr is the set of pseudo-relevant documents, R is the number of retrieved documents, r is the expansion term weight vector, α and β are tuneable parameters. Pseudo relevance feedback can be applied for cross language information retrieval, in different retrieval stages of pre-translation, post-translation or the combination of both with the aim of increasing retrieval performance. The PRF strategy gives an average improvement across query topics. It works well if there are many relevant documents retrieved in the initial retrieval, but is less successful when the initial retrieval effectiveness is poor

5. Our Proposed Approach

5.1 Design of English-Hindi based cross language Information retrieval system

The following are the steps of the system. Initially the query will be taken as input from the user the query can be in English or Hindi. The query if in Hindi will be converted to English with the help of bilingual dictionary. The problem of ambiguity will be handled using multiple selection technique. The documents will be retrieved using cosine similarity. The relevant documents will be shown to user if user is not satisfied query expansion option will be their where the pseudo relevance feedback and co-occurring term technique will be combined to restructure the query. After which the final retrieval will be done.

5.2 Algorithm for Retrieval System

Cross language information retrieval system is a way in which user enters the query in different language other than the documents stored in corpus. User retrieves the result in language of his choice that is known to him. The aim is to retrieve the relevant documents in Hindi and English language for the query in either Hindi or English language. User can give query in any Language, query is processed and then the system retrieves the document in the specified language. Documents are in English as well as in Hindi; Hindi documents are converted in English using Google translator for processing. The corpus consist of documents in English and Hindi related to computer science domain

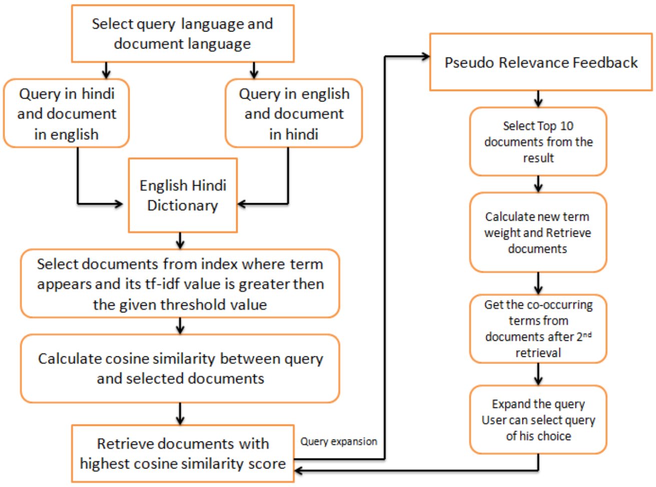
Select query language and document language
Query in hindi and document in english
Query in english and document in hindi
English Hindi Dictionary
Select documents from index where term appears and its tf-idf value is greater then the given threshold value
Calculate cosine similarity between query and selected documents
Retrieve documents with highest cosine similarity score
Query expansion
Pseudo Relevance Feedback
Select Top 10 documents from the result
Calculate new term weight and Retrieve documents
Get the co-occurring terms from documents after 2nd retrieval
Expand the query User can select query of his choice

Figure 1: Flow diagram of various steps

5.2.1 Pre-processing of Documents

The documents related to computer science domain are collected and pre-processing is done where Tokenization and stop word removal is done. The tf-idf values for every term is calculated, Documents ids in which term appears and the tf-idf value of that particular term is stored in the index in descending order. The index stores statistics about terms which make term-based search more efficient.

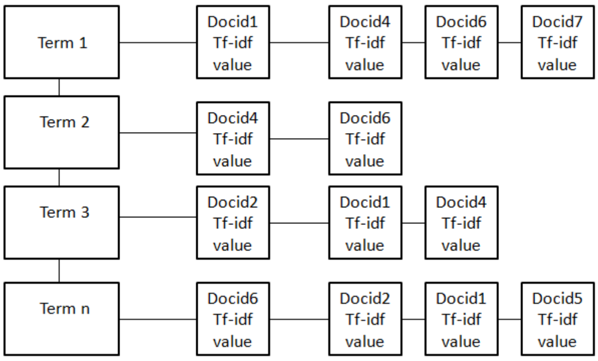

Figure 2: Structure of index

5.2.2 Conversion of Hindi words to English words:

For the conversion of Hindi words to English words, a dictionary database has been built. Hindi language consists of words which can have different meanings in English. Thus both the entries are stored in dictionary and while retrieving the documents both type of documents will be retrieved. To further improve the result query expansion option will be provided. Below is the example of how the dictionary will look

Table 1: English Hindi Dictionary

English Word	*Hindi Word*	*Hindi English Word*
computer	**lax.kd**	**sanganak**
study	**i<kbZ djuk**	**adhyayan**
structure	**lajpuk**	**sanrachana**
processing	**çlaLdj.k**	**prasanskaran**
engineering	**vfHk;kaf=dh**	**abhiyaantrikee**

5.2.3 Query Processing and retrieval of documents:

The query is processed by tokenization, stop word removal. Tf-idf values are calculated. Then cosine similarity is used to retrieve the documents. For calculating Cosine between document and query we use terms from query and those terms are searched in our inverted index. Cosine similarity will tell which documents are more similar to query. But calculating cosine for all the documents in corpus which contain the query term will be time consuming. Therefore we will provide a threshold value for selecting documents, the documents whose tf-idf value is equal to or greater than the threshold value will be considered for calculating cosine similarity. This will give us a selective number of documents for processing. Consider the following three documents in collection we will see how to calculate tf-idf and cosine similarity for the given query

Query: Computer Science applications

Document 1: Computer Science has many applications

Document 2: Computer science spans theory and practice of computer

Document 3: It was believed that computer could not be a scientific field of study

The documents are processed tokenization is done and stop words are removed

The TF IDF values for all the documents and query are shown in tables below

Table 2: Tf-Idf for Document1

Doc 1	*Computer*	*Science*	*Many*	*applications*
TF	0.2	0.2	0.2	0.2
IDF	1	1.17609126	1.4771212	1.47712125
Tf*Idf	0.2	0.23521825	0.2954242	0.29542425

Table 3: Tf-Idf for Document2

Doc 2	*Computer*	*science*	*spans*	*theory*	*Practice*
TF	0.25	0.125	0.125	0.125	0.125
IDF	1	1.1760912	1.4771212	1.477121	1.47712125
Tf*Idf	0.25	0.1470114	0.1846401	0.18464	0.18464016

Table 4: Tf-Idf for Document3

Doc 3	*believed*	*computer*	*scientific*	*filed*	*Study*
TF	0.07692	0.07692	0.07692	0.07692	0.07692
IDF	1.47712	1	1.477121	1.477121	1.47712125
Tf*Idf	0.11362	0.07692	0.113620	0.113620	0.11362017

Table 5: Tf-Idf for query

Query	*computer*	*science*	*applications*
TF	0.33333	0.33333	0.33333
IDF	1.47712125	1.47712125	1.47712125
Tf*Idf	0.49236883	0.49236883	0.49236883

Cosine Similarity (Query, Document1) =Dot Product (Query, Document1) / || Query || * || Document1 ||

Dot product (Query, Document1)
= ((0.49236883)*(0.2) + (0.49236883)*(0.23521825) + (0.49236883)*(0.29542425))
=0.35974559

||Query||
= sqrt ((0.49236883) ^2+ (0.49236883) ^2+ (0.49236883) ^2)
=0.85285887
||Document1||
= sqrt ((0.2) ^2+ (0.23521825) ^2+ (0.29542425) ^2)
=0.42732086

Cosine Similarity (Query, Document1)
=0.35974559/0.85285887*0.42732086
=0.98710695
Cosine Similarity (Query, Document2)
=0.79029023
Cosine Similarity (Query, Document3)
=0.577315832

From above computation document 1 is most similar to query then document 2 and document 3 is least similar

6. Future Work

The documents are retrieved but still there can be a problem of ambiguity. Ambiguity arises when the query entered is very short or user is not exactly sure about its requirements. This can we solved using query expansion. Combination of Pseudo Relevance feedback and co-occurring terms will be used to get the term which will be used for expansion and user can be given the option where the selection can be done. This process will give the exact query and help in retrieving more relevant documents. The query weight will be recalculated using the first set pseudo relevant documents. New weights will be used for second retrieval. The top n documents will be selected after second retrieval the co-occurring terms with the query terms will be extracted from the documents and will be added to the initial query for expansion.

7. Conclusion

We have retrieved the documents for the user query and shown the relevant documents. The tf-idf values were used for representing the documents and query in vector form and cosine similarity was computed to retrieve and rank the documents relevant to the query. The terms were stored in a inverted index were the corresponding documents in which the terms appears and their tf-idf values in descending order were also stored A threshold value was decided and documents with tf-idf greater than or equal to that value where only considered for computing cosine similarity. This helped in reducing the time complexity. Domain specific dictionary was built, the problem of ambiguity which arises when Hindi terms are converted in English is handled by considering all the different meaning of term in English for a particular Hindi term (multiple selection technique).Further we will combine the techniques of pseudo relevance feedback and co-occurring term technique where the weights of term in query will be recalculated and then will be used to retrieve new set of documents where the extracted terms can be added to the initial query for query expansion.

References

[1] Pratibha Bajpai, Parul Verma “Cross Language Information Retrieval: In Indian Language Perspective” International Journal of Research in Engineering and Technology (IJRET) Jun-2014.
[2] Benoit Gaillard, Jean-Leon Bouraoui, Emilie Guimier de Neef, Malek Boualem “Query Expansion for Cross Language Information Retrieval Improvement” 2010 IEEE
[3] Vivek Pemawat, Abhinav Saund, Anupam Agrawal “Hindi - English Based Cross Language Information Retrieval System for Allahabad Museum” 2010 International Conference on Signal and Image Processing
[4] Abdelghani Bellaachia and Ghita Amor-Tijani “Enhanced Query Expansion in English-Arabic CLIR” 19th international conference of database and expert system application.
[5] Lam Tung Giang, Vo Trung Hung and Huynh Cong Phap, “Improve Cross Language Information Retrieval with Pseudo-Relevance Feedback” International Journal of Engineering Research & Technology (IJERT), June 2015
[6] Rekha Vaidyanathan ,Sujoy Das and Namita Srivastava “Query Expansion Strategy based on Pseudo Relevance Feedback and Term Weight Scheme for Monolingual Retrieval “International Journal of Computer ,November 2014
[7] Xuwen Wang, Qiang Zhang, Xiaojie Wang and Yueping Sun ”LDA Based Pseudo Relevance Feedback For Cross Language Information Retrieval” Proceedings of IEEE CCIS2012

Author Profile

Aditi Agrawal received Bachelor of Engineering Degree in Computer Science and Engineering from Nagpur University, India in 2013 and is currently pursuing Master of Technology in Computer Science and Engineering from Shri Ramdeobaba College of Engineering and Management, Nagpur

Avinash J. Agrawal received Bachelor of Engineering Degree in Computer Technology from Nagpur University, India and Master of Technology degree in Computer Technology from National Institute of Technology, Raipur, India in 1998 and 2005 respectively. He has Ph.D. in Computer Science and Engineering from Visvesvaraya National Institute of Technology, Nagpur in 2013. His research area is Natural Language Processing and Artificial Intelligence. He is having 18 years of teaching experience. Presently he is Associate Professor in Shri Ramdeobaba College of Engineering and Management, Nagpur. He is the author of more than 50 research papers in International Journal and Conferences.