Tweet Contextualization (Answering Tweet Question) – the Role of Multi-document Summarization

Pinaki Bhaskar, Somnath Banerjee, and Sivaji Bandyopadhyay

Department of Computer Science and Engineering,
Jadavpur University, Kolkata - 700032, India
{pinaki.bhaskar, s.banerjee1980}@gmail.com, sivaji_cse_ju@yahoo.com

Abstract. The article presents the experiments carried out as part of the participation in the Tweet Contextualization (TC) track of INEX 2013. In our system there are three major sub-systems; i) Offline multi-document summarization, ii) Focused IR and iii) online multi-document Summarization. The Offline multi-document summarization system is based on document graph, clustering and sentence compression. In the Focused IR system, Wikipedia documents are indexed using Lucene with NE field. The most relevant documents are retrieved using the tweet. Online multi-document summary are generated from the most relevant Wikipedia documents and the offline summary of entity (if any). Most relevant sentences are retrieved and each retrieved sentence is assigned a ranking score in the online summary with a limit of 500 words. The three unique runs differ in the way of how many documents are retrieved per tweet. The evaluation score of informativeness is 0.9397 and Readability is 46.72% and both of which achieved 7th rank among the automatic runs.

Keywords: Question Answering, Multi-document Summarization, Automatic Summarization, Information Retrieval, Information Extraction, Tweet Contextualization, INEX 2013

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 techniques. More recently, Conference and Labs of Evaluation Forums (CLEF)[2] are

[1] http://trec.nist.gov/
[2] http://www.clef-initiative.eu//

organizing QA lab from 2010. INEX[3] has also started Question Answering track. 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). In 2012 they renamed this task as Tweet Contextualization [2].

Current INEX 2013 Tweet Contextualization (TC) track gives QA research a new direction by fusing IR and summarization with QA. The TC track of INEX 2013 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 [3] 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 ([4] and [5]) attempts to develop an understanding of the main concepts in a document and then expresses those concepts in clear natural language. Extractive Summaries [6] 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 Tweet Contextualization system of offline multi-document summarization, focused IR and online multi-document summarization for TC track of INEX 2013. The offline multi-document summarization system is based on document graph, clustering and sentence compression. The focused IR system is based on Nutch architecture and the online multi-document summarization system is based on TF-IDF based sentence ranking and sentence extraction techniques. The same sentence scoring and ranking approach of [7] and [8] has been followed. We have submitted three runs in the TC track (267, 270 and 271).

2 Related Works

Recent trend shows hybrid approach of tweet contextualization using Information Retrieval (IR) can improve the performance of the TC system. Reference [9] removed incorrect answers of QA system using an IR engine. Reference [10] successfully used methods of IR into QA system. Reference [11] used the IR system into QA and [12] and [13] proposed an efficient hybrid QA system using IR in QA.

Reference [14] presents an investigation into the utility of document summarization in the context of IR, more specifically in the application of so-called

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

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.

A lot of research work has been done in the domain of both query dependent and independent summarization. MEAD [15] is a centroid based multi document summarizer, which generates summaries using cluster centroids produced by topic detection and tracking system. NeATS [16] selects important content using sentence position, term frequency, topic signature and term clustering. XDoX [17] 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 [18], [7] and [8].

In the present work, we have used the offline multi-document summarization system as described in [7] and [8]. The IR system as described in [11], [12], [13], [19] and [20] and the online multi-document summarization system as discussed in [19], [20], [21] and [22]. In the later part of this paper, section 3 describes the corpus statistics and section 4 shows the system architecture of combined TC system of offline summarization, focused IR and online summarization for INEX 2013. The Offline Multi-document Summarization technique is described in section 5. Section 6 details the Focused Information Retrieval system architecture. Section 7 details the Online Multi-document Summarization system architecture. The evaluations carried out on submitted runs are discussed in Section 8 along with the evaluation results. The conclusions are drawn in Section 9.

3 Corpus statistics

The training data is the collection of documents that has been rebuilt based on recent English Wikipedia dump (from November 2012). 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.

A two columned text file containing the Entity list are provided in the test data set. The file contains all the entities extracted from the entire Wikipedia dump. Entity id is in the first column and corresponding entity is given in the second column. There are total 39,02,345 entities in the list.

Test data is made up of 598 tweets in English have been collected by the organizers from Twitter®. They were selected among informative accounts (for example, @CNN, @TennisTweets, @PeopleMag, @science...), in order to avoid purely personal tweets that could not be contextualized. There are two different formats of tweets, one is the full JSON format with all information such as the user name, tags or URLs as shown in the table 2 and another is a single xml file with three fields: topic, title and txt. The topic field contains the tweet id as attribute, the title field shows the tweet text, for people not wanting to bother with JSON format and the txt field contains full JSON format with all tweet metadata, as shown in the table 3.

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. A full JSON format with all tweet metadata of INEX 2013 test corpus

```
"created_at":"Fri, 03 Feb 2012 09:10:20 +0000",
"from_user":"XXX",
"from_user_id":XXX,
"from_user_id_str":"XXX",
"from_user_name":"XXX",
"geo":null,
"id":XXX,
"id_str":"XXX",
"iso_language_code":"en",
"metadata":{"result_type":"recent"},
"profile_image_url":"http://XXX",
"profile_image_url_https":"https://XXX",
"source":"<a href='http://XXX'>",
"text":"blahblahblah",
"to_user":null,
"to_user_id":null,
"to_user_id_str":null,
"to_user_name":null
```

Table 3. An example of a tweet topic in xml format of INEX 2013 test corpus

```
<topic id="306410030352195585">
        <title>How does pop art differ across the world and cultures?
        http://t.co/j6oFJK4pwp</title>
        <txt>
            {"source":"&lt;a href=&quot;http://www.hootsuite.com&quot;&gt;
            HootSuite&lt;/a&gt;", "geo":null, "profile_image_url":
            "http://a0.twimg.com/profile_images/1326615844/
            twitterlogo_normal.jpg", "id_str": "306410030352195585",
            "from_user_id":5225991, "profile_image_url_https":
            "https://si0.twimg.com/profile_images/1326615844/
            twitterlogo_normal.jpg", "iso_language_code": "en", "created_at":
            "Tue, 26 Feb 2013 14:26:58 +0000", "text": "How does pop art differ
            across the world and cultures? http://t.co/j6oFJK4pwp", "metadata":
            {"result_type":"recent"}, "from_user_name":"Tate", "id":
            306410030352195585, "from_user_id_str": "5225991", "from_user":
            "Tate"}
        </txt>
</topic>
```

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 three major sub-systems; i) Offline multi-document summarization, ii) Focused IR and iii) online multi-document Summarization. The Offline multi-document summarization system is based on document graph and clustering. 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 Tweet Contextualization system of INEX 2013 is shown in the Figure 1.

5 Offline Multi-document Summarization

All the entities in the Wikipedia documents are tagged using the entity tag and a list of all the entities present in the entire test Wikipedia documents are provided along with the test data set. Entities are the key terms or topic of a document. So, we retrieved most relevant 10 Wikipedia documents for each entity and then we generated a multi-document summary for each entity. This process does not need the tweets and that's why it is a complete offline process. We can prepare the offline

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

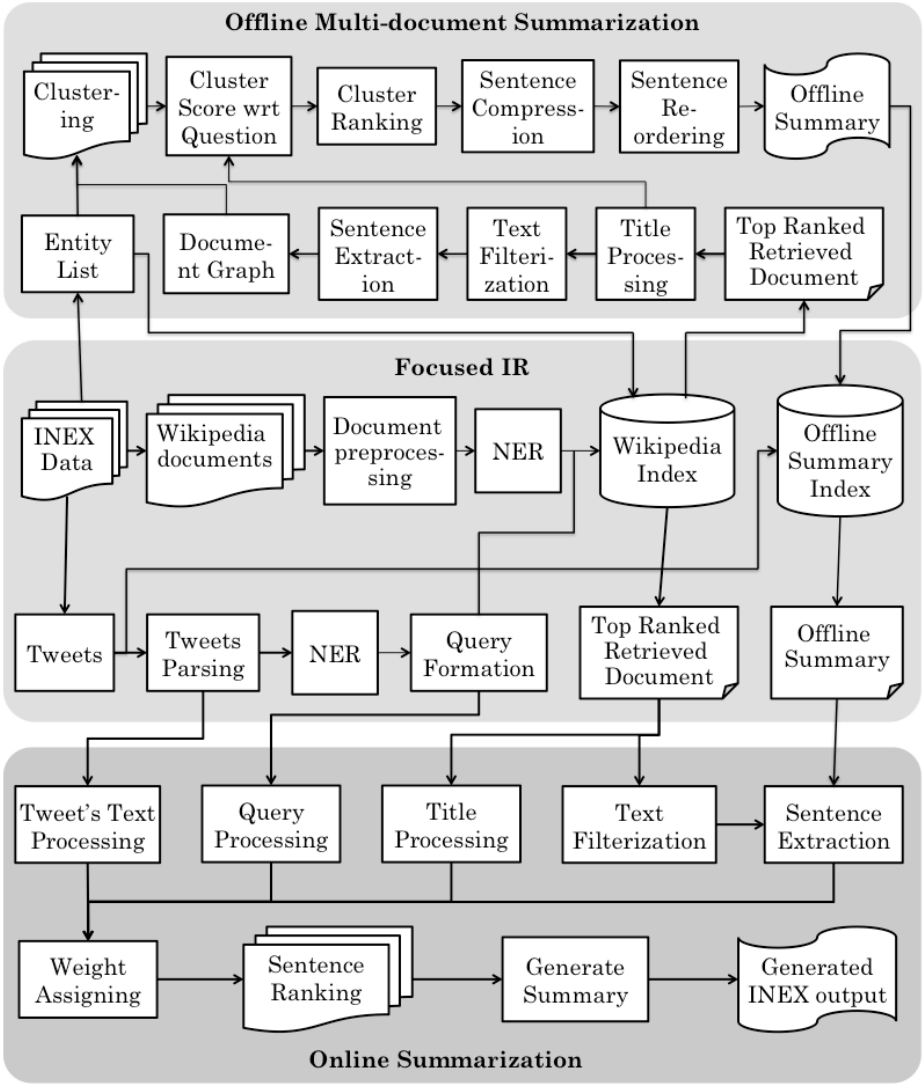

Fig. 1. Higher level system architecture of current INEX system

summaries using the Wikipedia documents and their entities only. For retrieval, traditional Lucene indexer and Nutch have been used.

5.1 Wikipedia Document Parsing, Indexing and Retrieval

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 2013 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.

Named Entity Recognizer (NER). After cleaning the corpus, the named entity recognizer identifies all the named entities (NE) in the documents using Stanford NER engine[6] and tags them according to their types, which are indexed during the document indexing.

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

Document Retrieval. After indexing, each entity is searched into the Lucene index using Nutch and a set of retrieved top 10 documents in ranked order for each entity is received. First of all, all entities were fired with AND operator. If at least ten documents are retrieved using the entity with AND operator then the entity is removed from the entity list and need not be searched again. If less than 10 documents are retrieved using AND search then the entity are fired again with OR operator. OR searching retrieves at least 10 documents for each entity. Now, the top 10 ranked relevant documents for each entity is considered for offline multi-document summary.

5.2 Graph based Multi-document Summarization

Graph Based Clustered Model. The proposed graph based multi-document summarization method consists of following steps:

(1) The document set $D = \{d_1, d_2, \ldots d_{10}\}$ is processed to extract text fragments, which are sentences in this system as it has been discussed earlier. Here, It has been assumed that the entire documents in a particular set are related i.e. they describe the same event. Some document clustering techniques may be adopted to find related documents from a large collection. Document clustering is out of the scope of this current discussion and is itself a research interest. Let for a document d_i, the sentences are $\{s_{i1}, s_{i2}, \ldots s_{im}\}$. But the system can be easily modified to work with phrase level

[6] http://www-nlp.stanford.edu/ner/

extraction. Each text fragment becomes a node of the graph i.e. all the sentences become a node.

(2) Next, edges are created between nodes across the documents where edge score represents the degree of correlation between inter-documents nodes.

(3) Seed nodes are extracted which identify the relevant sentences within D and a search graph is built to reflect the semantic relationship between the nodes.

(4) Now, each node is assigned a entity dependent score and the search graph is expanded.

(5) A entity dependent multi-document summary is generated from the search graph.

Each sentence is represented as a node in the graph. The text in each document is split into sentences and each sentence is represented with a vector of constituent words. If pair of related document is considered, then the inter document graph can be represented as a set of nodes in the form of bipartite graph. The edges connect two nodes corresponding to sentences from different documents.

Construct the Edge and Calculate Edge Score. The similarity between two nodes is expressed as the edge weight of the bipartite graph. Two nodes are related if they share common words (except stop words) and the degree of relationship can be measured by equation 1 adapting some traditional IR formula from [23].

$$\text{Edge\_Score} = \frac{\sum_{w \in (t(u) \cap t(v))} ((tf(t(u), w) + tf(t(v), w)) \times idf(w))}{size(t(u)) + size(t(v))} \quad (1)$$

where, $tf(d, w)$ is number of occurrence of w in d, $idf(w)$ is the inverse of the number of documents containing w, and $size(d)$ is the size of the documents in words. Actually for a particular node, total edge score is defined as the sum of scores of all out going edges from that node. The nodes with higher total edge scores than some predefined threshold are included as seed nodes.

But the challenge for multi-document summarization is that the information stored in different documents inevitably overlap with each other. So, before inclusion of a particular node (sentence), it has to be checked whether it is being repeated or not. Two sentences are said to be similar if they share for example, 70% words in common.

Construction of Search Graph. After identification of seed/topic nodes a search graph is constructed. For nodes, pertaining to different documents, edge scores are already calculated, but for intra document nodes, edge scores are calculated in the similar fashion as said earlier. Since, highly dense graph leads to higher search / execution time, only the edges having edge scores well above the threshold value might be considered.

Identification of Sub-topics through Markov Clustering. In this section, we will discuss the process to identify shared subtopics from related multi source documents. We already discussed that the subtopics shared by different news articles on same event form natural (separate) clusters of sentences when they are represented using

document graph. We use Markov principle of graph clustering to identify those clusters from the document graph.

Markov Graph Clustering Principle. The MCL algorithm is designed specifically for the settings of simple and weighted graph [24]. Given that the multi document summarization problem can be represented in the framework of weighted graph structure, it is possible to apply MCL for identifying subtopical groups already present in the input document set. MCL process consists of following steps: In the first step, the associated matrix of the input (document) graph M_G is transformed into Markov Matrix T_G according to $T_G = M_G d^{-1}$, where d denote the diagonal matrix that has diagonal entries as the column weights of M_G, thus $d_{kk} = \sum_i M_{ik}$, $d_{ij} = 0$, and $i \neq j$. The Markov matrix T_G is associated with a graph G′, which is called the associated Markov graph of G. In the second step, the MCL process simulates random walks in the Markov graph by iteratively performing two operations, expansion and inflation. The process will converge to a limit. The MCL process generates a sequence of stochastic matrices starting from the given Markov matrix. Expansion coincides with taking the power of stochastic matrix using the normal matrix product and Inflation corresponds to taking the Hadamard power (entrywise power) of the matrix, followed by scaling step, such that the resulting matrix is stochastic again, i.e., the matrix elements correspond to probability value. The MCL algorithm is described in figure 2.

```
T1 = Associated Markov Matrix
For k = 1 to ∞ do
1. T2k = Expand(T2k-1) ;
2. T2k+1 = Γr(T2k) ;
3. If T2k+1 is limit
Break;
End For
```

Fig. 2: Pseudo code of MCL Principle

The construction of entity independent part of the Markov clusters completes the document-based processing phase of the system.

Key Term Extraction. Key Term Extraction module has two sub modules, i.e., entity term extraction and Title words extraction.

Entity Term Extraction. First the entity is parsed using the Entity Parsing module. In this Entity Parsing module, the Named Entities (NE) are identified and tagged in the given entity using the Stanford NER engine. The remaining words after stop words removal are stemmed using Porter Stemmer.

Title Word Extraction. The titles of each retrieved documents are extracted and forwarded as input given to the Title Word Extraction module. After removing all the

stop words from the titles, the remaining tile words are extracted and used as the keywords in this system.

Entity Dependent Process. The nodes of the already constructed search graph are given a entity dependent score. Using the combined scores of entity independent score and dependent score, clusters are reordered and relevant sentences are collected from each cluster in order. Then each collected sentence has processed and compressed removing the unimportant phrases. After that the compressed sentences are used to construct the summary.

Recalculate the Cluster Score. There are three basic components in the sentence weight like entity terms dependent score, title words dependent score and synonyms of entity terms dependent score. We collect the list of synonyms of the each word in the entities from the WordNet 3.0[7] and form a set of synonyms.

The entity terms dependent score is calculated using equation 2.

$$w^e = \sum_{e=1}^{n_e} (n_e - e + 1) \left(\sum_p \left(1 - \frac{f_p^e - 1}{N_i} \right) \right) \times 3 \quad (2)$$

where, w^e is the entity terms dependent score of the sentence i, e is the no. of the entity term, n_e is the total no. of entity term, f_p^e is the possession of the word which was matched with the entity term e in the sentence i, N_i is the total no. of words in sentence i. A boost factor of the entity term, which is 3 for entity term, is multiplied at the end for boosting the entity dependent score.

The title words dependent score is calculated using equation 3.

$$w^t = \sum_{t=1}^{n_t} (n_t - t + 1) \left(\sum_p \left(1 - \frac{f_p^t - 1}{N_i} \right) \right) \times 2 \quad (3)$$

where, w^t is the title words dependent score of the sentence i, t is the no. of the title words, 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 i. A boost factor of the title words, which is 2, is multiplied at the end for boosting the title words dependent score.

The synonyms of entity terms dependent score is calculated using equation 4.

$$w^s = \sum_{s=1}^{n_s} (n_s - s + 1) \left(\sum_p \left(1 - \frac{f_p^s - 1}{N_i} \right) \right) \quad (4)$$

where, w^s is the synonyms dependent score of the sentence i, s is the no. of synonyms, n_s is the total no. of synonym, f_p^s is the possession of the word which was matched with the synonym s in the sentence i. There is no boosting the synonyms dependent score.

These three components i.e. w^q, w^t and w^s, are added to get the final weight of a sentence.

[7] http://wordnet.princeton.edu/

Recalculate the Cluster Ranking. We start by defining a function that attributes values to the sentences as well as to the clusters. We refer to sentences indexed by i and entity terms indexed by j. We want to maximize the number of entity term covered by a selection of sentences:

$$maximize \sum_{j} w_j^e e_j \tag{5}$$

where, w_j^e is the weight of entity term *j* in the sentence *i* and e_j is a binary variable indicating the presence of that entity term in the cluster.

We also take the selection over title words. We refer to title words indexed by *k*. We want to maximize the number of title word covered by a selection of sentences:

$$maximize \sum_{k} w_k^t t_k \tag{6}$$

where, w_k^t is the weight of title word *k* in the sentence *i* and t_k is a binary variable indicating the presence of that title word in the cluster.

To take the selection over synonyms of the entity terms, we refer to synonyms indexed by *l*. We want to maximize the number of synonyms covered by a selection of sentences:

$$maximize \sum_{l} w_l^s s_l \tag{7}$$

where, w_l^s is the weight of synonym *l* in the sentence *i* and s_l is a binary variable indicating the presence of that synonym in the cluster.

So, the entity dependent score of a cluster is the weighted sum of the entity terms it contains. If clusters are indexed by *x*, the entity dependent score of the cluster *x* is:

$$c_x^e = \sum_{i=1}^{n} \sum_{j} w_j^e e_j + \sum_{i=1}^{n} \sum_{k} w_k^t t_k + \sum_{i=1}^{n} \sum_{l} w_l^s s_l \tag{8}$$

where, c_x^q is the entity dependent score of the cluster x, n is the total no. of sentences in cluster x. Now, the new recalculated combined score of cluster x is:

$$c_x = c_x^g + c_x^e \tag{9}$$

where, c_x is the new score of the cluster *x* and c_x^g is the entity independent cluster score in the graph of cluster *x*. Now, all the clusters are ranked with their new score c_x.

Retrieve Sentences for Summary. Get the highest weighted two sentences of each cluster, by the following equation:

$$\max\left(\sum_{j} w_j^q q_j + \sum_{k} w_k^t t_k + \sum_{l} w_l^s s_l\right) \forall i \tag{10}$$

where, *i* is the sentence index of a cluster. The original sentences in the documents are generally very lengthy to place in the summary. So, we are actually interested in a selection over phrases of sentence. After getting the top two sentences of a cluster,

they are split into multiple phrases. The Stanford Parser[8] is used to parse the sentences and get the phrases of the sentence.

Sentence Compression. All the phrases which are in one of those 34 relations in the training file, whose probability to drop was 100% and also do not contain any entity term, are removed from the selected summary sentence as described by [7]. Now the remaining phrases are identified from the parser output of the sentence and search phrases that contain at least one entity term then those phrases are selected. The selected phrases are combined together with the necessary phrases of the sentence to construct a new compressed sentence for the summary. The necessary phrases are identified from the parse tree of the sentence. The phrases with `nsubj` and the `VP` phrase related with the `nsubj` are some example of necessary phrases.

Sentence Selection for Summary. The compressed sentences for summary have been taken until the length restriction of the summary is reached, i.e. until the following condition holds:

$$\sum_i l_i S_i < L \tag{11}$$

where, l_i is the length (in no. of words) of compressed sentence i, S_i is a binary variable representing the selection of sentence i for the summary and L (=1000 words) is the maximum summary length. After taking the top two sentences from all the clusters, if the length restriction L is not reached, then the second iteration is started similar to the first iteration and the next top most weighted sentence of each cluster are taken in order of the clusters and compressed. If after the completion of the second iteration same thing happens, then the next iteration will start in the same way and so on until the length restriction has been reached.

Sentence Ordering and Coherency. In this paper, we will propose a scheme of ordering; in that, it only takes into consideration the semantic closeness of information pieces (sentences) in deciding the ordering among them. First, the starting sentence is identified which is the sentence with lowest positional ranking among selected ones over the document set. Next for any source node (sentence) we find the summary node that is not already selected and have (correlation value) with the source node. This node will be selected as next source node in ordering. This ordering process will continue until the nodes are totally ordered. The above ordering scheme will order the nodes independent of the actual ordering of nodes in the original document, thus eliminating the source bias due to individual writing style of human authors. Moreover, the scheme is logical because we select a sentence for position p at output summary, based on how coherent it is with the (p-1)th sentence. The main sentence's number has been taken as the sentence number of the fused sentence.

[8] http://nlp.stanford.edu/software/lex-parser.shtml

Offline Summary. Now, each generated multi-document summary is indexed along with it entity. There are 39,02,345 entities in the entity list provided in the test data set. So it will be very time consuming to generate such a huge numbers of multi-document summaries. For the time constrain, we have simplified the task. We have selected all the entities, those are present in the tweets and then we have created multi-document summary of such matched entities.

6 Focused Information Retrieval (IR)

6.1 Tweets Parsing

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/.

Named Entity Recognizer (NER). After removing the stop words, the named entity recognizer identifies all the named entities (NE) in the tweet using Stanford NER engine and tags them according to their types, which are used during the scoring of the sentences of the retrieved document.

Stemming. Query tokens may appear in inflected forms in the tweets. For English, standard Porter Stemming algorithm[9] has been used to stem the query tokens. After stemming all the query tokens, queries are formed with the stemmed query tokens.

6.2 Document Retrieval using Tweet

The same Lucene index described in the section 5.1, has been used to retrieve documents using tweet. After searching each query into the Lucene index, a set of top 10 retrieved documents in ranked order for each tweet is received. Same Boolean logic as proposed in section 5.1, are used to retrieved the top ranked 10 relevant documents for each tweet and considered for final multi-document summarization.

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

7 Online Multi-document Summarization

The entity words are identified and marked by the # before each entity word in the tweet. So the entity words are extracted from the tweet and searched into the entity list. Then the pre-generated offline summaries of the matched entities and taken for the final / online summary generation. E.g. if a tweet contains a entity word of entity e_i and s_i is the offline summary of entity e_i, then offline summary s_i will also be taken along with the 10 retrieved Wikipedia documents for generating the final/online summary.

7.1 Sentence Extraction

The documents texts and the summary text are 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.

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[10]. 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.

7.2 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.

[10] http://en.wikipedia.org/wiki/List_of_Unicode_characters

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 engine.

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.

7.3 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 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/Tweet Term dependent score: Query/Tweet term dependent score is the most important and relevant score for summary. Priority of this query/tweet dependent score is maximum. The query dependent scores are calculated using equation 12.

$$Q_s = \sum_{q=1}^{n_q} F_q \left(20 + \left(n_q - q + 1 \right) \left(\sum_p \left(1 - \frac{f_p^q - 1}{N_s} \right) \right) \times p \right) \tag{12}$$

where, Q_S is the query/tweet term dependent score of the sentence s, q is the no. of the query/tweet 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,

$$F_q = \begin{cases} 0; \textit{if query term q is not found} \\ 1; \textit{if query term q is found} \end{cases} \tag{13}$$

and

$$p = \begin{cases} 5; \textit{ if query term is NE} \\ 3; \textit{if query term is not NE} \end{cases} \tag{14}$$

At the end of the equation 12, 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).

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 15.

$$T_s = \sum_{t=0}^{n_t} F_t\left(n_t - t + 1\right)\left(\sum_p \left(1 - \frac{f_p^t - 1}{N_s}\right)\right) \tag{15}$$

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

$$F_t = \begin{matrix} 0; \textit{if title word t is not found} \\ 1; \textit{if title word t is found} \end{matrix} . \tag{16}$$

After calculating all the above three scores the final weight of each sentence is calculated by simply adding all the two scores as mentioned in the equation 17.

$$W_s = Q_s + T_s \tag{17}$$

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.

7.4 Summary Generation

This is the final and most critical module of this system. This module generates the Summary from the ranked sentences. As in [15] using equation 11, the module selects the ranked sentences subject to maximum length L (=500 words) of the summary.

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

8 Evaluation

8.1 Informative Content Evaluation

The organizers did the Informative Content evaluation [1], [2] by selecting relevant passages. The organizers have used a standalone evaluation toolkit based on Porter stemmer and implementing a new normalized ad-hoc dissimilarity defined as following:

$$Dis(T,S) = \sum_{t \in T} \frac{f_T(t)}{f_T} \times \left(1 - \frac{\min(\log(P), \log(Q))}{\max(\log(P), \log(Q))}\right) \quad (18)$$

$$P = \frac{f_T(t)}{f_T} + 1 \quad (19)$$

$$Q = \frac{f_S(t)}{f_S} + 1 \quad (20)$$

where T is the set of terms in the reference and for every $t \in T$, $f_T(t)$ is its frequency in the reference and $f_S(t)$ its frequency in the summary. The idea was to have a dissimilarity which complement has similar properties to usual IR Interpolate Precision measures. Actually, 1–*Dis(T, S)* increases with the Interpolated Precision at 500 tokens where Precision is defined as the number of word n-grams in the reference. The introduction of the log is necessary to deal with highly frequent words.

As previously announced, we used this software to evaluate informativeness and like in INEX QA tracks, we considered as T three different sets based on Porter stemming:

– Unigrams made of single lemmas (after removing stop-words).
– Bigrams made of pairs of consecutive lemmas (in the same sentence).
– Bigrams with 2-gaps also made of pairs of consecutive lemmas but allowing the insertion between them of a maximum of two lemmas.

Informativity has been evaluated based on three overlapping references:

1. **prior** set of relevant pages selected by organizers while building the 2013 topics (40 tweets, 380 passages, 11 523 tokens),
2. **pool** selection of most relevant passages from participant submissions for tweets selected by organizers (45 tweets, 1 760 passages, 58 035 tokens),
3. **all** relevant text merged together with an extra selection of relevant passages from a random pool of ten tweets (70 tweets, 2 378 passages, 77 043 tokens)

Ranking. Runs are ranked by decreasing score of divergence with the final reference (All.skip). The organizers rank both manual and automatic runs. But we have recalculate the rank considering only the automatic runs, which has been shown here. Before our run there are two manual runs in the ranked list. So, the rank provided by the organizers are 2 more than what we have shown in the rank column.

We have submitted three runs (267, 270 and 271). The evaluation scores of informativeness by organizers of all topics are shown in the table 4.

Table 4. The evaluation scores of Informativeness by organizers of all topics

Run	Rank	All. skip	All .bi	All .uni	Pool .skip	Pool. bi	Pool .uni	Prior .skip	Prior .bi	Prior .uni
270	**7**	**0.9397**	0.9365	0.8481	0.9274	0.9246	0.8418	0.9686	0.9642	0.8529
267	8	0.9468	0.9444	0.8838	0.9389	0.9362	0.8802	0.9625	0.9596	0.883
271	9	0.95	0.9475	0.8569	0.9446	0.9421	0.8543	0.9793	0.9759	0.867

8.2 Readability Evaluation

For Readability evaluation [1], [2] 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).

Readability has been evaluated by organizers over the ten tweets having the largest text references (t-rels). For these tweets, summaries are expected to have almost 500 words since the reference is much larger. For each participant summary, we have then check the number of words over 500 in passages that are:

1. **Relevant (T)** i.e. clearly related to the tweet,
2. **Sound (A)** i.e. no issues about resolving references to earlier or later items in the discourse.
3. **Non redundant (R)** with previous passages.
4. **Syntactically (S)** correct.

Non-relevant passages have also been considered non-sound, redundant and syntactically incorrect.

Ranking. Runs are ranked according to mean average scores per summary over Soundness, Non redundancy and Syntactically correctness among Relevant passages. The readability evaluation scores are shown in the table 5. Here also we have recalculate the rank considering only the automatic runs, which has been shown here.

Table 5. The evaluation scores of Readability Evaluation

Run	Rank	Mean Average	Relevancy (T)	Non redundancy (R)	Soundness (A)	Syntax (S)
267	**7**	**46.72%**	50.54%	40.90%	49.56%	49.70%
270	8	44.17%	46.84%	41.20%	45.30%	46.00%
271	9	38.76%	41.16%	35.38%	39.74%	41.16%

9 Conclusion and Future Works

The tweet contextualization system has been developed as part of the participation in the Tweet Contextualization track of the INEX 2013 evaluation campaign. The overall system has been evaluated using the evaluation metrics provided as part of this track of INEX 2013. Considering that this is the second participation in the track, the

evaluation results are satisfactory, 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, 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 Department of Electronics and Information Technology (DeitY), Ministry of Communications & Information Technology (MCIT), Government of India funded project "Development of Cross Lingual Information Access (CLIA) System Phase II".

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. SanJuan, E., Moriceau, V., Tannier, X., Bellot, P., & Mothe, J. (2012, September). Overview of the INEX 2012 tweet contextualization track. In: Copyright cG2012 remains with the author/owner (s). The unreviewed pre-proceedings are collections of work submitted before the December workshops. They are not peer reviewed, are not quality controlled, and contain known errors in content and editing. The proceedings, published after the Workshop, is the authoritative reference for the work done at INEX. (pp. 148-159).
3. 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)
4. 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)
5. 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)
6. 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, University of Shahid Bahonar Kerman, UK (2008)
7. 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)
8. 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)
9. Rodrigo, A., Iglesias, J.P., Pe˜nas, A., Garrido, G., Araujo, L.: A Question Answering System based on Information Retrieval and Validation, ResPubliQA (2010)
10. 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)
11. 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)

12. P. Bhaskar, P. Pakray, S. Banerjee, S. Banerjee, S. Bandyopadhyay and A. Gelbukh, Question Answering System for QA4MRE@CLEF 2012, In the proceedings of the Question Answering for Machine Reading Evaluation (QA4MRE) at Conference and Labs of the Evaluation Forum (CLEF) 2012, 17-20 Sep. 2012, Rome, Italy.
13. 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)
14. Tombros, A., Sanderson, M.: Advantages of Query Biased Summaries in Information Retrieval. In: SIGIR (1998)
15. Radev, D.R., Jing, H., Styś, M., Tam, D.: Centroid- based summarization of multiple documents. J. Information Processing and Management. 40, 919–938 (2004)
16. Lin, C.Y., Hovy, E.H.: From Single to Multidocument Summarization: A Prototype System and its Evaluation. In: ACL, pp. 457--464 (2002)
17. 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)
18. 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)
19. Bhaskar, P., Banerjee, S., Neogi, S., Bandyopadhyay, S.: A Hybrid QA System with Focused IR and Automatic Summarization for INEX 2011. In: Geva, S., Kamps, J., Schenkel, R.(eds.): Focused Retrieval of Content and Structure: 10th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2011. Saarbruecken, Germany, December 12-14, 2012. Revised and Selected Papers. Lecture Notes in Computer Science Volume 7424, 2012, pp 207-218, DOI 10.1007/978-3-642-35734-3_18, ISBN 978-3-642-35733-6, ISSN 0302-9743, Springer Verlag, Berlin, Heidelberg (2012)
20. Bhaskar. P., Banerjee. S. and Bandyopadhyay.S., A Hybrid Tweet Contextualization System using IR and Summarization, In: the proceedings of the Initiative for the Evaluation of XML Retrieval, INEX 2012 at Conference and Labs of the Evaluation Forum (CLEF) 2012, Forner. P., Karlgren. J., Womser-Hacker. C. (Eds.): CLEF 2012 Evaluation Labs and Workshop, pp. 164-175, ISBN 978-88-904810-3-1, ISSN 2038-4963, Rome, Italy. (2012)
21. Bhaskar. P. and Bandyopadhyay. S., Language Independent Query Focused Snippet Generation, T. Catarci et al. (Eds.): Information Access Evaluation. Multilinguality, Multimodality, and Visual Analytics: Third International Conference of the CLEF Initiative, CLEF 2012, Rome, Italy, September 17-20, 2012. Proceedings, Lecture Notes in Computer Science Volume 7488, 2012, pp 138-140, ISBN 978-3-642-33246-3, ISSN 0302-9743, Springer Verlag, Berlin, Heidelberg, Germany, 2012.
22. Bhaskar. P. and Bandyopadhyay. S., Cross Lingual Query Dependent Snippet Generation, In: International Journal of Computer Science and Information Technologies (IJCSIT), ISSN: 0975-9646, Vol. 3, Issue 4, 2012, pp. 4603 – 4609.
23. Varadarajan, R., Hristidis, V. 2006. A system for query specific document summarization. CIKM, pp. 622--631.
24. Van Dongen, S. 2000. Graph clustering by flow simulation. PhD Thesis, University of Utrecht, The Netherlands.