Cross Lingual Query Dependent Snippet Generation for CLIA

Pinaki Bhaskar
Department of Computer Science and Engineering
Jadavpur University
Kolkata – 700032, India
pinaki.bhaskar@gmail.com

Sivaji Bandyopadhyay
Department of Computer Science and Engineering
Jadavpur University
Kolkata – 700032, India
sivaji_cse_ju@yahoo.com

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 that appear in the generated snippet. The snippet generation algorithm is 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 the overall average evaluation score of 0.71 has been achieved in the scale of 0 to 1.***

Keywords- *Snippet Generation, Cross Lingual, Multilingual, Information Retrieval.*

I. Introduction

Snippet is the most salient information in a document or in a retrieved document in case of search engine that is conveyed in a short space. In Information Retrieval or in 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 sentences in multiple languages present in a document. Most of the times, the text or sentences of each language does not convey or contain the same information. So, identification and extraction of information from sentences of each language using the same system is a very challenging task in NLP.

As said in [1] Snippets are used by almost every text search engine to complement the 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. But judging the relevancy of the snippet is a big debatable issue - should snippets be relevant to the query or to the document?

The consortia of Cross Lingual Information Access (CLIA) was formed in the year of 2006 with 10 consortia members of IIT Bombay, IIT Kharagpur, IIIT Hyderabad, CADC Pune, CDAC Noida, Jadavpur University, AU-KBC,

AU-CEG, ISI Kolkata and Utkal University. The CLIA project is funded by the Department of Information Technology, Government of India. The objective of this consortium is to develop a Cross Lingual Information Access system, which can cross search in three different languages: One Indian Language (IL), Hindi and English. So in the CLIA system if you submit a 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 the query was submitted and in Hindi as well as in English. In the CLIA system, two cross lingual search are available, one is IL-Hindi and another is IL-English. Hence we had to develop a Snippet generation module which can generate snippet from each document in any of these seven languages, i.e., English and the six Indian languages. In the phase II of the CLIA project, three more Indian languages have been added, namely, Assamese, Oriya and Gujarati. The snippet generation module is being extended to handle these Indian languages as well.

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 basic text fragment as a sentence.

II. Related Works

Most of the research works related to this task are on the development of summarization systems. Very little research work 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 applied 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. A methodology for correlated summarization for multiple news articles has been proposed in [9]. 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 in [11], [12] and [13].

An investigation into the utility of document

summarization is presented in [14] in the context of IR, more specifically in the application of so-called query-biased summaries that are customized to reflect the information need expressed in a query. The utility of the generated summaries 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 the 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.

The algorithms and the data structures that are required as part of a search engine are explored in [15] to allow efficient generation of query-biased snippets. A document compression method has been proposed that reduces the snippet generation time by 58% over a baseline using the *zlib* compression library. These experiments reveal 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 effect on snippet quality. They observed that the scheme effectively doubles the number of documents that can fit in a fixed size cache.

The snippet generation system, eXtract [1] has addressed the important problem of snippet generation. They identified that a good XML result snippet should be a self-contained meaningful information unit of a small size that effectively summarizes the 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. The minimum-spanning tree identified over the document graph is identified as the summary in [11]. 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 based on the basic architecture of Nutch1, which use the architecture of Lucene2. Nutch is an open source search engine, which supports only the monolingual search. Various new or modified features of the CLIA system have been added or modified into the 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.

1 http://nutch.apache.org/
2 http://lucene.apache.org/

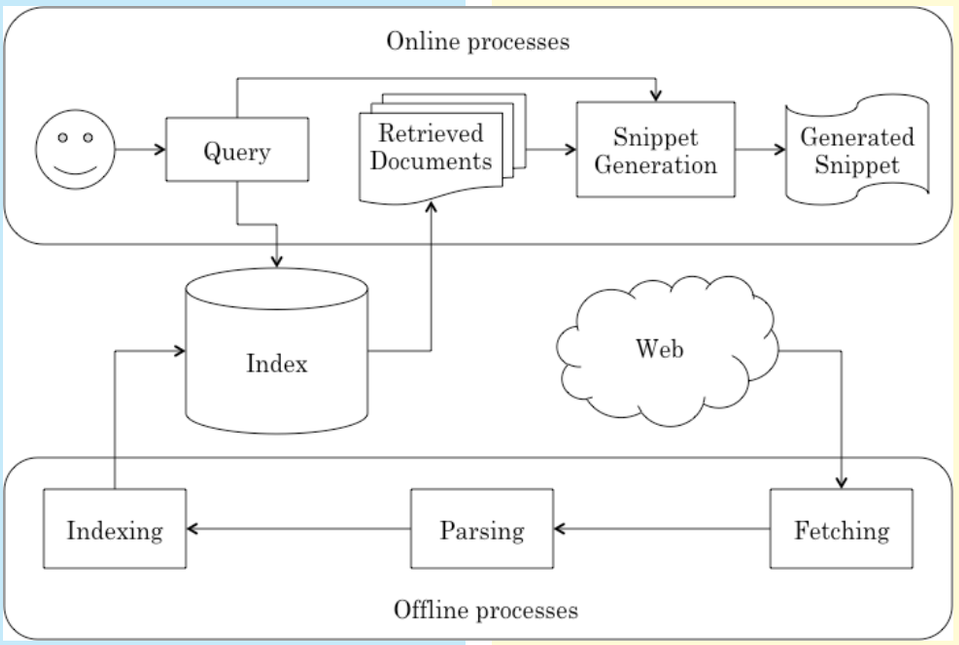

Figure 1. Higher level system architecture of CLIA system

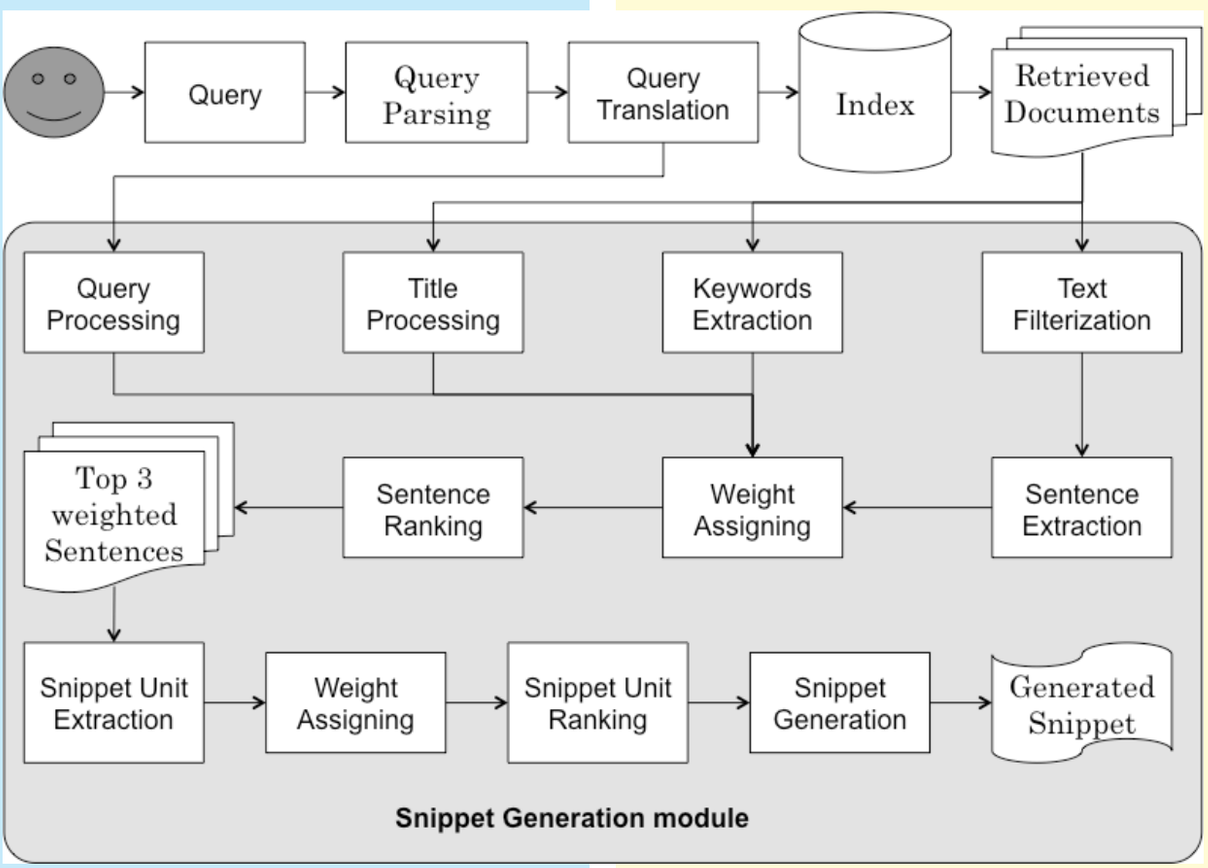

Figure 2. System architecture of Snippet Generation module.

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 Term Extraction, ii) Sentence Extraction, iii) Top Sentence Identification, iv) Snippet Unit Identification and finally v) Snippet Generation which are 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.

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 to English or Hindi. The Query Translation module includes 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 keyword field is extracted from the document. All the meta keywords are extracted for use as additional keywords of the document. As these meta keywords are written by the author of the document, these are the most appropriate keywords regarding the document. Meta keywords are found in most of the English documents.

VI. Sentence Extraction

The document text is parsed and the parsed text is used to generate the snippet. This module takes the parsed text of the documents as input, filters the input parsed text and extracts all the sentences from the parsed text. This module has two sub modules, Text Filtering and Sentence Extraction.

A. Text Filtering

The parsed text may content some junk or unrecognized characters or symbols. First these characters or symbols are identified and removed. The text in the query language are identified and extracted from the document using the Unicode character list in Table 1, which has been collected from Wikipedia3. 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.

<?> http://en.wikipedia.org/wiki/List_of_Unicode_characters

B. Sentence Extraction

In Sentence Extraction module, filtered parsed text is scanned to identify and extract all sentences in the documents. Sentence identification and extraction is not an easy task in English documents. The sentence marker '.' (dot) is used not only as a sentence marker but also as decimal point and in abbreviations like Mr., Prof., U.S.A. etc. So it creates lot of ambiguity. A possible list of abbreviations has been created. Most of the times the end quote (") is placed wrongly after the end of sentence marker. All 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 the search and ranked on the basis of the calculated weight. This module has two sub modules: Weight Assigning and Sentence Ranking, which are described below.

A. Weight Assigning

This sub module calculates the weight 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.

1) Query Term dependent score: Query term dependent score is the most important and relevant score for a snippet. This query dependent score has maximum priority. The score is calculated using Equation 1.

TABLE I. UNICODE CHARACTER RANGE FOR EACH LANGUAGE

Language	Hexadecimal code	ASCII code
English	0030 – 0039 (digit), 0061 – 007A (small alphabates), 0041 – 005A (capital alphabates)	48 – 57 (digit), 65 – 90 (small alphabates), 97 – 122 (capital alphabates)
Hindi	0901 – 097F	2305 – 2431
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

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

where, Q_s is the query term dependent score of the sentence *s*, *q* refers to each query term, n_q is the total number of terms in the query, f_p^q is the position of the word in the sentence *s* which has matched with the query term *q*, N_s is the total number of words in the 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 assign it the highest priority among all the scores.

2) Title Word dependent score: Title words are extracted from the title. A title word dependent score is calculated for each sentence. Generally title words are also relevant words for the document. So the sentence containing any of the title words can be a relevant sentence on the main topic of the document. Title word dependent scores are calculated using Equation 3.

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

where, T_s is the title word dependent score of the sentence *s*, *t* refers to each title word, n_t is the total number of terms in the title, f_p^t is the position of the word which has matched with the title word *t* in the sentence *s*, N_s is the total number of words in sentence *s* and

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

At the end of the equation 3, the calculated title word dependent score is multiplied by 2 to give the score 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 document creation time. Thus, these keywords are relevant to the actual topic or concept of the document. The meta keyword dependent score is calculated using equation 5.

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

where, K_s is the meta keyword dependent score of the sentence *s*, *k* refers to each meta keyword, n_k is the total number of meta keywords, f_p^k is the position of the word which has matched with the meta keyword *k* in the sentence *s*, N_s is the total number of words in sentence *s* and

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

.

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

$$W_s = Q_s + T_s + K_s \qquad (7)$$

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

In this sub module we have faced a major challenge to match the query terms or title words or meta keywords with the document words. As words can appear in their inflected forms, language specific stemmers are necessary before matching. Since the design goal was to develop a language independent system, we are searching for document words that maximally match the query words. Moreover, the snippet generation unit is receiving the stemmed query words. For example, if a query word is 'India' and the document word is 'Indian', then both the words match and are considered as the same word, as 'Indian' starts with 'India'. In this way we solved the necessity of language specific stemmers.

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 are 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. without chunking or full parsing.

B. Weight Assigning

Weights of all the extracted snippet units have to be calculated to identify the most relevant and most important snippet units. The same weight assigning module is used to calculate the weights of snippet units. So using equation 1 to equation 6, the three scores according to the query terms, tile words and meta keywords are calculated. These three scores are added 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 as the 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. Using the equation 8 as followed in [12], the module selects the ranked snippet units until the maximum length of the snippet has been reached.

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

where l_i is the length (in number of words) of snippet unit i, S_i is a binary variable representing the selection of snippet unit i for the snippet and L (=150 characters, say) 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 symbol (…) otherwise two snippet units are concatenated with ellipsis. After integrating all the selected snippet units in the final snippet, all the query words in the generated snippet are tagged with the html tag to highlight them in the output. So, html tagged generated snippets are returned for display as shown in the figures 3, 4 and 5.

IX. ChalLenges

During this work some challenges have been faced which are described below:

1. Sometimes query word appears only in the title or in the url but not in the body text of the web page. In this case, the 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, the 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. The language identifier may fail to correctly identify the language of a page. For example, a Hindi / Marathi page may be wrongly identified as a Marathi / Hindi page. The snippet will be generated in the language of the document and not in the identified language of the document.

X. Evaluation

As discussed before, evaluation of the generated snippet or judgment of the relevancy of a snippet is an important task. 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 based on his/her satisfaction with the generated snippet. A total of 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.

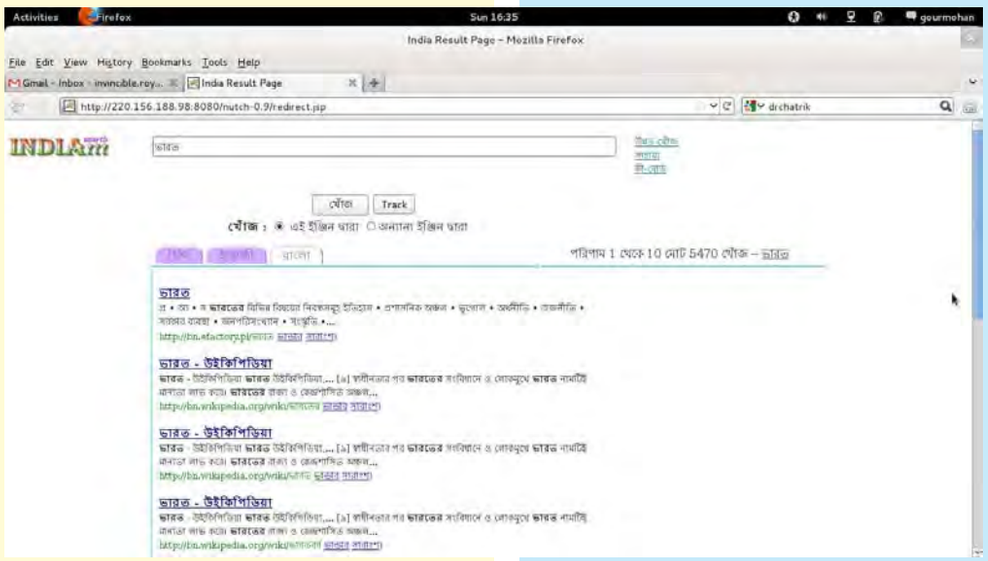

Figure 3. Sceen shot of the output page of the Bengali mono lingual seach in CLIA

Figure 4. Sceen shot of the output page of the Bengali to Hindi cross lingual seach in CLIA

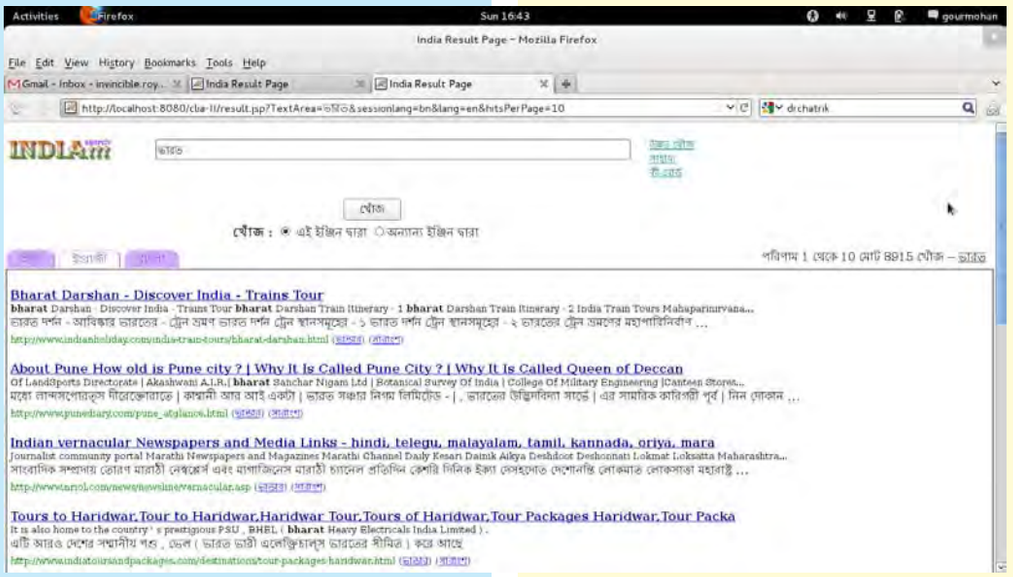

Figure 5. Sceen shot of the output page of the Bengali to English cross lingual seach in CLIA

TABLE II. EVALUATION SCORES OF SNIPPET

Language	Evaluation score
English	1.00
Hindi	0.87
Marathi	0.75
Bengali	0.70
Punjabi	0.90
Tamil	0.45
Telugu	0.30
Overall	**0.71**

The evaluation score for English is 100% and highest. The evaluation scores are very satisfactory also for the Indian languages except for Tamil and Telugu. Currently, the CLIA system is not performing satisfactorily as the ranking is very poor. Most of the retrieved pages are not relevant to the query. As the retrieved documents are not 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. In figure 6, a chart shows the snippet and ranking scores for each language.

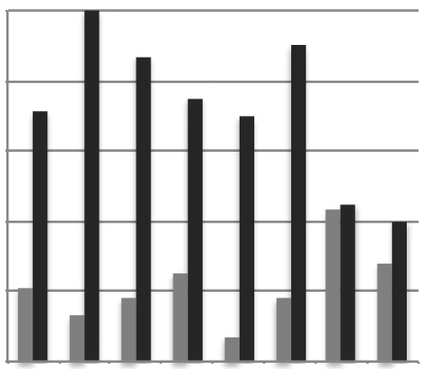

Figure 6. Chart showing the evaluation scrores of Ranking and Snippet Generation for each languages

XI. Conclusion and Future Works

The Snippet Generation module is the main module in the output generation unit of the CLIA system. The 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 in Nutch is 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 from the links or the menu items. If the parser extracts only the main text of the document and cleans the parse text, then the generated snippet will be more accurate as well as fluent.

Another problem for Snippet Generation module is the query formation for cross lingual search. The current query translation module does not provide 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. The coverage of the parallel list is an issue. But the transliteration system has low accuracy and it always gives five transliterated output for each word. Hence if a query has three words and none of these are found in the parallel list, then the ransliterated query will have 15 transliterated query words. To reduce the inaccuracy of the query translation module the query in the cross lingual search is formed by the disjunction (OR) of these 15 translated query words. This kind of query formation technique reduces the performance of the system and retrieved less relevant documents. The performance 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, the NE and MWE tags will be integrated in the parse text as well as in the query. If we get the parse text and the query with NE and MWE tags, then sentence scoring for the Snippet generation will improve and more domain relevant snippet s will be generated.

Acknowledgment

The work has been carried out as part of the Department of Information Technology (DIT), Government of India funded Consortium 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