2017 2nd IEEE International Conference On Recent Trends in Electronics Information & Communication Technology (RTEICT), May 19-20, 2017, India

Semantic Desktop Search Application for Hindi-English Code-Mixed user Query with Query Sequence analysis

Amit Jena
Department of Computer Science and Engineering, IIIT Bhubaneswar,
amitcuj@gmail.com

Rakesh Chandra Balabantaray
Department of Computer Science and Engineering, IIIT Bhubaneswar,
rakeshbray@gmail.com

Abstract- With the advancement of technology, the hard disk space in our personal computers is getting bigger. Now a days it seems easy to search the web and get the relevant result than searching our personal computers. This is because of absence of any sort of interconnection between the documents, media files, and rich text present in the PC. It is really tough to create a knowledge graph for the same. In this rapidly growing digitization world, in county like India where we have listed 22 languages in the constitution the task gets even tough. People from different demographic and socio-economic background are expected to adapt to this fast evolving scenario. Under the influence of social media due deep penetration of smart phones and internet, Indian users are taking up to this. If we observe closely the users are more comfortable in typing their native language in Roman script (i.e. English.) In this work we have developed a desktop search application which can also take Hindi-English Code-Mixed user query. Subsequently the query is converted into correctly spelled English words. Then the query is analyzed for inherent sequential continuity and final reformulation is done. Code-Mixing query is frequently observed in social media and search engines especially form multilingual users. We have considered Hindi-English mixed term queries written in common roman script. Secondly, we have also included the query sequence analysis to automate the Query Reformulation process. The user may reformulate the query if he fails to get the relevant result. Here we expect the user to enter the query like text based conversational search system. On the backend, the data on the PC in indexed into the Apache Solr using Apache Tika and other dependency parsers.

Keywords - Query Reformulation, Language Identification, Indexing, Code-Mixed, Query Optimization

I. INTRODUCTION

In today's world of rapid growth in technology we are having our personal computers stacked with huge quantities of data. The data being stored may be structured like databases or unstructured like media files. Many a times we fail to find a document even though we know that we have saved it on our PC. To add to the problem, the new files types are also getting added at a rapid pace. The files being stored in our PC lacks any sort of order or rule. We are free to store any file with any given name at any place in our PC. The files may be self-created, downloaded from internet, fetched from other persons or might have come from any other possible source. The heterogeneity in source of origin of files, makes the metadata available for the files to be very different form each other. We might be having exactly same files with different file names residing in our PC. All these factors make the desktop search a very complex task. The approach need to be different from normal web search to deal with this inherent differences.

In country like India, where we have more than 22 languages, to make the technology accessible by the masses, language barrier should be removed. Though the user queries in Indian language scripts is beyond the scope of this paper. Here we present a novel approach where the user can enter search queries in Hindi-English code-mixed form but the script will be roman i.e. the user query will basically be a combination of Hindi and English terms written in English. The query then will be processed to get correctly spelled English words. And finally we will keep track of the user's query sequence to automate the query reformulation task.

At the back end, we indexed the data present on our PC using Apache Tika into Apache Solr. Once indexed into Solr, we can have a highly reliable search platform which will give us relevant documents. The Apache Tika toolkit detects and extracts metadata and text from over a thousand different file types (such as PPT, XLS, and PDF). All of these file types can be parsed through a single interface, making Tika useful for search engine indexing, content analysis, translation, and much more. Apache Solr is an open source search platform built upon a Java library called Lucene. Solr is a popular search platform for enterprise search because it can be used to index and search documents and email attachments.

II. RELATED WORKS

In our work we are having three modules: desktop search engine, handling of Code-Mixed Hindi-English search query and query sequence analysis. We will briefly explain the work done in all the three aspects stated above.

A. Desktop Search Engines

The Desktop Search applications currently available, either integrated into operating systems or as standalone applications have very limited capabilities. They search using keywords from metadata or in some cases from the headings and sub-headings of the documents. The desktop is a bit complex to search because different applications format different type of documents differently [3]. Also desktop files can be either structured like in a database or documents with embedded tags or unstructured like media files like images and audio files.

Desktop search engines use a crawler to crawl the documents in the local drive and index them. This step is similar to the web-crawler used in web search engines. The major solutions which were traditionally provided in this

978-1-5090-3704-9/17/$31.00 © 2017 IEEE

segment were by the industry leaders: Google, Microsoft and Yahoo.

The latest desktop search application by Yahoo is X1 Search. Previously it was Yahoo! Desktop Search (YDS), but now it has been discontinued. The users searching for YDS is now being redirected to X1 Search. X1 search promises to work on desktop as well as virtualization and capable to handle documents, audio, video and e-mails. It provides search results on the basis of Boolean, proximity and keyword searching. It covers more than 500 file types in their native format. But it fails to deliver results in the case of code-mixed query. In that case the query result was not at all satisfactory.

Google Desktop was an application with desktop search capabilities. It allowed text based search for documents, audio, video and emails. In September 2011, Google discontinued this search application giving the reason that the storage is now getting shifted from personal computers to cloud. This application used to show search results as top 6 most relevant fetches. Though it performed well with English queries but didn't do well when searched with Code-Mixed query.

Windows Search [5] allows us to perform instant search of our desktop. It allows to search for emails, documents and media files. Using this we can index any program's content then it will give us instant results when searched for. It also provided language packs to support local languages. But it didn't perform well with Code-Mixed query. The latest integrated Microsoft search platform is Cortana. It searches for desktop and web both.

The above applications include no metadata in their searching process, but just regular text-based index. All the above three applications have inbuilt integration with internet. When searched for a user query they also search the web for relevant results.

In the work published by L3S Research Center, University of Hanover they used the activity based metadata of the user. They showed that by exploiting the Web cache content we can achieve a better result. The main characteristic of their work was event triggered metadata generation. They indexed the data on the fly as and when there is any addition of files.

In another work by NTU, Singapore they proposed a visualization and evaluation technique for desktop search. There work did a comparative study using visualization techniques for various desktop search applications. They used various 2D and 3D graphical visualization techniques to show the query result tree view, map view, tile view and any more views to better understand the results.

B. Code-Mixed Query terms

Liu and Solorio (2008) used existing taggers for both English and Spanish language over code-switched domain for POS tagging of English- Spanish log. They achieved an accuracy of 93.48%. But the data used was manually transcribed hence failed to address the inherent problems of code-mixed query. Vyas et al. (2014) formally studied the problem, reported challenges in processing Hindi-English code-mixed query. He performed initial experiments on POS tagging. Their study showed that identification of code-mixed term's language and normalization are critical for the POS tagging task.

C. Query Reformulation Strategies

The manual query expansion technique works in close association with the user. The user provides reinforced information for the search query terms or phrases. The user may suggest some additional terms to update the query. Some wen based search engines like Google and Bing uses auto suggestion in response to the entered query. The user gets a list of closely related queries for the entered query. The most widely used query expansion technique uses global analysis of query using thesaurus. For all the query terms entered for the query, the query should be reformulated by substituting the terms with their synonyms to match all similar results. This can be achieved by using a thesaurus. This idea when combined with the term weighting factor, can boost the search result significantly.

Query reformulation techniques using the query logs is widely accepted technique to understand user intention behind the query and improving the result retrieval effectiveness. In the paper the author suggested that the readily available anchor text [11] can be an effective alternative to the query log technique. They studied various existing query refinement and reformulation techniques based on stemming, query log based substitution, and query expansion using available TREC collections. The methods demonstrated by highlights that for standard collections, log based query reformulation techniques are more efficient. But research suggest query expansion to be much more effective form of query reformulation than mere word substitution. Their results showed that for the above discussed techniques using anchor text as substitute for query log does not give any significant improvement.

Query reformulation using concept based matching [13] uses the main idea was to do concept based term matching using a list of words. In their work, they demonstrated that depending upon surface level term similarity results into higher number of false positives. There are cases which have queries with similar topics but varied intent are falsely identified as being reformulated. The researchers proposed a changed representation of web based search queries by identifying the semantics and user intention behind the query [11]. This model showed significant improvement in query reformulation performance by using features of query semantics.

III. PROPOSED WORK

In our work, we developed a desktop search engine which takes code-mixed Hindi-English search query from the user. The query then passes through the language identification module, where each token in the query is assigned a language label: Hindi, English or Rest. Then the query tokens are further processed to get normalized tokens in their respective script i.e. Hindi tokens are now written in Hindi script. These tokens are free from spelling and other possible syntactic errors.

Then the reformulated query is sent to the Apache Solr. In Solr we have already indexed the documents, files, media files using Apache Tika. We converted the video files to text by

first extracting the audio content out of it and then indexed it in Solr. In our application we maintained a search window of size 10. All the user queries were analyzed by comparing with previous 10 queries to find any dependency between them. If found, the query is auto-reformulated before being sent to Solr for searching.

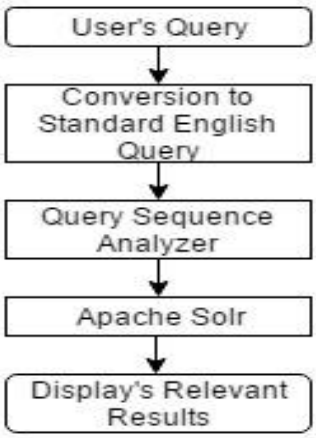

Figure 1. Overall Architecture of our System

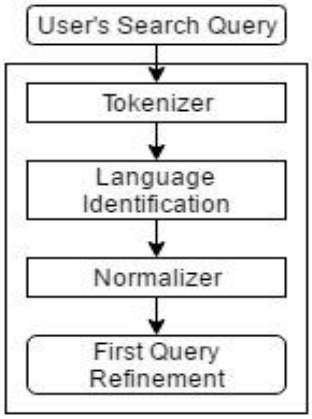

Figure 2. Conversion of Hindi-English mixed query to Standard English Query

A. Conversion of code-mixed search query to standard words

This stage has the Language Identification and Normalization layers. Our system takes the user entered query as input on which each module runs sequentially. The query may be bilingual code-mixed Hindi-English query or monolingual query, but the script is Roman. We used the NLTK package to tokenize original query into individual tokens. The Language Identification stage processes the tokens and tags then with a label from our predefined language labels. Depending upon the language label tag of the individual tokens, the Normalizer module runs the corresponding Hindi normalizer or the English normalizer.

B. Language Identification

Each token of the original query is given a tag out of the 'en', 'hi' and 'rest' to identify its language. Words that a Hindi-English bilingual speaker could identify as either Hindi or English were marked as 'hi' or 'en'. The label 'rest' was given to everything else.

The feature set comprised of 1) BNC: We took the normalized frequency of the 10, 00,000 English words in British National Corpus (BNC) [20]. 2) LEXNORM: It's a binary feature related to the presence of the word in the Han et al. (2011) dataset. 3) HINDI DICT: It's also a binary feature marking the presence of the word in a Hindi corpus of 30,823 transliterated words, released by Gupta (2012). 4) NGRAM: We have taken the n-grams of tokens up to n=3. We used CRF to perform the Language identification work by formulating the problem as a sequence labelling task.

1) Normalization

Words in Roman script that were given the tag 'hi' in the language identification stage were labelled with their standard form in the native script of Devanagari. Similarly, English words with language tag 'en' were corrected with their standard spelling. Words with language tag 'rest' were left unaltered.

After the language identification task is over, the noisy non-standard tokens, like Hindi words inconsistently written in many ways using the Roman script, in the entered query were transformed into standard words in the identified language. To address this, a normalization model which does language-specific transformations, resulting into correctly spelled words for a given token is built.

The Hindi normalizer, converted the words identified as Hindi to Devanagari script from roman script. We used GIZA++ and CRF to obtain Hindi words in Devanagari script from roman script. GIZA++ was used to obtain character alignment between non-standard Hindi words written in roman to normalized words format. Then CRF was employed for conversion from Roman script to Devanagari Script using learnt letter transformation. Then we used a standard Hindi dictionary to convert normalised Hindi word from Devanagari script to equivalent English words in roman script.

The English normalizer too worked on the same principle. Using edit distance and comparing with BNC (British National Corpus), we converted the words to Standard English words. These words were free from spelling errors and have no abbreviations.

C. Query Reformulation

First, we need to identify if there is a need of query. The reformulation strategy will take into consideration, the inherent continuity in the query sequence, the flow of concept and user intention behind the query. Multiple strategies exist to reformulate a query resulting in to different types of query reformulations:

1) Generalization: A generalization [19] reformulation occurs when the second query is intended to seek more general information compared to the first query.

2) Specification: A specification reformulation occurs when the second query is intended to seek more specific information compared to the first query.

3) Same Intent: A same intent reformulation occurs when the second query is intended to express the same intent as the first query [13].

We used the features given below to predict if query reformulation is needed:

1) Length of the queries and difference between them.

2) Count of out-of-vocabulary words in Q1, Q2 and the difference between them.

3) Count of common "exact match" concepts.

4) Count of common "approximate match" concepts.

5) Count of common "semantic match" concepts.

6) Count of concepts in query window and the difference between them.

7) Count of concepts in current query but not in previous query.

8) Count of concepts in previous query but not in current query.

9) Then do the evaluation by replacing the concepts with keywords.

We are maintaining a sliding window size of 10 for searched queries $\{Q_i, Q_{i+1}, \ldots, Q_{i+10}\}$, i.e. we will keep track of 9 previous queries while analysing the current query. At present our system is able to handle the following types of queries:

- Pronoun replacement: If the subsequent query have any pronoun then it is substituted with the noun from the previous query [17, 18].
- Name entity identification: If there is no pronoun in the subsequent query, but there is some named entity in the queries then it tries to combine the queries belonging to the same named entity.
- Continuity of concept: The query is segmented into semantic phrases depending upon the mutual information score. Whenever the point wise mutual information between two consecutive words drop below a predefined threshold, a segment break is inserted.

D. Named entity identification

For the search query we identify the named entities present in the query. With each entered query, we search for presence of named entities like company name, hotel, monuments and establish a link between them and the place (city or town), if present in the previous query. For e.g. if Q_1 is "Hotels at New Delhi" and Q_2 is "Hotel Taj", then the reformulated query becomes "Hotels Taj at New Delhi".

E. Continuity of concept

To capture concept similarity, we compute the concept similarity by measuring the semantic similarity between two queries in consideration. We used the following formula to compute similarity between two sequence of words, $Q = \{q_1, q_2, \ldots, q_I\}$ and $S = \{s_1, s_2, \ldots, s_J\}$.

$$P(S|Q) = \prod_{i=1}^{I} \sum_{j=1}^{J} P(S_i|Q_j) P(Q_j|Q) \quad (1)$$

where P(S|Q) is the unigram probability of word 'S' in the entire Query Q.

F. Integration of Apache Tika, Solr and other modules

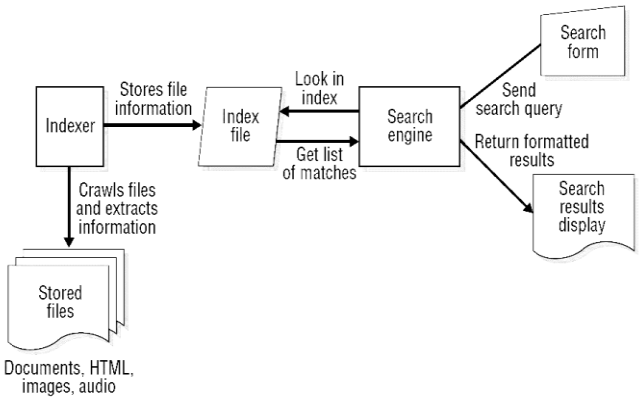

Figure 3. Using Apache Tika we indexed the documents and files into Apche Solr. The Media files like video and audio were converted to text before being indexed into Apache Solr.

IV. RESULTS

The accuracy of language identification using CRF is 88%. And the accuracy of the Hindi and English normalizer was 76.5% and 81% respectively. The normalization module yields an overall accuracy of 78% [22]. The reformulated queries were tested by putting them into our Apache Solr based local search platform. In this, we have indexed 1000 files including rich text documents like pdf, audio files and normal text documents. For precision calculation we have taken top 5 documents fetched for each query.

TABLE I. PRECISION OF ORIGINAL QUERY

Query #	Original Query	P@3	P@5
Q1	bharat ki rajdhani	0	0
Q2	capital of the country	0.5	0.35
Q3	last viceroy of india	1	1
Q4	ind ka prim minister kaun hai	0	0.2
Q5	unka kitna age	0	0
Q6	gov jobs jiska exam hota hai	0	0.2
Q7	festivals here	0	0
Q8	mickel jackson fav step	0	0
Q9	mumbai to kolkata kese jana hai	0.3	0.28
Q10	what people here eat	0	0

TABLE II. PRECISION OF REFORMULATED QUERY

Query #	Original Query	P@3	P@5
Q1	bharat's capital	0.6	0.5

Q2	capital of bharat	0.6	0.5
Q3	last viceroy of india	1	1
Q4	india's prime minister who is	1	0.95
Q5	india's prime minister how much age	1	1
Q6	india govt job whose exam have	0.65	0.7
Q7	festivals india	1	1
Q8	mickel jackson favourite step	0	0
Q9	mumbai to kolkata how to go	0.6	0.73
Q10	what people india eat	0.6	0.5

Each document (pdf/doc file) is given a relevancy score as follows:

- Document is relevant : 1
- Document is partial relevant : 0.5
- Document is not relevant : 0

The relevancy score is awarded on the basis of what the user intend to search but not on what search query he is entering alone.

V. CONCLUSION AND FUTURE WORK

Our desktop search application performed well with Code-Mixed Hindi-English query. It was also able to handle the unstructured files on our computers in an efficient way. The queries were automatically reformulated upon discovery of continuity of concept in the subsequent queries. The use of Apache Solr and Apache Tika allowed us to index various text rich file types. Using our application, users were able to search their computer system for existence of relevant files. The search was done not only on the basis of file names but using the file meta-data and their contents too. It also gave the users the ease of searching by allowing to enter search queries as Hindi-English code-mixed query. The users demonstrated higher level of satisfaction due to the increased ease of use.

We intend to make the application as a generic one. The user will be able to provide own training files to train the system. In this way, multilingual users will be able to train the system to work with other code-mixed query languages. But care should be taken that the script for the input query has to be roman.

REFERENCES

[1] Brown, P.F., Pietra, V.J.D., Pietra, S.A.D. and Mercer, R.L., 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, *19*(2), pp.263-311.

[2] Chirita, P.A., Gavriloaie, R., Ghita, S., Nejdl, W. and Paiu, R., 2005, May. Activity based metadata for semantic desktop search. In *European Semantic Web Conference* (pp. 439-454). Springer Berlin Heidelberg.

[3] Foo, S. and Hendry, D., 2007, December. Desktop search engine visualisation and evaluation. In *International Conference on Asian Digital Libraries* (pp. 372-382). Springer Berlin Heidelberg.

[4] Schumacher, K., Sintek, M. and Sauermann, L., 2008, June. Combining fact and document retrieval with spreading activation for semantic desktop search. In *European Semantic Web Conference* (pp. 569-583). Springer Berlin Heidelberg.

[5] Wang, H.F., Lee, K.F. and Yang, Q., Microsoft Corporation, 2004. *Search engine with natural language-based robust parsing for user query and relevance feedback learning*. U.S. Patent 6,766,320.

[6] Kirsch, S.T., Chang, W.I. and Miller, E.R., Infoseek Corporation, 1999. *Real-time document collection search engine with phrase indexing*. U.S. Patent 5,920,854.

[7] White, R.W., Chu, W., Hassan, A., He, X., Song, Y. and Wang, H., 2013, May. Enhancing personalized search by mining and modeling task behavior. In *Proceedings of the 22nd international conference on World Wide Web* (pp. 1411-1420). ACM.

[8] Sharma, A., Gupta, S., Motlani, R., Bansal, P., Srivastava, M., Mamidi, R. and Sharma, D.M., 2016. Shallow Parsing Pipeline for Hindi-English Code-Mixed Social Media Text. *arXiv preprint arXiv:1604.03136*.

[9] Manning, C.D., Surdeanu, M., Bauer, J., Finkel, J.R., Bethard, S. and McClosky, D., 2014, June. The stanford corenlp natural language processing toolkit. In ACL (System Demonstrations) (pp. 55-60).

[10] Huang, J. and Efthimiadis, E.N., 2009, November. Analyzing and evaluating query reformulation strategies in web search logs. In *Proceedings of the 18th ACM conference on Information and knowledge management* (pp. 77-86). ACM.

[11] Dang, V. and Croft, B.W., 2010, February. Query reformulation using anchor text. In *Proceedings of the third ACM international conference on Web search and data mining* (pp. 41-50). ACM.

[12] Cucerzan, S. and Brill, E., 2004, July. Spelling Correction as an Iterative Process that Exploits the Collective Knowledge of Web Users. In *EMNLP* (Vol. 4, pp. 293-300).

[13] Hassan, A., 2013. Identifying Web Search Query Reformulation using Concept based Matching. In *EMNLP* (Vol. 13, pp. 1000-1010).

[14] Boldi, P., Bonchi, F., Castillo, C., Donato, D., Gionis, A. and Vigna, S., 2008, October. The query-flow graph: model and applications. In *Proceedings of the 17th ACM conference on Information and knowledge management* (pp. 609-618). ACM.

[15] Gao, J., He, X., Xie, S. and Ali, A., 2012, July. Learning lexicon models from search logs for query expansion. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning* (pp. 666-676). Association for Computational Linguistics.

[16] White, R.W., Chu, W., Hassan, A., He, X., Song, Y. and Wang, H., 2013, May. Enhancing personalized search by mining and modeling task behavior. In *Proceedings of the 22nd international conference on World Wide Web* (pp. 1411-1420). ACM.

[17] Anick, P., 2003. Learning noun phrase query segmentation. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 88-95).

[18] Bergsma, S. and Wang, Q.I., 2007, June. Learning Noun Phrase Query Segmentation. In *EMNLP-CoNLL* (Vol. 7, pp. 819-826).

[19] Mitra, M., Singhal, A. and Buckley, C., 1998, August. Improving automatic query expansion. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 206-214). ACM.

[20] *The British National Corpus*, version 3 (BNC XML Edition). 2007. Distributed by Oxford University Computing Services on behalf of the BNC Consortium. URL: http://www.natcorp.ox.ac.uk/

[21] Chirita, P.A., Costache, S., Nejdl, W. and Paiu, R., 2006, June. Beagle++: Semantically enhanced searching and ranking on the desktop. In *European Semantic Web Conference* (pp. 348-362). Springer Berlin Heidelberg.

[22] Jena, Amit, Chandra Balabantaray, Rakesh, 2017, January. Query Optimization using Query Sequence for Hindi-English Code-Mixed Query. In 2nd International Conference on Sustainable Computing Techniques in Engineering, Science and Management (SCESM 2017).

[23] Mogotsi, I.C., 2010. Christopher d. manning, prabhakar raghavan, and hinrich schütze: Introduction to information retrieval.