A Hybrid QA System with Focused IR and Automatic Summarization for INEX 2011

Pinaki Bhaskar, Somnath Banerjee, Snehasis Neogi and Sivaji Bandyopadhyay

Department of Computer Science and Engineering, Jadavpur University, Kolkata, India
{pinaki.bhaskar, s.banerjee1980, snehasis.neogi}@gmail.com, sivaji_cse_ju@yahoo.com

Abstract. The article presents the experiments carried out as part of the participation in the QA track of INEX 2011. We have submitted two runs. The INEX QA task has two main sub tasks, Focused IR and Automatic Summarization. In the Focused IR system, we first preprocess the Wikipedia documents and then index them using Nutch. Stop words are removed from each query tweet and all the remaining tweet words are stemmed using Porter stemmer. The stemmed tweet words form the query for retrieving the most relevant document using the index. The automatic summarization system takes as input the query tweet along with the tweet's text and the title from the most relevant text document. Most relevant sentences are retrieved from the associated document based on the TF-IDF of the matching query tweet, tweet's text and title words. Each retrieved sentence is assigned a ranking score in the Automatic Summarization system. The answer passage includes the top ranked retrieved sentences with a limit of 500 words. The two unique runs differ in the way in which the relevant sentences are retrieved from the associated document. Our first run got the highest score of 432.2 in Relaxed metric of Readability evaluation among all the participants.

Keywords: Information Retrieval, Automatic Summarization, Question Answering, Information Extraction, INEX 2011

1 Introduction

With the explosion of information in Internet, Natural language Question Answering (QA) is recognized as a capability with great potential. Traditionally, QA has attracted many AI researchers, but most QA systems developed are toy systems or games confined to laboratories and to a very restricted domain. Several recent conferences and workshops have focused on aspects of the QA research. Starting in 1999, the Text Retrieval Conference (TREC)[1] has sponsored a question-answering track, which evaluates systems that answer factual questions by consulting the documents of the TREC corpus. A number of systems in this evaluation have successfully combined information retrieval and natural language processing

[1] http://trec.nist.gov/

techniques. More recently, Conference and Labs of Evaluation Forums (CLEF)[2] are organizing QA lab from 2010. INEX[3] has also started Question Answering track. This year, INEX 2011 designed a QA track [1] to stimulate the research for real world application. The Question Answering (QA) task performed by the participating groups of INEX 2011 is contextualizing tweets, i.e., answering questions of the form "what is this tweet about?" using a recent cleaned dump of the Wikipedia (April 2011).

Current INEX 2011 Question answering track gives QA research a new direction by fusing IR and summarization with QA. The QA track of INEX 2011 had two major sub tasks. The first task is to identify the most relevant document from the Wikipedia dump, for this we need a focused IR system. And the second task is to extract most relevant passages from the most relevant retrieved document. So we need an automatic summarization system. The general purpose of the task involves tweet analysis, passage and/or XML elements retrieval and construction of the answer, more specifically, the summarization of the tweet topic.

Automatic text summarization [2] has become an important and timely tool for assisting and interpreting text information in today's fast-growing information age. Text Summarization methods can be classified into abstractive and extractive summarization. An Abstractive Summarization ([3] and [4]) attempts to develop an understanding of the main concepts in a document and then expresses those concepts in clear natural language. Extractive Summaries [5] are formulated by extracting key text segments (sentences or passages) from the text, based on statistical analysis of individual or mixed surface level features such as word/phrase frequency, location or cue words to locate the sentences to be extracted. Our approach is based on Extractive Summarization.

In this paper, we describe a hybrid QA system of focused IR and automatic summarization for QA track of INEX 2011. The focused IR system is based on Nutch architecture and the automatic summarization system is based on TF-IDF based sentence ranking and sentence extraction techniques. The same sentence scoring and ranking approach of [6] and [7] has been followed. We have submitted two runs in the QA track (ID46RJU_CSE_run1 and ID46RJU_CSE_run2).

2 Related Works

Recent trend shows hybrid approach of QA using Information Retrieval (IR) can improve the performance of the QA system. Reference [8] removed incorrect answers of QA system using an IR engine. Reference [9] successfully used methods of IR into QA system. Reference [10] used the IR system into QA and [11] proposed an efficient hybrid QA system using IR in QA.

Reference [12] 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

[2] http://www.clef-initiative.eu//

[3] https://inex.mmci.uni-saarland.de/

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.

A lot of research work has been done in the domain of both query dependent and independent summarization. MEAD [13] is a centroid based multi document summarizer, which generates summaries using cluster centroids produced by topic detection and tracking system. NeATS [14] selects important content using sentence position, term frequency, topic signature and term clustering. XDoX [15] 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 also proposed for generating summaries. A document graph based query focused multi-document summarization system has been described by [16], [6] and [7].

In the present work, we have used the IR system as described in [10] and [11] and the automatic summarization system as discussed in [6] and [7]. In the later part of this paper, section 3 describes the corpus statistics and section 4 shows the system architecture of combined QA system of focused IR and automatic summarization for INEX 2011. Section 5 details the Focused Information Retrieval system architecture. Section 6 details the Automatic Summarization system architecture. The evaluations carried out on submitted runs are discussed in Section 7 along with the evaluation results. The conclusions are drawn in Section 8.

3 Corpus statistics

The training data is the collection of 3,217,015 documents that has been rebuilt based on recent English Wikipedia dump (April 2011). All notes and bibliographic references have been removed from Wikipedia pages to prepare plain xml corpus for an easy extraction of plain text answers. Each training document is made of a title, an abstract and sections. Each section has a sub-title. Abstract and sections are made of paragraphs and each paragraph can have entities that refer to Wikipedia pages. Therefore, the resulting corpus has this simple DTD as shown in table 1.

Test data is made up of 132 tweets (questions) from the New York Times (NYT) paper. Each tweet includes title and first sentence of NYT paper in XML format as shown in table 2. For example,

```
<topic id="2011001">
    <title>At Comic-Con, a Testing Ground for Toymakers</title>
    <txt>This summer's hottest toys won't be coming to a toy aisle near you. The
        only place to get them will be at Comic-Con International in San
        Diego.</txt>
</topic>
```

Table 1. The DTD for Wikipedia pages

```
<!ELEMENT xml (page)+>
<!ELEMENT page (ID, title, a, s*)>
<!ELEMENT ID (#PCDATA)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT a (p+)>
<!ELEMENT s (h, p+)>
<!ATTLIST s o CDATA #REQUIRED>
<!ELEMENT h (#PCDATA)>
<!ELEMENT p (#PCDATA | t)*>
<!ATTLIST p o CDATA #REQUIRED>
<!ELEMENT t (#PCDATA)>
<!ATTLIST t e CDATA #IMPLIED>
```

Table 2. XML tag format of NYT tweets of INEX 2011 corpus

```
<xml>
   <topic id="number">
      <title> tweeted NYT title </title>
      <text> first sentence of the news </text>
   </topic>
</xml>
```

4 System Architecture

In this section the overview of the system framework of the current INEX system has been shown. The current INEX system has two major sub-systems; one is the Focused IR system and the other one is the Automatic Summarization system. The Focused IR system has been developed on the basic architecture of Nutch[4], which use the architecture of Lucene[5]. Nutch is an open source search engine, which supports only the monolingual Information Retrieval in English, etc. The Higher-level system architecture of the combined QA system of Focused IR and Automatic Summarization is shown in the Figure 1.

5 Focused Information Retrieval (IR)

5.1 Wikipedia Document Parsing and Indexing

The web documents are full of noises mixed with the original content. In that case it is very difficult to identify and separate the noises from the actual content. INEX 2011

[4] http://nutch.apache.org/
[5] http://lucene.apache.org/

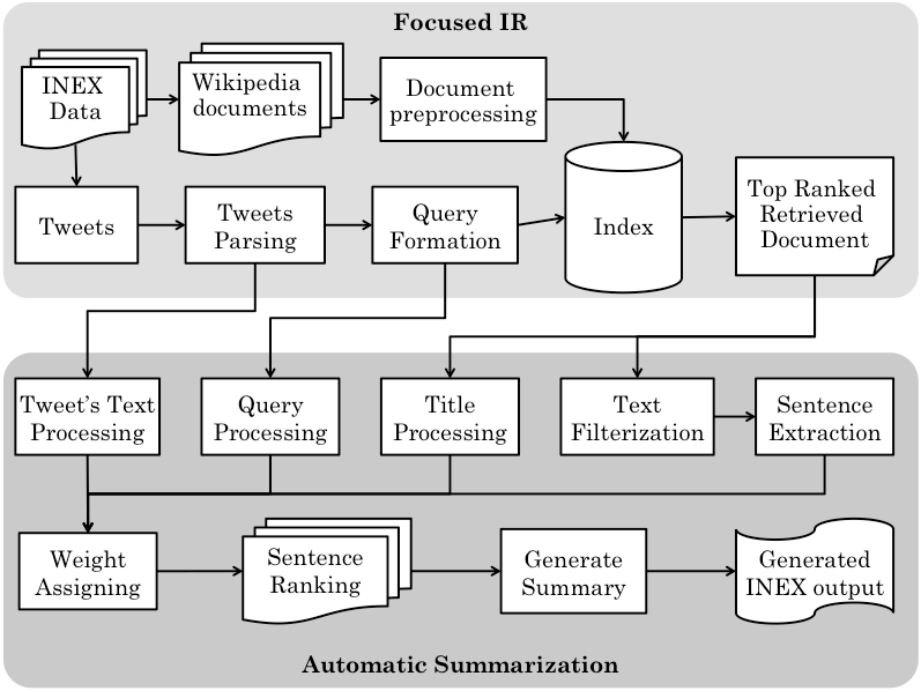

Fig. 1. Higher level system architecture of current INEX system

corpus, i.e., Wikipedia dump, had some noise in the documents and the documents are in XML tagged format. So, first of all, the documents had to be preprocessed. The document structure is checked and reformatted according to the system requirements.

XML Parser. The corpus was in XML format. All the XML test data has been parsed before indexing using our XML Parser. The XML Parser extracts the Title of the document along with the paragraphs.

Noise Removal. The corpus has some noise as well as some special symbols that are not necessary for our system. The list of noise symbols and the special symbols is initially developed manually by looking at a number of documents and then the list is used to automatically remove such symbols from the documents. Some examples are """, "&", "''''", multiple spaces etc.

Document Indexing. After parsing the Wikipedia documents, they are indexed using Lucene, an open source indexer.

5.2 Tweets Parsing

After indexing has been done, the tweets had to be processed to retrieve relevant documents. Each tweet / topic was processed to identify the query words for submission to Lucene. The tweets processing steps are described below:

Stop Word Removal. In this step the tweet words are identified from the tweets. The Stop words and question words (what, when, where, which etc.) are removed from each tweet and the words remaining in the tweets after the removal of such words are identified as the query tokens. The stop word list used in the present work can be found at http://members.unine.ch/jacques.savoy/clef/.

Stemming. Query tokens may appear in inflected forms in the tweets. For English, standard Porter Stemming algorithm[6] has been used to stem the query tokens. After stemming all the query tokens, queries are formed with the stemmed query tokens.

5.3 Document Retrieval

After searching each query into the Lucene index, a set of retrieved documents in ranked order for each query is received.

First of all, all queries were fired with AND operator. If at least one document is retrieved using the query with AND operator then the query is removed from the query list and need not be searched again. The rest of the queries are fired again with OR operator. OR searching retrieves at least one document for each query. Now, the top ranked relevant document for each query is considered for Passage selection. Document retrieval is the most crucial part of this system. We take only the top ranked relevant document assuming that it is the most relevant document for the query or the tweet from which the query had been generated.

6 Automatic Summarization

6.1 Sentence Extraction

The document text is parsed and the parsed text is used to generate the summary. 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.

[6] http://tartarus.org/~martin/PorterStemmer/java.txt

Text Filterization. The parsed text may content some junk or unrecognized character or symbol. 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, which has been collected from Wikipedia[7]. The symbols like dot (.), coma (,), single quote (‘), double quote (“), ‘!’, ‘?’ etc. are common for all languages, so these are also listed as symbols.

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 for English document. As the sentence marker ‘.’ (dot) is not only used as a sentence marker, it has other uses also like decimal point and in abbreviations like Mr., Prof., U.S.A. etc. So it creates lot of ambiguity. A possible list of abbreviation had to created to minimize the ambiguity. Most of the times the end quotation (”) is 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.

6.2 Key Term Extraction

Key Term Extraction module has three sub modules like Query Term, i.e., tweet term extraction, tweet text extraction and Title words extraction. All these three sub modules have been described in the following sections.

Query/Tweet Term Extraction. First the query generated from the tweet, is parsed using the Query Parsing module. In this Query Parsing module, the Named Entities (NE) are identified and tagged in the given query using the Stanford NER[8] engine.

Tweet’s Text extraction. Tweet’s texts are extracted and then all the keywords from the tweet text field are extracted to be used as more keywords. As these texts are provided along with the tweets, these are the most appropriate keywords regarding the tweets or topics.

Title Word Extraction. The title of the retrieved document is extracted and forwarded as input given to the Title Word Extraction module. After removing all the stop words from the title, the remaining tile words are extracted and used as the keywords in this system.

6.3 Top Sentence Identification

All the extracted sentences are now searched for the keywords, i.e., query terms, tweet’s text keywords and title words. Extracted sentences are given some weight

[7] http://en.wikipedia.org/wiki/List_of_Unicode_characters
[8] http://www-nlp.stanford.edu/ner/

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 below.

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

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

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

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 terms, 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,

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

and

$$p = \begin{matrix} 5; \textit{ if query term is NE} \\ 3; \textit{if query term is not NE} \end{matrix} \quad (3)$$

At the end of the equation 1, the calculated query term dependent score is multiplied by p to give the priority among all the scores. If the query term is NE and contained in a sentence then the weight of the matched sentence are multiplied by 5 as the value of p is 5, to give the highest priority, other wise it has been multiplied by 3 (as p=3 for non NE query terms).

Tweet's Text Keyword dependent score: Tweet's text keywords are provided along with the tweet. Hence, it should be relevant to the actual topic or concept of the tweet. So, this tweet's text keyword dependent score is also very important in the weight calculation of the sentences. Equation 4 has been use to calculate the tweet's text keyword dependent score.

$$K_s = \sum_{k=0}^{n_k} F_k (n_k - k + 1) \left(\sum_p \left(1 - \frac{f_p^k - 1}{N_s} \right) \right) \times 2 \quad (4)$$

where, K_S is the tweet's text keyword dependent score of the sentence s, k is the number of the tweet's text keyword, n_k is the total number of tweet's text keyword,

f_p^k is the possession of the word which was matched with the tweet's text keyword k in the sentence s, N_s is the total no. of words in sentence s and

$$F_k = \begin{matrix} 0; \textit{if tweet's text keyword k is not found} \\ 1; \textit{if tweet's text keyword k is found} \end{matrix} . \quad (5)$$

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.

Title Word dependent score: Title words are extracted from the title field of the top ranked retrieved document. A title word dependent score is 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 6.

$$T_s = \sum_{t=0}^{n_t} F_t\left(n_t - t + 1\right)\left(\sum_p \left(1 - \frac{f_p^t - 1}{N_s}\right)\right) \quad (6)$$

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 number of title words, f_p^t is the position of the word which matched with the title word t in the sentence s, N_s is the total number of words in sentence s and

$$F_t = \begin{matrix} 0; \textit{if title word t is not found} \\ 1; \textit{if title word t is found} \end{matrix} . \quad (7)$$

After calculating all the above three scores the final weight of each sentence is calculated by simply adding all the three scores as mentioned in the equation 8.

$$W_s = Q_s + K_s + T_s \quad (8)$$

where, W_S is the final weight of the sentence s.

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.

6.4 Summary Generation

This is the final and most critical module of this system. This module generates the Summary from the ranked sentences. As in [13] using equation 9, the module selects the ranked sentences subject to maximum length of the summary.

$$\sum_i l_i S_i < L \quad (9)$$

where l_i is the length (in no. of words) of sentence i, S_i is a binary variable representing the selection of sentence i for the summary and L (=500 words) is the maximum length of the summary.

Now, the selected sentences along with their weight are presented as the INEX output format.

7 Evaluation

7.1 Informative Content Evaluation

The organizers did the Informative Content evaluation [1] by selecting relevant passages. 50 topics were evaluated which was the pool of 14 654 sentences, 471 344 tokens, vocabulary of 59 020 words. Among them, 2801 sentences, 103889 tokens, vocabulary of 19037 words, are relevant. There are 8 topics with less than 500 relevant tokens. The evaluation measures of Information content divergences over {1,2,3,4gap}-grams (FRESA package) because it was too sensitive to smoothing on the qa-rels. So simple log difference of equation 10 was used:

$$\sum \log\left(\frac{\max\left(P(t/reference), P(t/summary)\right)}{\min\left(P(t/reference), P(t/summary)\right)}\right) \qquad (10)$$

We have submitted two runs (ID46RJU_CSE_run1, ID46RJU_CSE_run2). The evaluation scores with the baseline system scores of informativeness by organizers of all topics are shown in the table 3 and evaluation scores of informativeness based on NYT textual content are show in table 4.

Table 3. The evaluation scores of Informativeness by organizers of all topics

Run	*unigram*	*bigram*	*with 2-gap*	*Average*	*Ranking*
ID46RJU_CSE_run1	0.056092	0.0876557	0.115557	0.0876168	0.115557
ID46RJU_CSE_run2	0.056122	0.0876816	0.11558	0.087643	0.11558
Baselinesum	0.0536912	0.0859148	0.114346	0.0858814	0.114346
Baselinemwt	0.0557855	0.0886043	0.117854	0.0887005	0.117854

Table 4. The evaluation scores of Informativeness based on NYT textual content

Run	*unigram*	*bigram*	*with 2-gap*	*Average*
ID46RJU_CSE_run1	0.0487001	0.080679	0.108948	0.0806496
ID46RJU_CSE_run2	0.0487017	0.0806804	0.10895	0.080651
Baselinesum	0.0460489	0.0781008	0.10646	0.0780837
Baselinemwt	0.0475077	0.0793851	0.10766	0.0793874

7.2 Readability Evaluation

For Readability evaluation [1] all passages in a summary have been evaluated according to Syntax (S), Anaphora (A), Redundancy (R) and Trash (T). If a passage contains a syntactic problem (bad segmentation for example) then it has been marked as Syntax (S) error. If a passage contains an unsolved anaphora then it has been marked as Anaphora (A) error. If a passage contains any redundant information, i.e., an information that have already been given in a previous passage then it has been marked as Redundancy (R) error. If a passage does not make any sense in its context (i.e., after reading the previous passages) then these passages must be considered as trashed, and readability of following passages must be assessed as if these passages were not present, so they were marked as Trash (T).

There are two readability metrics, Relaxed and Strict and they are defined as,

Relaxed metric: "Count as VALID if not Trash (T)";

Strict metric: "Count as VALID if not Trash (T) or Redundancy (R) or Anaphora (A) or Syntax (S)".

In both cases, the score is the average, normalized number of words in valid passages. Summary word numbers are normalized to 500 words each. Relaxed score can (rarely) be lower than strict score, as assessor can consider as "not trash" a passage with anaphora or syntax error. The readability evaluation scores are shown in the table 5. Participants are ranked according to this score. Our run 1's relaxed metric score is the best score and strict metric score is the 4th best score among all the runs from all the participants.

Table 5. The evaluation scores of Readability

Run	*Relaxed Metric Score*	*Strict Metric Score*
ID46RJU_CSE_run1	432.2000	347.9200
ID46RJU_CSE_run2	416.5294	330.1400
Baselinesum	447.3019	409.9434
Baselinemwt	137.8000	148.2222

8 Conclusion and Future Works

The question answering system has been developed as part of the participation in the Question Answering track of the INEX 2011 evaluation campaign. The overall system has been evaluated using the evaluation metrics provided as part of the QA track of INEX 2011. Considering that this is the first participation in the track, the evaluation results are satisfactory as readability scores are very high and in the relaxed metric we got the highest score of 432.2, which will really encourage us to continue work on it and participate in this track in future.

Future works will be motivated towards improving the performance of the system by concentrating on co-reference and anaphora resolution, named entity (NE) identification, multi-word identification, para phrasing, feature selection etc. In future, we will also try to use semantic similarity, which will increase our relevance score.

Acknowledgements. We acknowledge the support of the IFCPAR funded Indo-French project "An Advanced Platform for Question Answering Systems"

References

1. SanJuan, E., Moriceau, V., Tannier, X., Bellot, P., Mothe, J.: Overview of the INEX 2011 Question Answering Track (QA@INEX). In: Focused Retrieval of Content and Structure, 10th International Workshop of the Initiative for the Evaluation of XML Retrieval (INEX), Geva, S., Kamps, J., Schenkel, R. (Eds.). Lecture Notes in Computer Sc., Springer (2011)
2. Jezek, K., Steinberger, J.: Automatic Text summarization. In: Snasel, V. (ed.) Znalosti 2008. ISBN 978-80-227-2827-0, pp.1--12. FIIT STU Brarislava, Ustav Informatiky a softveroveho inzinierstva (2008)
3. Erkan, G., Radev, D.R.: LexRank: Graph-based Centrality as Salience in Text Summarization. In: Journal of Artificial Intelligence Research, vol. 22, pp. 457--479 (2004)
4. Hahn, U., Romacker, M.: The SYNDIKATE text Knowledge base generator. In: the first International conference on Human language technology research**,** Association for Computational Linguistics , ACM, Morristown, NJ, USA (2001)
5. Kyoomarsi, F., Khosravi, H., Eslami, E., Dehkordy, P.K.: Optimizing Text Summarization Based on Fuzzy Logic. In: Seventh IEEE/ACIS International Conference on Computer and Information Science, pp. 347--352. IEEE, University of Shahid Bahonar Kerman, UK (2008)
6. Bhaskar, P., Bandyopadhyay, S.: 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)
7. Bhaskar, P., Bandyopadhyay, S.: A Query Focused Automatic Multi Document Summarizer. In: the International Conference on Natural Language Processing (ICON), pp. 241--250. IIT, Kharagpur, India (2010)
8. Rodrigo, A., Iglesias, J.P., Pe˜nas, A., Garrido, G., Araujo, L.: A Question Answering System based on Information Retrieval and Validation, ResPubliQA (2010)
9. Schiffman, B., McKeown, K.R., Grishman, R., Allan, J.: Question Answering using Integrated Information Retrieval and Information Extraction. In: NAACL HLT, pp. 532--539 (2007)
10. Pakray, P., Bhaskar, P., Pal, S., Das, D., Bandyopadhyay, S., Gelbukh, A.: JU_CSE_TE: System Description QA@CLEF 2010 – ResPubliQA. In: Multiple Language Question Answering (MLQA 2010), CLEF-2010, Padua, Italy (2010)
11. Pakray, P., Bhaskar, P., Banerjee, S., Pal, B.C., Bandyopadhyay, S., Gelbukh, A.: A Hybrid Question Answering System based on Information Retrieval and Answer Validation. In: Question Answering for Machine Reading Evaluation (QA4MRE), CLEF-2011, Amsterdam (2011)
12. Tombros, A., Sanderson, M.: Advantages of Query Biased Summaries in Information Retrieval. In: SIGIR (1998)
13. Radev, D.R., Jing, H., Styś, M., Tam, D.: Centroid- based summarization of multiple documents. J. Information Processing and Management. 40, 919–938 (2004)
14. Lin, C.Y., Hovy, E.H.: From Single to Multidocument Summarization: A Prototype System and its Evaluation. In: ACL, pp. 457--464 (2002)
15. Hardy, H., Shimizu, N., Strzalkowski, T., Ting, L., Wise, G. B., Zhang. X.: Cross-document summarization by concept classification. In: SIGIR, pp. 65--69 (2002)
16. Paladhi, S., Bandyopadhyay, S.: A Document Graph Based Query Focused Multi-Document Summarizer. In: the 2nd International Workshop on Cross Lingual Information Access (CLIA), pp. 55-62 (2008)