A Query Focused Automatic Multi Document Summarizer

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

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

Abstract

The development of a query focused automatic multi-document summarizer has been described. A document graph is constructed, where the nodes are sentences of the documents and edge scores reflect the correlation measure between the nodes. The system clusters similar texts having related (sub) topical features from the graph using this edge score. Each cluster gets a weight and has a cluster center. Next, query dependent weights for each sentence are added to the edge score of the sentence as well as to the corresponding cluster score. Top two ranked sentence of each cluster is identified in order and compressed using a dependency parser. If all the top ranked sentences from each cluster do not fill the maximum word length limit, then a cluster may be revisited following the rank order until the maximum word length limit is reached. The compressed sentences from each cluster are fused to a single sentence. The compressed and fused sentences are included into the output summary. The summarizer has been tested on the standard TAC 2008 test data sets of the Update Summarization Track. Evaluation using the ROUGE-1.5.5 tool has resulted in ROUGE-2 and ROUGE–SU-4 scores of 0.11103 and 0.142970 respectively.

1 Introduction

Text Summarization, as the process of identifying the most salient information in a document or set of documents (for multi document summarization) and conveying it in less space, became an active field of research in both Information Retrieval (IR) and Natural Language Processing (NLP) communities. Summarization shares some basic techniques with indexing as both are concerned with identification of the essence of a document. Also, high quality summarization requires sophisticated NLP techniques in order to deal with various Parts Of Speech (POS) taxonomy and inherent subjectivity. Typically, one may distinguish various types of summarizers.

Multi document summarization requires creating a short summary from a set of documents which concentrate on the same topic. Sometimes an additional query is also given to specify the information need of the summary. Generally, an effective summary should be relevant, concise and fluent. It means that the summary should cover the most important concepts in the original document set, contain less redundant information and should be well-organized.

In this paper, we propose a query focused multi document summarizer, based on clustering technique and sentence fusion. Unlike traditional extraction based summarizers which do not take into consideration the inherent structure of the document, our system will add structure to documents in the form of graph. During initial preprocessing, text fragments are identified from the documents which constitute the nodes of the graph. Edges are defined as the correlation measure between nodes of the graph. We define our text fragments as sentence.

Since the system produces multi document summary based on user's query, the response time of the system should be minimal for practical purpose. With this goal, our system takes following steps: First, during preprocessing stage (offline) it performs some query independent tasks like identifying seed summary nodes and constructing graph over them. Then at query time (online), given a set of query words and keywords, it performs query word and keyword search over the cluster to find a sentence identifying relevant phrases satisfying the query words and keywords. With the relevant phrases, the new compressed sentence has been constructed and then fused for summary. The performance of the system depends much on the identification of relevant phrases and compression of the sentences. Although, we have presented all the examples in the current discussion for English language only, we argue that our system can be adapted to work on other language (i.e. Hindi, Bengali etc.) with some minor addition in the system like incorporating language dependent stop word list, the stemmer and the parser for the language.

2 Related Work

Currently, most successful multi-document summarization systems follow the extractive summarization framework. These systems first rank all the sentences in the original document set and then select the most salient sentences to compose summaries for a good coverage of the concepts. For the purpose of creating more concise and fluent summaries, some intensive post-processing approaches are also appended on the extracted sentences. For example, redundancy removal (Carbonell and Goldstein, 1998) and sentence compression (Knight and Marcu, 2000) approaches are used to make the summary more concise. Sentence re-ordering approaches (Barzilay et al., 2002) 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 multi-document summarization (both query dependent and independent). MEAD (Radev et al., 2004) is a centroid based multi document summarizer which generates summaries using cluster centroids produced by topic detection and tracking system. NeATS (Lin and Hovy, 2002) selects important content using sentence position, term frequency, topic signature and term clustering. XDoX (Hardy et al., 2002) 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 (Mani and Bloedorn, 2000) uses a graph-connectivity model to identify salient information. Zhang et al. (2004) proposed the methodology of correlated summarization for multiple news articles. In the domain of single document summarization a system for query-specific document summarization has been proposed (Varadarajan and Hristidis, 2006) based on the concept of document graph. A document graph based query focused multi-document summarization system has been described by Paladhi and Bandyopadhyay, (2008). In the present work, the same clustering approach has been followed. While the basic unit of clustering in (Paladhi and Bandyopadhyay, 2008) is a paragraph, sentences have been considered as the basic clustering unit in the present work. After the clusters are developed, the summarization method is completely different. In Paladhi and Bandyopadhyay's (2008) work, the minimum spanning tree identified over the document graph is identified as the summary. But in the present work we have parsed the top ranked sentences and compressed the sentences removing the unimportant or irrelevant phrases of the sentence.

3 Offline / Query Independent Process

3.1 Graph Based Clustered Model

The proposed graph based multi-document summarization method consists of following steps:

(1) The document set $D = \{d_1, d_2, \ldots d_n\}$ 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 document 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 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 offline to reflect the semantic relationship between the nodes.

(4) At query time, each node is assigned a query dependent score and the search graph is expanded.

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

3.2 Identification and Extraction of Nodes

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.

3.3 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 (Varadarajan and Hristidis, 2006).

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

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. The score can be accurately set if stemmer and lexicon are used to match the equivalent words. WordNet can be used to match the equivalent words if they are synonymous. With the idea of page ranking algorithms, it can be easily observed that a sentence in a document is relevant if it is highly related to many relevant sentences of other document. If some less stringent rules are adopted, then a node from a document is selected as topic node if it has high edge scores with nodes of other document. 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 (non stop words[1]) in common.

Offline 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.

How Clustering is useful

We experimented with some set of documents on same topics and observed the fact that apart from different views and writing style of different author, all of them contain certain subtopics about the topic and these are common to almost all documents on that topic. For example, in case of some Accidents and Natural Disasters, some common subtopics are:

[1] The stop word list used can be found at http://members.unine.ch/jacques.savoy/clef/.

WHAT: what happened,
WHEN: date, time, other temporal placement information,
WHERE: physical location,
WHY: reasons for accident/disaster,
WHO_AFFECTED: casualties (death, injury), or individuals otherwise negatively affected by the accident/disaster,
DAMAGES: damages caused by the accident/disaster,
COUNTERMEASURES: countermeasures, rescue efforts, prevention efforts, other reactions to the accident/disaster etc.

In case of some topic related to Health and Safety, some common subtopics are:

WHAT: what is the issue
WHO_AFFECTED: who is affected by the health/safety issue
HOW: how they are affected
WHY: why the health/safety issue occurs
COUNTERMEASURES: countermeasures, prevention efforts

Thus, if we can automatically detect these set of common sub topics, they will indeed be good candidates for summary of that topic/event.

Markov Graph Clustering Principle

The MCL algorithm is designed specifically for the settings of simple and weighted graph (Van Dongen,S., 2000b). 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≠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 1.

```
T1 = Associated Markov Matrix
For k = 1 to ∞ do
1. $T_{2k}$ = Expand($T_{2k-1}$) ;
2. $T_{2k+1}$ = $\Gamma_r$($T_{2k}$) ;
3. If $T_{2k+1}$ is limit
Break;
End For
```

Figure 1: Pseudo code of MCL Principle

Eventually, iterating expansion and inflation operation results in the separation of the graph into different segments. There are no longer any paths between these segments and the collection of resulting segments is simply interpreted as clustering. The inflation operator can be altered using the parameter r and generally, it can be varied from 1 to 5. Increasing the operator has effect of making the inflation operation stronger, and this increases the granularity of clusters.

Thus in terms of stochastic flow, expansion causes flow to dissipate within clusters whereas inflation removes flow between different clusters. It continues until stable state is reached, i.e., formation of doubly idempotent matrix (limit matrix) that does not change with further expansion or inflation steps. The graph associated with such matrix consists of different connected components each of which is interpreted as a cluster.

The construction of query independent part of the Markov clusters completes the offline processing phase of the system.

4 Online / Query Dependent Process

At query time, first, the nodes of the already constructed search graph are given a query dependent score. The score signifies the relevance of the sentence with respect to given query and keywords. With the combined scores of query independent score and query 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.

4.1 Recalculate the Cluster Score and Cluster Ranking

We start by defining a function that attributes values to the sentences as well as to the clusters. Query focused Summarization can thus be expressed as an optimization problem, a search over clusters subject to query and keywords. For clarity, we refer to sentences (indexed by *i*), though in general these could be replaced by any textual units (paragraphs, sentences, phrases etc.). Query words (indexed by j) are a general set of query words which will be word n-grams in our experiments. We want to maximize the number of query words covered by a selection of sentences:

$$maximize \sum_j w_j^q q_j \qquad (1)$$

where w_j^q is the weight of query word *j* in the sentence *i* and q_j is a binary variable indicating the presence of that query word in the cluster. While this function gives a selection over query words, similarly we also take the selection over keywords. Keywords which belong to a general set, are indexed by *l*. So we also want to maximize the number of keywords covered by a selection of sentences:

$$maximize \sum_l w_l^k k_l \qquad (2)$$

where w_l^k is the weight of keyword *l* in the sentence *i* and k_l is a binary variable indicating the presence of that keyword in the cluster.

We also take the selection over the synonyms of the query words and keywords. We collect the list of synonyms of the each words in the query and keywords from the WordNet 3.0[2]. The general set of synonyms are indexed by *s*. So we also want to maximize the number of synonyms covered by a selection of sentences:

$$maximize \sum_s w_s^s s_s \qquad (3)$$

So, the query dependent score of a cluster is the weighted sum of the query words and keywords it contains. If clusters are indexed by *x*, the query dependent score of the cluster *x* is:

$$c_x^q = \sum_{i=x_1}^{x_n} \sum_j w_j^w q_j + \sum_{i=x_1}^{x_n} \sum_l w_l^k k_l + \sum_{i=x_1}^{x_n} \sum_s w_s^s k_s \qquad (4)$$

where c_x^q is the query dependent score of the cluster *x*, x_1 is the starting sentence no. and x_n is the ending sentence no. of the cluster *x*. Now, the new recalculated combined score of cluster *x* is:

$$c_x = c_x^g + c_x^q \qquad (5)$$

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

4.2 Retrieve Sentences for Summary

Get the highest weighted two sentences of each cluster, by the following equation:

$$\max_i \left(\sum_j w_j^q q_j + \sum_l w_l^k k_l + \sum_s w_s^s k_s \right) \forall i, x_1 \le i \le x_n \qquad (6)$$

where x_1 is the first sentence and x_n is the nth i.e. last sentence of a cluster.

The highest weighted two sentences are taken from each cluster in order one by one. 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 is used to parse the sentences and get the phrases of the sentence.

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

4.3 Sentence Compression

Build Knowledge to Compress Sentence

TAC[3] 2008 Update Summarization's data set has 96 (48x2) sets of document and evaluation data set has 4 model summaries for each set of documents, i.e. there are 384 (96x4) model summaries. A comparison matrix has been developed by comparing the sentences of model summaries and the original sentences of the documents. The summary of the top three participants were also taken for this comparison. All the original sentences of the documents were compared with all the sentences of four model summaries and top tree participant's summaries. If 90% of a summary sentence (s_n) matched with a original sentence (o_n) from the documents then it consider that s_n summary sentence was taken from o_n original sentence. The comparison matrix has eight columns where 1st column of each row contains the original sentence in the documents from which the summary sentence has been extracted or generated and next corresponding columns of each row contain the summary sentences.

Now, all the sentences of the generated matrix have been parsed using the Stanford parser[4]. Now the training file has been developed to identify which relations of the original sentence (i.e. the sentence in the 1st column) are not present or dropped in the corresponding summary sentences. After developed the training file we observed that out of 104 relations, 34 relations have been dropped every time, i.e. the probability to drop those 34 relations is 100%. Those 34 relations are

```
conj_but, csubj, measure,
predet, prep,
prep_according_to,
prep_along, prep_amid,
prep_around, prep_as,
prep_because_of,
prep_besides, prep_beyond,
prep_compared_with,
prep_concerning, prep_during,
prep_following,
prep_including, prep_like,
prep_near, prep_onto,
prep_out_of, prep_over,
prep_such_as, prep_through,
prep_throughout, prep_until,
prep_within, prepc_about,
prepc_after, prepc_as,
prepc_from, prepc_while,
prepc_with and purpcl.
```

Similarly 4 relations have not been dropped ever, i.e. those 4 relations should not drop in the compressed sentence. Those 4 relations are

```
conj_negcc, prep_behind,
prep_without and
prepc_between.
```

Compress Sentence for Summary

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 query word, are removed from the selected summary sentence. Now the remaining phrases are indentified from the parser output of the sentence and search phrases that contain at least one query word 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.

Figure 2 shows an example of compression of a sentence for summary. In this example the original sentence has an relation '`prep_as`' which is one of those 34 relations. So the phrase with this relation should drop if the phrase does not contain any query word or keyword. As this phrase does not contains any query word or keyword, it were dropped in the compressed summary sentence. In the Figure 2, this unimportant phrase is highlighted.

4.4 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:

$$\sum_i l_i S_i < L \qquad (7)$$

[3] http://www.nist.gov/tac/
[4] http://nlp.stanford.edu/software/lex-parser.shtml

Topic: Airbus A380
Original Sentence: The A380 will take over from the Boeing 747 as the biggest jet in the skies.

Parser's output of the original sentence:

```
(ROOT
  (S
    (NP (DT The) (NN A380))
    (VP (MD will)
      (VP (VB take)
        (PRT (RP over))
        (PP (IN from)
          (NP (DT the) (NNP Boeing) (CD 747)))
        (PP (IN as)
          (NP
            (NP (DT the) (JJS biggest) (NN jet))
            (PP (IN in)
              (NP (DT the) (NNS skies)))))))
    (. .)))

det(A380-2, The-1)
nsubj(take-4, A380-2)
aux(take-4, will-3)
prt(take-4, over-5)
det(Boeing-8, the-7)
prep_from(take-4, Boeing-8)
num(Boeing-8, 747-9)
det(jet-13, the-11)
amod(jet-13, biggest-12)
prep_as(take-4, jet-13)
det(skies-16, the-15)
prep_in(jet-13, skies-16)
```

Compressed Sentence: The A380 will take over from the Boeing 747.

Figure 2: An example of sentence compression

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 (=100 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.

4.5 Sentence Fusion

As all the selected sentences from a cluster contain at least 70% common information, they can be fused to a single sentence. After selection and compression of the top ranked selected sentences from each cluster, they are fused using a simple fusion technique. All the selected sentences of a cluster are fused into a single sentence. So, after the sentence fusion we get single sentence from each cluster.

Phrase Extraction

All the possible phrases are extracted from all the sentences of a cluster using the Stanford parser. In table 1, an example of the extracted phrases of a sentence has been shown. With the help of the dependencies given by the Stanford parser, the phrases are extracted. If one phrase contains another sub-phrase, then it will be separated and the sub-phrase will dependent on the main phrase. In the example of table 1, phrases 4 and 5 were one phrase. But when it was separated, the sub-phrase 5 depended on main phrase 4.

No.	Phrase	Dependency
1	The A380	0
2	will take over	1
3	from the Boeing 747	2
4	as the biggest jet	2
5	in the skies	4

Table 1: Example of the extracted phrases of a sentence with their dependency.

Phrase Selection

Now, all the extracted phrases of each sentence are compared with each other and identify the common phrases. If one phrase is completely same with a part of another phrase then the smaller phrase will not be selected in the final fused sentence. If two phrases are common in two sentences, then the common phrases are marked. After the identification of the common phrases, in which sentence maximum phrases are common with another phrases of another sentences, all the common phrases of that sentence will not be selected. Then rest of the pair of common phrases in the rest of the sentences the common phrases in the lower ranked sentence will not be selected in the final fused sentence. After this process all the unique or uncommon phrases has been selected for the generation of the fused sentence.

Sentence Generation

After successfully identify the unique or uncommon phrases in the sentences, the single sentence generation has to be done. First, the sentence with most number of unique or uncommon phrases has to be identified and taken as the main sentence. Rest of the sentences are merged or fused into this main sentence. Some hand craft rules are developed to fuse the sentences. The unique or uncommon phrases of the rest of the sentences are places into the main sentence following those rules. If rest of the phrases could not be placed into the main sentence in the proper position, then the sentence are placed as a compound clause in the main sentence. After this process, a single sentence from each cluster has been produced for each cluster. These fused sentences are used to generate the final summary.

5 Sentence Ordering and Coherency

In the previous sections, the techniques for generation of summary sentences have been discussed. Here, we will investigate the method for ordering them into a coherent text. In case of single document summarization, sentence/paragraph ordering is done based on the position of extracted sentences/paragraphs in the original document. But in multi-document scenario, the problem is nontrivial since information is extracted from different documents and no single document can provide ordering. Besides, the ordering of information in two different documents may be significantly varying because the writing styles of different authors are different. In case of news event summarization, chronological ordering is a popular choice which considers the temporal sequence of information pieces, when deciding the ordering process.

In this paper, we will propose a scheme of ordering which is different from the above two approaches 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.

6 Corpus Preparation

The web documents are full of noises mixed with the original content. In that case it is very difficult to identify and separate those noises form the actual content. TAC 2008 data had many noise in the documents and the documents are in tagged format. So, first of all the document had to preprocess. The document structure is checked and reformatted as per the system requirements.

6.1 Clean Tags

The TAC 2008 data is well structured with the several tag set. The Query or Topic and the title or headline and narrative filed which are separated with the tag were extracted from the document. Then all the tags were removed from the documents.

6.2 Remove Noise and Symbols

The TAC 2008 data has some Noise as well as some special symbols. The list of some noise and special symbols are '`&amp;`', '`&Cx1f;`', '`&Cx13;`', '`&Cx11;`', '`&HT;`', '`&QL;`', '`___`',

'///' etc. This kind of noisy symbols had to be identified manually. Then all the above mentioned symbols, multiple spaces and multiple blank lines were automatically searched and removed in the corpus cleaning process.

6.3 Extract Query words and Keywords

The Query and the title or headline and narrative filed have been processed. The stop words and the common words like 'Describe' are removed from all the fields and remaining words were stemmed using Porter stemmer[5] to identify the proper query words. In this process proper query words are retrieved from the title field and the list of keywords from the narrative field.

6.4 Sentence Extraction

In English language sentence identification is ambiguous. Because the main sentence delimiter '.' is not used only for the sentence delimiter. The abbreviations like 'Mr.', 'Prof.', 'Dr.', 'U.S.A.' etc or points like '8.5' can create ambiguity. So after cleaning all the documents, the sentences were identified properly and extracted from the document. We have only a few simple rules for removing formatting markup and such a hand-crafted list of rules improve both content and linguistic quality.

7 Evaluation

Evaluation of summarization methods is generally performed in two ways. Evaluation measure based on information retrieval task is termed as the extrinsic method, while the evaluation based on user judgments is called the intrinsic measure. We adopted the first method. We evaluate our summaries by ROUGE[6], an automatic evaluation tool. We have run our system on Text Analysis Conference (TAC, formerly DUC, conducted by NIST) 2008 Update Summarization track's data sets[7]. This data set contains 48 topics and each topic has two sets of 10 documents, i.e. there are 960 documents. The evaluation data set has 4 model summaries for each document set, i.e. 8 model summaries for each topic. We have evaluated our output summaries on those model summaries using ROUGE-1.5.5. The baseline scores provided by the organizer were 0.058 and 0.093 of ROUGE-2 and ROUGE-SU4 respectively. The system's score is 0.103 and 0.140 respectively. All the results are shown in Table 2.

ROUGE Evaluation	Score				
	Average_R			Average_P	Average_F
	Proposed System	Baseline	Top score		
ROUGE -1	0.53291			0.51216	0.52118
ROUGE-2	**0.11103**	**0.058**	**0.111**	**0.09735**	**0.09991**
ROUGE -3	0.03475			0.03223	0.03336
ROUGE -4	0.01604			0.01471	0.01530
ROUGE -L	0.39162			0.37607	0.38282
ROUGE -W-1.2	0.13060			0.23011	0.16870
ROUGE- SU4	**0.142970**	**0.093**	**0.143**	**0.13361**	**0.13636**

Table 2: Evaluation scores of ROUGE

8 Conclusion and Future Work

In this work we present a graph based approach for query dependent multi document summarization system. Considering its possible application in the web environment, we have clearly divided the computation into two segments. Extraction of seed/topic summary nodes to construct offline graph and cluster the document set, is a part of query independent computation. At query time, the pre-constructed clusters are processed to extract the best multi document summary. We have tested our algorithm with news articles from TAC 2008 data of Update Summarization track. The experimental results suggest that our algorithm is effective. We experimented with 48 document sets each of 10 news articles on a same topic. The proposed algorithm can be improved to handle more noisy WEB articles or work on other domain too.

[5] http://tartarus.org/~martin/PorterStemmer/java.txt
[6] http://berouge.com/default.aspx
[7] http://www.nist.gov/tac/data/index.html

The important aspect of our system is that it can be modified to compute query independent summary which consists of topic nodes, generated during preprocessing stage. The sentence ordering module can be used to define ordering among those topic sentences. Another important aspect is that our system can be tuned to generate summary with custom size specified by users. Lastly, it is shown that our system can generate summary for other non-English documents also if some simple resources of the language are available.

The performance of our algorithm greatly depends on quality of selection of topic nodes. So if we can improve the identification of topic phrases rather than topic sentences at the time of generating the graph and shared topics among multiple documents it would surely enhance the quality of our system, because phrases can be more compressed and relevant to the query and keywords.

The top ranked sentences from each cluster are fused to a single sentence. The fused sentences are used to generate the final output summary. But with the ROUGE evaluation, the grammatical correctness can not be evaluated. So the evaluation to check the grammatical correctness of the fused sentences yet to be done properly.

Now we are also working on generating the update summary. In the News domain Update Summary is a very important and useful concept. On a same news topic every day or every hour there are some new or updated news arrived. So one who already read the previous news article, (s)he will not be interested to read the whole article again. (S)He will want to know the updated news only. With the help of the Update summary, reader can read and track news very easily. Later we will develop a system which will produce the update summary too.

Acknowledgments

The work has been carried out with support from IFCPAR funded Project “An Advanced platform for question answering systems” (Project No. 4200-IT-1).

References

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

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

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

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

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

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

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

Porter, M. 1980. An algorithm for suffix stripping. Program, 14(3), 130–137.

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

Van Dongen,S. 2000a. A stochastic uncoupling process for graphs. Report No. INS- R0011, Centre for Mathematics and Computer Science(CWI), Amsterdam.

Van Dongen,S. 2000b. Graph clustering by flow simulation. PhD Thesis, University of Utrecht, The Netherlands.

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

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.

Ashok K. Chandra, Dexter C. Kozen, and Larry J.Stockmeyer. 1981. Alternation. *Journal of the Association for Computing Machinery*, 28(1):114-133.

Dan Gusfield. 1997. *Algorithms on Strings, Trees and Sequences*. Cambridge University Press, Cambridge, UK.