ISSN:0975-9646

Cross Lingual Query Dependent Snippet Generation

Pinaki Bhaskar, Sivaji Bandyopadhyay

Computer Science and Engineering Department,
Jadavpur University
Kolkata – 700032, India

***Abstract*— The present paper describes the development of a cross lingual query dependent snippet generation module. It is a language independent module, so it also performs as a multilingual snippet generation module. It is a module of the Cross Lingual Information Access (CLIA) system. This module takes the query and content of each retrieved document and generates a query dependent snippet for each retrieved document. It highlights all the query words, which appear in the generated snippet. The algorithm of this module based on the sentence extraction, sentence scoring and sentence ranking. Subjective evaluation has been done to evaluate the output of this module. English snippet got the best evaluation score, i.e. 1 and overall average evaluation score of 0.83 has been achieved in the scale of 0 to 1.**

***Keywords*— Snippet Generation, Summarization, Information Extraction, Information Retrieval, Cross Lingual, Multilingual.**

I. INTRODUCTION

Snippet is the most salient information in a document or in a retrieved documents (in case of search engine) and conveying it in short space, became an active field of research in both Information Retrieval (IR) and Natural Language Processing (NLP) communities. As per Wikipedia, a snippet is defined as a small piece of something, it may in more specific contexts refer to: (i) Snippet (programming), a short reusable piece of computer source code, (ii) Sampling (music), the use of a short phrase of a recording as an element in a new piece of music and (iii) Snipets, (sic) a series of short TV interstitials produced by Kaiser Broadcasting and Field Communications in the 1970's and early 1980's. But in the case of Information Retrieval or any Search Engine, Snippet is a one or two line query-biased summary of the retrieved document.

Snippet generation shares some basic techniques with indexing as both are concerned with identification of the essence of a document. Also, high quality snippet generation requires sophisticated NLP techniques in order to deal with various Parts Of Speech (POS) taxonomy and inherent subjectivity. Multilingual or cross lingual snippet generation requires creating a snippet from a set of text or sentence in multiple languages which presents in a same document. Most of the times, the text or sentences of each language does not convey or contains same information. So, identify and extract information from sentences of each language using same system is a very challenging task in NLP.

As said in [1] Snippets are used by almost every text search engine to complement ranking scheme in order to effectively handle user searches, which are inherently ambiguous and whose relevance semantics are difficult to assess. Generally, an effective snippet should be relevant, concise and if possible, fluent. It means that the snippet should cover the most important information in the original document about the query, no matter in which language is it. But relevancy of the snippet or how to judge the relevancy of a snippet is a big debatable issue. Should snippet relevant to the query or to the document?

The consortia of CLIA formed in the year of 2006 with 10 consortia members of IIT Bombay, IIT Kgp, IIIT Hyderabad, CADC Pune, CDAC Noida, Jadavpur University, AU-KBC, AU-CEG, ISI Kolkata and Utkal University. The objective of this consortium is to develop a Cross Lingual Information Access system, which can cross search in three different languages: One IL (Indian Language), Hindi and English. So in the CLIA system if you give and query in any of the six Indian languages (Hindi, Marathi, Bengali, Punjabi, Tamil and Telugu), system will search for the documents in that specific Indian language in which query was given and in Hindi as well as in English. The CLIA system there are two cross lingual search available one is IL-Hindi and another is IL-English. Hence we had to develop a Snippet generation module which can generated snippet from documents in any of these seven languages i.e. English and six Indian languages those are mention before or from the document in mixed languages.

In this paper, a cross lingual query dependent snippet generation system has been proposed based on the sentence scoring and sentence ranking. During initial preprocessing, text fragments are filtered and identified from the document; those are later ranked using some calculated score or weight. We define our text fragments as sentence.

II. RELATED WORK

Most of the research works related to this task or field is on development of summarization system. Very little and less number of research works have been done on snippet generation. Currently, most successful summarization systems follow the extractive summarization framework. These systems first rank all the sentences in the original document set and then select the most salient sentences to compose summaries for a good coverage of the concepts. For the purpose of creating more concise and fluent

summaries, some intensive post-processing approaches are also appended on the extracted sentences. For example, redundancy removal [2] and sentence compression [3] approaches are used to make the summary more concise. Sentence re-ordering approaches [4] are used to make the summary more fluent. In most systems, these approaches are treated as independent steps. A sequential process is usually adopted in their implementation, applying the various approaches one after another.

A lot of research work has been done in the domain of both query dependent and independent summarization. MEAD [5] is a centroid based multi document summarizer, which generates summaries using cluster centroids produced by topic detection and tracking system. NeATS [6] selects important content using sentence position, term frequency, topic signature and term clustering. XDoX [7] identifies the most salient themes within the document set by passage clustering and then composes an extraction summary, which reflects these main themes.

Graph based methods have been proposed for generating query independent summaries. Websumm [8] uses a graph-connectivity model to identify salient information. Reference [9] proposed the methodology of correlated summarization for multiple news articles. In the domain of single document summarization a system for query-specific document summarization has been proposed [10] based on the concept of document graph. A document graph based query focused multi-document summarization system has been described by [11], [12] and [13].

Reference [14] presents an investigation into the utility of document summarization in the context of IR, more specifically in the application of so-called query-biased summaries: summaries customized to reflect the information need expressed in a query. Employed in the retrieved document list displayed after retrieval took place, the summaries' utility was evaluated in a task-based environment by measuring users' speed and accuracy in identifying relevant documents. This was compared to the performance achieved when users were presented with the more typical output of an IR system: a static predefined summary composed of the title and first few sentences of retrieved documents. The results from the evaluation indicate that the use of query-biased summaries significantly improves both the accuracy and speed of user relevance judgments.

Reference [15] explored the algorithms and data structures required as part of a search engine to allow efficient generation of query-biased snippets. They began by proposing and analyzing a document compression method that reduces snippet generation time by 58% over a baseline using the zlib compression library. These experiments revealed that finding documents on secondary storage dominates the total cost of generating snippets, and so caching documents in RAM is essential for a fast snippet generation process. Using simulation, they examined snippet generation performance for different size RAM caches. Finally they proposed and analyzed document reordering and compaction, revealing a scheme that increases the number of document cache hits with only a marginal affect on snippet quality. They demand that their scheme effectively doubles the number of documents that can fit in a fixed size cache.

Reference [1] presented a system, eXtract, which addressed this important yet open problem. They identified that a good XML result snippet should be a self-contained meaningful information unit of a small size that effectively summarizes this query result and differentiates it from others, according to which users can quickly assess the relevance of the query result. They have designed and implemented a novel algorithm to satisfy these requirements and verified its efficiency and effectiveness through experiments.

In the present work, the same sentence scoring and ranking approach of [12] has been followed. While the basic unit of clustering in [12] is a paragraph, sentences have been considered as the basic unit in the present work. After the clusters are developed, the summarization method is completely different. In [11] work, the minimum-spanning tree identified over the document graph is identified as the summary. But in the present work we have parsed the top ranked sentences and compressed the sentences removing the unimportant or irrelevant phrases of the sentence.

III. System Architecture

A. *CLIA System*

In this section the overview of the system framework of the current CLIA system has been described. The CLIA system has been developed on the basic architecture of Nutch [16], which use the architecture of Lucene [17]. Nutch is an open source search engine, which supports only the monolingual search in English, etc. The architecture of Nutch has been used in CLIA. Various new or modified features of CLIA system have been added or modified into Nutch architecture. The main feature of CLIA is the cross lingual search, which needs the query translation, snippet translation and language independent output generation module such as Snippet Generation and Summary Generation.

Higher-level system architecture of CLIA system has been shown in the figure 1. The major module in the output processing of the CLIA system is the Snippet Generation module, which generates and displays the snippets of all the retrieved documents.

B. *Snippet Generation Module*

The Language Independent Snippet Generation system framework has been shown in the figure 2. The system is defined in five parts like i) Key Terms Extraction, ii) Sentence Extraction, iii) Top Sentence Identification, iv) Snippet Unit Identification and finally v) Snippet Generation which were described thoroughly in the following sections.

IV. Key Term Extraction

Key Term Extraction module has three sub modules like Query Term extraction, Title Words Extraction and Meta Keywords Extraction. All these three sub modules have been described in the following sections.

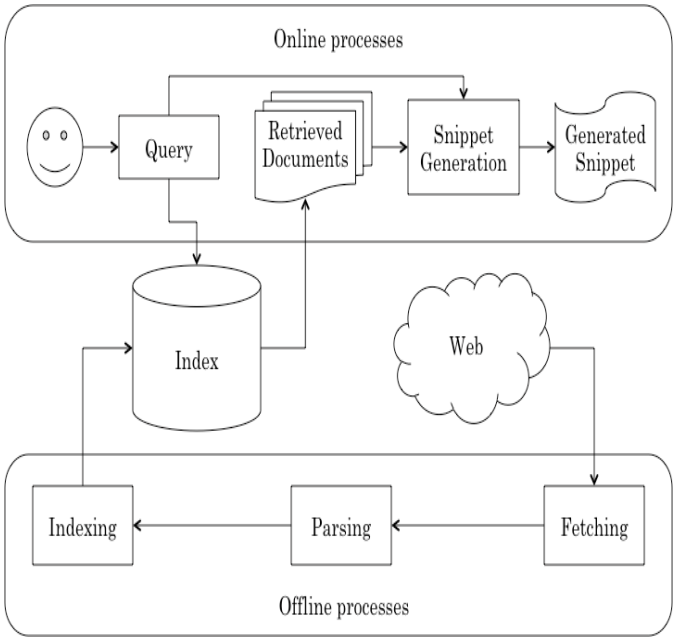

Fig. 1 Higher-level system architecture of CLIA system

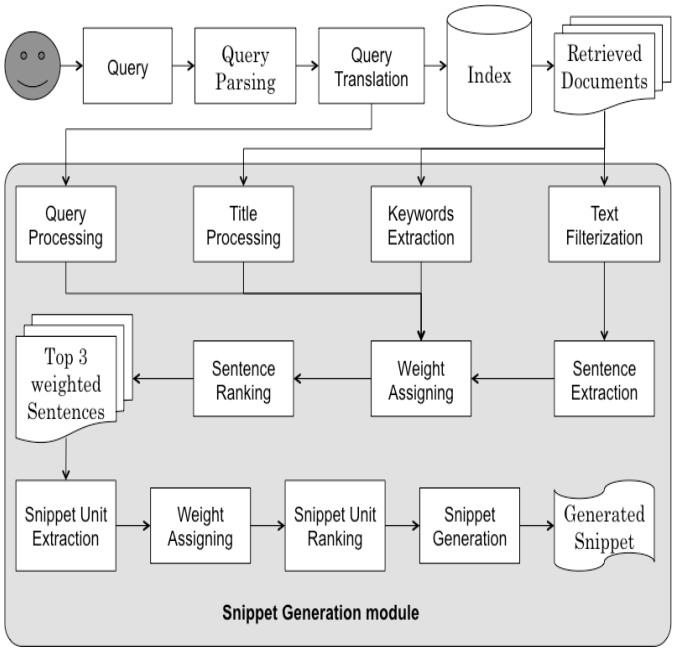

Fig. 2 System architecture of Snippet Generation module.

A. *Query Term Extraction*

First the query given by the user is parsed using the Query Parsing module. In this Query Parsing module, the Multiword Word Expressions (MWE) and Named Entities (NE) are identified and tagged in the given query using the corresponding engines. All the stop words are removed from the untagged query words. Then, if user wants the cross lingual search then the query is translated into the desired language between English or Hindi. The Query Translation module has both the translation and transliteration modules.

Query Term Extraction module gets the parsed and translated query. Now it extracts all the query terms from the query with their Boolean relations (AND or OR).

B. *Title Word Extraction*

The title of the retrieved document comes from the index to the Title Word Extraction module. After removing all the stop words from the title, all the tile words are also extracted and used as the keywords of the document in this system.

C. *Meta Keywords extraction*

If the meta keywords are available in the meta tag of the document, the meta keywords field is extracted from the document and then all the meta keywords from that field are extracted to use as more keywords of the document. As these meta keywords are written by the author of the document itself, these are the most appropriate keywords regarding the document. Meta keywords are found in most of the English documents.

V. SENTENCE EXTRACTION

The document text is parsed and the parsed text is used to generate the snippet. This module will take the parsed text of the documents as input, filter the input parsed text and extract all the sentences from the parsed text. So this module has two sub modules, Text Filterization and Sentence Extraction.

A. *Text Filterization*

The parsed text may content some junk or unrecognized character or symbol. First these kinds of character or symbols are identified and removed. The text in the query language are identified and extracted from the document using the Unicode character list of table1, which has been collected from Wikipedia [18]. The symbols like dot (.), coma (,), single quote (‘), double quote (“), ‘!’, ‘?’ etc. are common for all languages, so these are also listed as symbols in the table 1.

B. *Sentence Extraction*

In Sentence Extraction module, filtered parsed text has been parsed to identify and extract all sentences in the documents. Sentence identification and extraction is not an easy task in English document. As the sentence marker ‘.’ (dot) is not only use as a sentence marker, it has other use also like point and in abbreviation like Mr., Prof., U.S.A. etc. So it creates lot of ambiguity. A possible list of abbreviation has to create to minimize the ambiguity. Most of the times the end quotation (”) placed wrongly at the end of the sentence like .”. These kinds of ambiguities are identified and removed to extract all the sentences from the document.

VI. TOP SENTENCE IDENTIFICATION

All the extracted sentences are now searched for the keywords i.e. query terms, title words and meta keywords. Extracted sentences are given some weight according to search and ranked on the basis of the calculated weight. For this task this module has two sub modules: Weight

Assigning and Sentence Ranking, which are described bellow.

A. Weight Assigning

This sub module calculates the weights of each sentence in the document. There are basic three components in the sentence weight like query term dependent score, title word dependent score and meta keyword dependent score. These three components are calculated and added to get the final weight of a sentence.

TABLE I

UNICODE CHARACTER RANGE FOR EACH LANGUAGE

Language	Hexadecimal code	ASCII code
English	0030 – 0039 (digit), 0061 – 007A (small alphabets), 0041 – 005A (capital alphabets)	48 – 57 (digit), 65 – 90 (small alphabets), 97 – 122 (capital alphabets)
Hindi / Marathi	0901 – 097F	2305 – 2431
Bengali	0981 – 09FA	2433 – 2554
Tamil	0B82 – 0BFA	2946 – 3066
Telugu	0C01 – 0C7F	3073 – 3199
Punjabi	0A01 – 0A75	2561 – 2677
Language independent symbols	0021 – 002F, 003A – 0040, 005B – 005E, 007B – 007D	33 – 47, 58 – 64, 91 – 94, 123 - 125

1) Query Term dependent score

Query term dependent score is the most important and relevant score for snippet. Priority of this query dependent score is maximum. The query dependent score are calculated using equation 1.

$$Q_s = \sum_{q=1}^{n_q} F_q \left(20 + \left(n_q - q + 1\right) \left(\sum_p \left(1 - \frac{f_p^q - 1}{N_s} \right) \right) \times 3 \right) \quad (1)$$

where, Q_s is the query term dependent score of the sentence *s*, *q* is the no. of the query term, n_q is the total no. of query term, f_p^q is the possession of the word which was matched with the query term *q* in the sentence *s*, N_s is the total no. of words in sentence *s* and

$$F_q = \begin{matrix} 0; \textit{ if query term } q \textit{ is not found} \\ 1; \textit{ if query term } q \textit{ is found} \end{matrix} \quad .2$$

At the end of the equation 1, the calculated query term dependent score is multiplied by 3 to give the most or highest priority among all the scores.

2) Title Word dependent score

Title word are extracted from the title as described before in section #. A title word dependent score also calculated for each sentence. Generally title words are also the much relevant words of the document. So the sentence containing any title words can be a relevant sentence of the main topic of the document. Title word dependent scores are calculated using equation 3.

$$T_s = \sum_{t=0}^{n_t} F_t \left(n_t - t + 1\right) \left(\sum_p \left(1 - \frac{f_p^t - 1}{N_s} \right) \right) \times 2 \quad .3$$

where, T_s is the title word dependent score of the sentence *s*, *t* is the no. of the title word, n_t is the total no. of title word, f_p^t is the possession of the word which was matched with the title word *t* in the sentence *s*, N_s is the total no. of words in sentence *s* and

$$F_t = \begin{matrix} 0; \textit{ if title word } t \textit{ is not found} \\ 1; \textit{ if title word } t \textit{ is found} \end{matrix} \quad .4$$

At the end of the equation 3, the calculated title word dependent score is multiplied by 2 to give the second highest priority among all the scores.

3) Meta Keyword dependent score

Meta keywords are written in the document by the author manually at the time of creation of the document. As this keywords are written manually by the author itself, it should be relevant to the actual topic or concept of the document. So, this meta keyword dependent score is also very important in the weight calculation of the sentences. Equation 5 has been use to calculate the meta keyword dependent score.

$$K_s = \sum_{k=0}^{n_k} F_k \left(n_k - k + 1\right) \left(\sum_p \left(1 - \frac{f_p^k - 1}{N_s} \right) \right) \quad .5$$

where, K_s is the meta keyword dependent score of the sentence *s*, *k* is the number of the meta keyword, n_k is the total number of meta keyword, f_p^k is the possession of the word which was matched with the meta keyword *k* in the sentence *s*, N_s is the total no. of words in sentence *s* and

$$F_k = \begin{matrix} 0; \textit{ if meta keyword } k \textit{ is not found} \\ 1; \textit{ if meta keyword } k \textit{ is found} \end{matrix} \quad .6$$

After calculating all the above three scores the final weight of each sentence is calculated by simply adding all the three scores like mentioned in the equation 7.

$$W_s = Q_s + T_s + K_s \quad .7$$

where, W_s is the final weight of the sentence *s*.

In this sub module we have faced a major problem or challenge to match the query terms or title words or meta keywords with the document words. To match two words, both the words should be stemmed and converted to its root word and then the two root words should be matched. Because word can be appears in inflected form. So, here the language specific stemmer has a big role. But to make the system language independent, the language specific stemmer could not be used in this system. So, as the query comes to this module after stemming, the query words are already stemmed. Hence we were not exactly matching the query words with the document words, we just searching for those document words, which are starts with a query word. E.g. if a query word is 'India' and the document words is 'Indian', then both the words matched and

considered as the same word, as 'Indian' starts with 'India'. In this way we solve the necessity of language specific stemmer.

B. Sentence Ranking

After calculating weights of all the sentences in the document, sentences are sorted in descending order of their weight. In this process if any two or more than two sentences get equal weight, then they sorted in the ascending order of their positional value i.e. the sentence number in the document. So, this Sentence Ranking module provides the ranked sentences.

Now, top three ranked sentences are taken for the Snippet Generation. If all these three sentences are small enough to fit into the snippet without trimming themselves and overflowing the maximum length of a snippet, then after this module the system goes directly to the Snippet Generation module to generate the snippet of the document. Otherwise it goes through the Snippet Unit Selection module.

VII. SNIPPET UNIT SELECTION

A. Snippet Unit Extraction

If the total length of the top three ranked sentences of the document is larger than the maximum length of a snippet, then all these three sentences are split into snippet units. Snippet unit is basically a phrase or clause of a sentence. The snippet units are extracted in this module using the syntactic information available in the sentences. The sentences are split into snippet units according to brackets, semi colon (';'), coma (',') etc.

B. Weight Assigning

Weights of all the extracted snippet units have to be calculated to identify most relevant and most important snippet units. The same Weight assigning module is used to calculate the weights of snippet units too. So using equation 1 to equation 6, the three scores according to the query terms, tile words and meta keywords are calculated and then added these three scores to get the weight of a snippet unit.

C. Snippet Unit Ranking

After calculating weights of all the snippet units of the top three ranked sentences, they are sorted in descending order of their weight in the same way of Sentence Ranking module. In this process if any two or more than two snippet units get equal weight, then they get same rank. So, this Snippet Unit Ranking module provides the ranked list of snippet units.

VIII. SNIPPET GENERATION

This is the final and most critical module of this system. This module generates the Snippet from the sorted snippet units. As [12] using equation 8, the module selects the ranked snippet units subject to maximum length of the snippet has been reached.

$$\sum_i l_i S_i < L \qquad (8)$$

where l_i is the length (in no. of words) of snippet unit i, S_i is a binary variable representing the selection of snippet unit i for the snippet and L (=100 words) is the maximum length of the snippet.

Now, the selected snippet units are reordered according to their order of appearance in the text. If two consecutive snippet units are selected then they are concatenated without an ellipsis other wise two snippet units are concatenated with ellipsis. After contamination of the selected snippet units, all the query words in the generated snippet are tagged with the html tag to highlight them in the output. So, Html tagged generated snippet are returned for display as shown in the figure 3, 4 and 5.

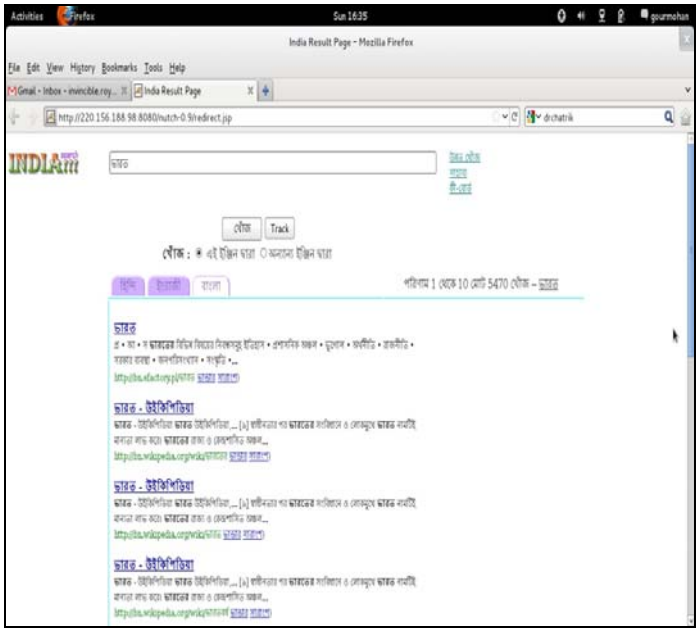

Fig. 3. Screen shot of the output page of the Bengali mono lingual search in CLIA

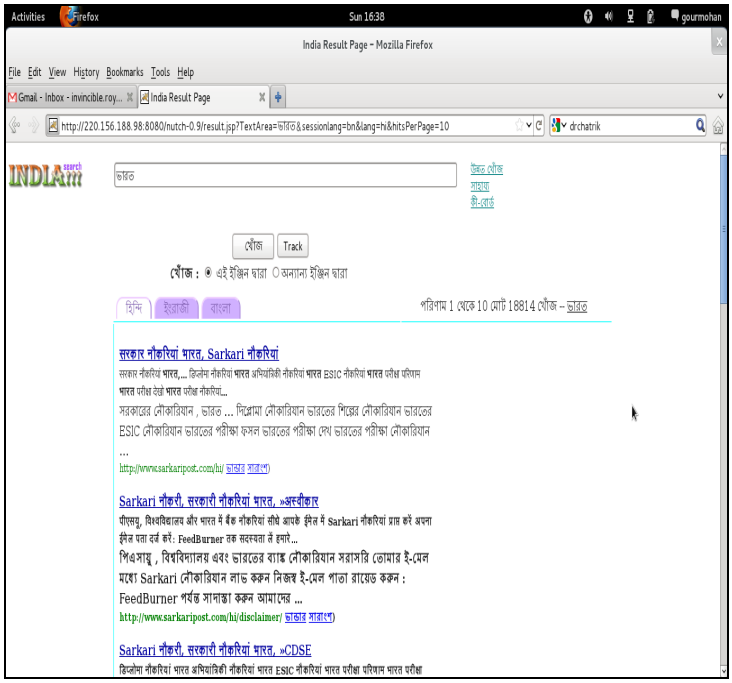

Fig. 4 Screen shot of the output page of the Bengali to Hindi cross lingual search in CLIA

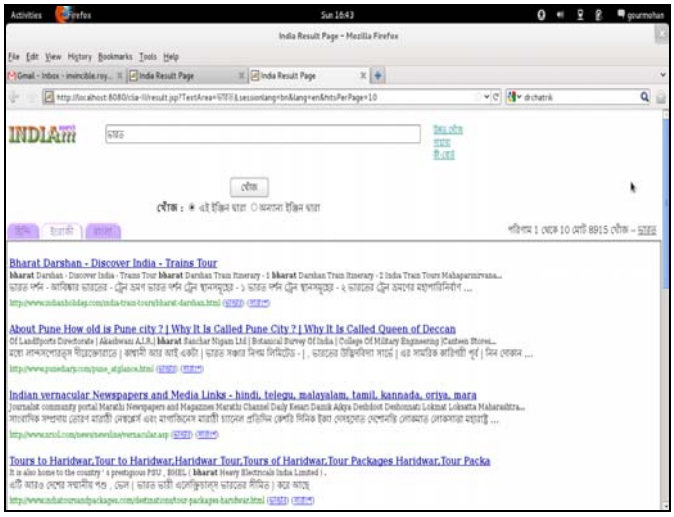

Fig. 5 Screen shot of the output page of the Bengali to English cross lingual search in CLIA

IX. CHALLENGES

During this work some difficulties or challenges were faced which are described bellow:

1. Sometimes query word appears only in the title or url, not in the body text of the web page. In this case, snippet is generated with the help of the title words and meta keywords if available. So the snippet does not contain the query words.
2. Sometimes the retrieved document is not an English page though some part of it is written in English script, so language identifier identifies it as an English page. Some English page contains some non-English characters or words written in English script. In both the cases, the generated snippet is in a language other than the document language.
3. Some page is not a Hindi / Marathi page, it is a Marathi / Hindi page. But language identifier identifies it as a Hindi / Marathi page. The generated snippet is not in the identified document language.

X. EVALUATION

As discussed before, evaluation of snippet or judgment the relevancy of a snippet is a one of the most debatable issue. So, subjective evaluation has been done to evaluate the generated snippet. Scoring parameter was set between 0 to 1, 0 for worst snippet and 1 for the best snippet. The evaluators gave a score between 0 to 1 as per how much he/she satisfied with generated snippet. Total 22 evaluators in 7 different languages have evaluated the output of this system. The evaluation scores for all the seven languages have been shown in the table 2.

TABLE II EVALUATION SCORE OF SNIPPET

Language	Evaluation score	Language	Evaluation score
English	1.00	Punjabi	0.90
Hindi	0.87	Tamil	0.81
Marathi	0.75	Telugu	0.76
Bengali	0.70	**Overall**	**0.83**

The evaluation score for English is highest and 100%. The evaluation scores are very satisfactory also for the Indian languages except for Tamil and Telugu. The CLIA system has very low performance for p@5 and p@10 i.e the ranking is very poor. So, most of the retrieved pages are not relevant to the query. As the retrieved documents are to relevant to the query the generated snippets are also not relevant to the query. Hence evaluators were not satisfied with the generated snippets especially for these two languages, Tamil and Telugu. Because they have judged the relevance of the snippet respect to the query not to the retrieved document. In the figure 6, a graph shows the snippet and ranking scores for each language.

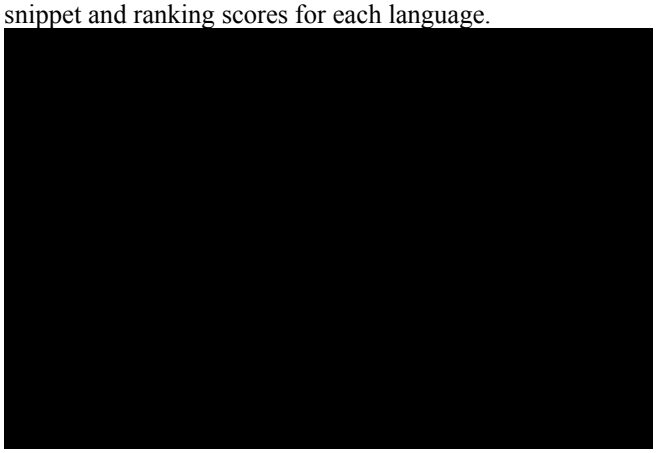

Fig. 6 Chart showing the evaluation scores of Ranking and Snippet Generation for each language

XI. CONCLUSION AND FUTURE WORKS

The Snippet Generation module is the main module of the output generation of CLIA system. But this module has many more dependency in the system, like Snippet generation is heavily dependent on the output of the parser i.e. the parse text and on the query. But in the current CLIA system the basic Html parser of Nutch are used to parse the html files. So the parse text is not cleaned enough to generate the snippet from it. All the anchor texts or menu texts are merged with actual sentences especially to the first and last sentences of the document. So, sometimes snippet contains some garbage or junk words like links or menu. If the parser extracts only the main text of the documents and cleans the parse text, then generated snippet will be more accurate, fluent.

Another problem for Snippet Generation module is query formation for cross lingual search. Now the current query translation module is not confident enough to give only one translation of the Indian language query. It translates the query words with the help of a bilingual parallel word list. So, when a query word is not found in the list then it is transliterated. So, the coverage of the parallel list is an issue. But the transliteration system has low accuracy, so that it always gives five transliterated output for each word. Hence if a query has three words and none of them are found in the parallel list, then the translated rather transliterated query will have 15 transliterated query words. To reduce the inaccuracy of the query translation module the query in the cross lingual search formed completely as OR between those 15 translated query words. This kind of query formation

technique reduces the performance of the system and retrieved less relevant documents and percussion of the system decreases.

In future we are planning to use the WordNet to match the synonyms. In the next phase of the CLIA system will incorporate the NE and MWE tags in the parse text as well as in the query. If we get the parse text with NE and MWE tags and query too, then sentence scoring for the Snippet generation will improve and more domain relevant snippet will generate as the location NEs can be identified with help of the NE and MWE tags and will give more weight to the containing sentence.

ACKNOWLEDGMENT

The work has been carried out with support from Department of Information Technology (DIT), Govt. of India funded Project Development of "Cross Lingual Information Access (CLIA)" System.

REFERENCES

[1] Yu Huang, Ziyang Liu and Yi Chen, Query Biased Snippet Generation in XML Search. In SIGMOD'08, Vancouver, BC, Canada, 2008.

[2] Carbonell, J., Goldstein, J. 1998. The Use of MMR, Diversity-based Reranking for Reordering Documents and Producing Summaries. ACM SIGIR, pp. 335--336.

[3] Knight, K., Marcu, D. 2000. Statistics-based summarization --- step one: Sentence compression. The American Association for Artificial Intelligence Conference (AAAI-2000), pp 703--710.

[4] Barzilay, R., Elhadad, N., McKeown, K. R. 2002. Inferring strategies for sentence ordering in multidocument news summarization. J. Artificial Intelligence Research. 17, 35—55

[5] Radev, D.R., Jing, H., Styś, M., Tam, D. 2004. Centroid- based summarization of multiple documents. J. Information Processing and Management. 40, 919–938

[6] Lin, C.-Y., Hovy, E.H. 2002. From Single to Multidocument Summarization: A Prototype System and its Evaluation. ACL, pp. 457-464.

[7] Hardy, H., Shimizu, N., Strzalkowski, T., Ting, L., Wise, G. B., Zhang. X. 2002. Cross-document summarization by concept classification. SIGIR, pp. 65--69.

[8] Mani, I., Bloedorn, E. 2000. Summarizing Similarities and Differences Among Related Documents. J. Information Retrieval, 1(1), 35-67

[9] Zhang, Y., Ji, X., Chu, C. H., Zha, H. 2004. Correlating Summarization of Multisource News with KWay Graph Biclustering. J. SIGKDD Explorations. 6(2), 34-42Association for Computing Machinery. 1983. *Computing Reviews*, 24(11):503-512.

[10] Varadarajan, R., Hristidis, V. 2006. A system for query specific document summarization. CIKM, pp. 622--631.

[11] Paladhi, S., Bandyopadhyay, S. 2008. A Document Graph Based Query Focused Multi-Document Summarizer. The 2nd International Workshop on Cross Lingual Information Access (CLIA), pp. 55-62

[12] P. Bhaskar and S. Bandyopadhyay, A Query Focused Multi Document Automatic Summarization, In the 24th Pacific Asia Conference on Language, Information and Computation (PACLIC 24), Tohoku University, Sendai, Japan, 2010.

[13] P. Bhaskar and S. Bandyopadhyay, A Query Focused Automatic Multi Document Summarizer, In the International Conference on Natural Language Processing (ICON), IIT, Kharagpur, India, 2010.

[14] A. Tombros and M. Sanderson. Advantages of Query Biased Summaries in Information Retrieval. In SIGIR, 1998.

[15] A. Turpin, Y. Tsegay, D. Hawking, and H. E. Williams, Fast Generation of Result Snippets in Web Search. In SIGIR, 2007

[16] The Nutch website. [Online]. Available: http://nutch.apache.org/

[17] The Lucene website. [Online]. Available: http://lucene.apache.org/

[18] (2012) List of Unicode characters on Wikipedia. [Online]. Available: http://en.wikipedia.org/wiki/List_of_Unicode_characters