A New Automatic Multi-Document Text Summarization using Topic Modeling

Rajendra Kumar Roul[1], Samarth Mehrotra[1], Yash Pungaliya[1], Jajati Keshari Sahoo[2]

[1] Department of Computer Science, BITS-Pilani, K.K.Birla Goa Campus, Zuarinagar, Goa-403726, India, rkroul@goa.bits-pilani.ac.in, samarth.1397@gmail.com, yashpungaliya@gmail.com

[2] Department of Mathematics, BITS-Pilani, K.K.Birla Goa Campus, Zuarinagar, Goa-403726, India, jksahoo@goa.bits-pilani.ac.in

Abstract. Summarization is a technique that can help in making an effective and efficient use of a large amount of textual data. In this paper, we propose a novel methodology to generate an extractive text summary from a corpus of documents. Unlike most existing methods, our approach is designed in such a way that the final generated summary covers all the important topics from the corpus of documents. We propose a heuristic method which uses the Latent Dirichlet Allocation algorithm to identify the optimum number of independent topics present in a corpus of documents. Some of the sentences are identified as the important sentences from each independent topic using a set of word and sentence level features. In order to ensure that the final summary is coherent, we suggest a novel technique to reorder the sentences based on sentence similarity. The use of topic modeling ensures that all the important content from the corpus of documents is captured in the extracted summary which in turn strengthen the summary. The proposed approach is experimented on DUC datasets and the empirical results found are promising.

Keywords: Extractive, Multi-document, ROUGE, Summarization, TF-IDF, Topic Modeling

1 Introduction

The tremendous growth of the internet and portable computing systems has resulted a boom in the amount of data generated. Analysis and interpretation of such large amounts of data using various technologies and generating a brief summary is one of the most active research areas in the field of computer science. According to WordNet [1], a summary can be defined as 'some brief statements that present the main aspects of a subject (i.e., a corpus of documents) in a concise manner'. Some of the techniques that have been proposed for text summarization include 'graph-based summarization', 'sentence correlation based summarization', 'clustering-based summarization', 'supervised machine learning based summarization', and 'summarization based on Fuzzy logic' [2]. The tools and techniques used for automatic text summarization help to convert raw textual information into a summary without any human intervention. The atomic stages of the summarization are *interpretation*, *transformation*, and *generation*.

The first technique for automatic text summarization was formulated and published around 60 years back [3]. Early methods mostly focused on using term frequency as a criterion of ranking the importance of sentences. However, the computing systems were not powerful enough to carry out such complex tasks on large volume of data and hence, the progress in this field was limited. In spite of such constraints, researchers continued to hypothesize newer and more efficient methods to improve the accuracy of the generated summaries. Some of the ideas introduced are related to the 'importance of certain keywords in documents', 'importance of the position of sentences in documents', 'similarity among sentences of a document' etc. Based on the nature of the summary generated from text documents, summarization can be of two types: *abstractive* and *extractive* [4]. Abstractive summaries are generated by interpreting the raw text and generating the same information in a different and concise form. Modern day abstractive summarization systems uses complex neural network based architectures such as RNNs and LSTMs [5]. On the other hand, extractive summarization is implemented by identifying the important sections of the text, processing and combining them to form a meaningful summary [6]. Based on the number of documents used to generate the summary, text summarization can be further divided into two categories: *single* and *multi-text* summarization [7]. In single text summarization, text is summarized from one document and hence, there is no need to consider the effect of the content novelty. Multi-document text summarization systems are able to generate reports that are rich in important information, and concisely present varying views that span multiple documents, thus preferable compared to single text summarization and have become an important domain for research.

Many researchers have worked in the field of text summarization [8] [9] [10] [10] [11] [12] [13] [14] [15] [16] [17] [18], however research work in the domain of multi-document text summarization using topic modeling is limited. The following are some of the points which highlight the importance of the proposed approach and distinguish it from other existing text summarization approaches:

- The proposed approach is one of such techniques where topic modeling is used extensively for text summarization. Latent Dirichlet Allocation is used to filter out a certain number of sentences for each topic.
- A heuristic method is proposed to tackle the problem of identifying the appropriate number of topics in a corpus of documents (identification of exact number of topics in a corpus is an unsolved problem). Finally, filtering of sentences is performed using fifteen most important word and sentence level features.
- Unlike the existing techniques used for text summarization, the summary generated using the proposed approach covers all the topic that are present in the corpus of documents.
- The task of organising the extracted data and presenting it in a coherent manner has not yet received the importance that it should. The proposed heuristic method reorders the sentences based on their similarity to ensure that the final summary is coherent.

A four-step procedure is followed to generate the final summary from the corpus. In the first step, topic modeling is used to select only a subset of sentences per topic. The second step identifies fifteen most important word and sentence level features. In

the third step, an aggregate score for each sentence is calculated using those fifteen features. This aggregate score is used to select the important sentences which are to be the part of the summary. In the fourth and final step, the important sentences are ordered to form a coherent summary. Empirical results on different DUC datasets justify the suitability and importance of the proposed approach for multi-document text summarization.
Rest of the paper is as follows: Section 2 describes the detailed methodology to summarize the corpus of documents. Results and analysis of the experimentation have been carried out in Section 3, which is followed by the conclusion of the work in Section 4.

2 Proposed Approach

Consider a corpus C of documents $d = \{d_1, d_2, ..., d_i\}$. We begin by merging all these documents into a large document called D_{large}. This merging of documents does not follow any particular order. Then D_{large} is split into n sentences, i.e., $\{s_1, s_2..., s_n\}$. Our approach reduces these n sentences to n/X (i.e., length of the final summary), where X is a user defined value and greater than 2. The results are carried out by using X=13[3], as n is a reasonably large value in most collection of documents in the DUC dataset. The final summary is generated using the following steps.

1. *Using LDA to reduce number of sentences from n to 2n/X*:
   Assume that the set of sentences $\{s_1, s_2, ..., s_n\}$ consists of k independent topics. Although the problem of identification of the exact number of topics in a corpus is an unsolved problem, we propose a heuristic method which help us to decide the value of k. This heuristic method is discussed later in this step. After identifying a reasonably good k, we perform topic modeling on the set of n-sentences (each sentence is consider as an individual document) using LDA [19]. Gensim[4] a python library is used for this purpose. Based on these k-topics that are generated from the n-sentences, a sentence-topic matrix is created as shown in Table 1 where, each entry w_{ij} denotes the weight j^{th} topic of i^{th} sentence. To ensure that the final

Table 1: Sentence-Topic Matrix

	$Topic_1$	$Topic_2$	...	$Topic_k$
s_1	w_{11}	w_{12}	...	w_{1k}
s_2	w_{21}	w_{22}	...	w_{2k}
s_3	w_{31}	w_{32}	...	w_{3k}
.	.	.	.	...
.	.	.	.	...
.	.	.	.	...
s_n	w_{n1}	w_{n2}	...	w_{nk}

   summary covers all the topics, we select a total of $2n/(Xk)$ sentences from each topic (i.e., from each column of the Table 1) having maximum weight. Since this procedure is carried for k topics, we will get a total of $2n/X$ sentences. However,

[3] experimental results generate a good summary for X=13
[4] https://radimrehurek.com/gensim/

some sentences might be chosen from multiple columns, therefore, we will always have less than or equal to $2n/X$ sentences. The implementation detail of this step is illustrated in Algorithm 1.

Algorithm 1: Selection of $2n/Xk$ sentences from each topic

Input: $n \times k$ sentence-topic matrix
Output: $2n/Xk$ sentences
$reduced\_matrix[iterations][k] \leftarrow \phi$
for all $t \in (0, k-1)$ **do**
  for all $i \in (0, iterations-1)$ **do**
    $max\_elem \leftarrow max(sentence\_topic\_matrix[][t])$
    $max\_index \leftarrow index\ of\ max\_elem\ in\ matrix$
    $reduced\_matrix[i][t] \leftarrow max\_index$
    $sentence\_topic\_matrix[max\_index][t] \leftarrow 0$
  end for
end for
return $reduced\_matrix$

Heuristic method to decide the number of independent topics (k):

To decide an appropriate number of independent topics[5] (k), the following steps are used:

i. Initially the number of topics i.e., $k_{initial}$ is set to a large value, say M (M is set to $2i$ where i is the number of documents of the corpus C).
ii. Topic modeling is performed on n sentences as discussed in step 1 assuming that the number of topics as $k_{initial}$ and then the similarities between topics are calculated.
iii. Since each topic is a probability distribution over the vocabularies of C, the topic similarity [20] is computed using *KL*-Divergence [21], Hellinger's Distance[6], and Jensen Shannon Divergence [22].
iv. A topic-topic similarity matrix is generated as shown in Table 2, where $Topic_{ij}$ denotes the aggregate similarity between $Topic_i$ and $Topic_j$, and $Topic_{ii}$ =1.
v. In this entire topic-topic similarity matrix, apart from the diagonal elements (as diagonal elements are 1), if any pair of topics which have a similarity greater than 0.40 (user defined threshold) is found out then that is an indication that the initial estimate of the number of topics $k_{initial}$ is higher than the appropriate number of independent topics (k). So, we reduce $k_{initial}$ by 1, i.e., $k_{initial}$ = $k_{initial}$-1 and repeat the Steps (ii-v) again.
vi. Eventually a stage will come where we would have a number of topics k i.e., in the topic-topic similarity matrix between any pair of topics, the similarity score is less than 0.4. This will ensure that we will reach a suitable number of independent topics which serves the purpose of summarization as it covers the breadth of topics of the original corpus. Algorithm 2 discusses the details.

[5] by independent means low similarity between topics
[6] www.encyclopediaofmath.org/index.php?title=Hellinger_distance& oldid=16453

Table 2: Topic-Topic matrix

	$Topic_1$	$Topic_2$	...	$Topic_k$
$Topic_1$	$Topic_{11}$	$Topic_{12}$	...	$Topic_{1k}$
$Topic_2$	$Topic_{21}$	$Topic_{22}$	...	$Topic_{2k}$
$Topic_3$	$Topic_{31}$	$Topic_{32}$	...	$Topic_{3k}$
.	.	.	...	.
.	.	.	...	.
.	.	.	...	.
$Topic_k$	$Topic_{k1}$	$Topic_{k2}$	...	$Topic_{kk}$

Algorithm 2: Selection of appropriate number of independent topics

Input: the set of sentences $\{s_1, s_2, ..., s_n\}$
Output:num_topics //number of topics
$num\_topics \leftarrow 2 * num\_docs$// number of documents
while num_topics >0 **do**
  $ldamodel \leftarrow$ generate an LDA model using num_topics
  $similairity\_matrix[num\_topics][num\_topics] \leftarrow \phi$
  $similarity\_matrix \leftarrow$ generated topic-topic similarity matrix
  if (element $\in$ similarity_matrix) >0.4 and element is not a diagonal element **then**
    $num\_topics \leftarrow num\_topics - 1$
  else
    exit
  end if
end while
return num_topics

2. *Identifying important word and sentence level features*:
   We need to identify a set of features of the sentences that are most probable to become a part of the summary. The following features are used to generate an aggregate score for a sentence s of a document d.
   i. *Length of the sentence*: The weight of s is decided based on the number of words present in s.
   ii. *Weight of the sentence*: This feature finds the weight of s by adding the individual TF-IDF weight of each word of s.
   iii. *Sentence density* : Density of s is the ratio between the total number of keywords in s and the total number of words including stop-words of s.
   iv. *Title words in the sentence*: If most of the words of s and title of d to which s belongs are common then s is consider as an important sentence because it precisely describes d.
   v. *Upper-case words in the sentence* : Upper-case words in s are useful as they represent important acronyms, names, places, etc., hence checking the presence of those features in s are essential. If the Upper-case words exists in the sentence, then s receives a score of one else zero (which is in binary form).
   vi. *Quoted text in the sentence* : If some part of s is within the quotation marks then it asserts something specific and that information is likely to be important. If

the quote exists in the sentence, then s receives a score of one else zero (which is in binary form).

vii. *Numerical words in the sentence*: The proportion of the numerical words in the total words is calculated and accordingly the score is assigned to the sentence. More the numerical words, better is the sentence considered.

viii. *Alphanumeric words in the sentence*: The proportion of the alphanumeric words in the total words is calculated and accordingly the score is assigned to the sentence. More the alphanumeric words, better is the sentence considered.

ix. *Cue-phrases in the sentence*: Generally, if s contains the phrases like 'significantly', 'the most important', 'the paper describes', 'in conclusion', 'the best', 'important', 'hardly' etc., then s is consider as important because it can be a good source for d where valuable information can find out. We use WordNet[7] to find more such words/phrases and generate a list$L$ of cue-words. If s contains any of the elements from the list L, then s receives a score of one else zero (which is in binary form).

x. *Similarity amongst sentences*: The sentences that contain frequent words (words that occur in most of the sentences) are more important as they provide key information about a document. Thus, based on the occurrence of words in other sentences, a score (fss) is assigned to each sentence s and is computed using Equation 1 where, sentence occurrence count (C_s)= $\sum_{w \in L} C_w$ and C_w is the number of other sentences in which w occurs.

$$fss = \frac{C_s}{\max\limits_{s \in d} C_s} \tag{1}$$

xi. *Similarity amongst paragraphs*: It is same as the word similarity among sentences, but instead of individual sentences, the whole paragraph is used to extract this feature. Thus, each sentence in a paragraph will obtain the same score (fsp) and computed using Equation 2 where, paragraph occurrence count (C_p) = $\sum_{w \in L} C_w$ and C_w is number of other paragraphs in which w occurs.

$$fsp = \frac{C_p}{\max\limits_{p \in d} C_p} \tag{2}$$

xii. *LSI-based score*: Compute the TF-IDF vector of each sentence as discussed in feature 2 (Weight of the sentence). Perform LSI[8] on the term-document matrix (without reduction) to generate the term-concept (U) matrix. Let, $s_{lsi\ score}$ = 0. For every $w \in s$, the LSI-based score ($s_{lsiscore}$) is computed using the Equation 3.

$$s_{lsiscore} += row_{sum} \tag{3}$$

where row_{sum} = sum of rows corresponding to word w in the U matrix.

xiii. *Concept-based score*: Using mutual information (MI) and windowing process, a concept is generated from the input document d. The windowing process take place by moving a virtual window of size l from left to right of d till it reaches the end of d and it divides d into windows of l sentences. Equation 4 shows

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

[8] https://nlp.stanford.edu/IR-book/html/htmledition/latent-semantic-indexing-1.html

how the mutual information score is calculated between number of words in the sentence (w_i) and number of words in the document (w_j).

$$MI(w_i, w_j) = log_2 \frac{prob(w_i, w_j)}{prob(w_i)Prob(w_j)} \tag{4}$$

where,

$$prob(w_i) = \frac{\text{no. of windows containing } w_i}{\text{total no. of windows}}$$

$$prob(w_i, w_j) = \frac{\text{joint prob of } w_i \text{ and } w_j}{\text{total no. of windows}}$$

The concept-based score of a sentence ($s_{concept\ score}$) is computed using Equation 5.

$$s_{concept\ score} = \sum_{w_i \in s, w_j \in d} MI(w_i, w_j) \tag{5}$$

xiv. *Doc2Vec*: Doc2vec embeddings [23] is one of the latest feature that is used to determine the importance sentences in a document. Doc2vec is an unsupervised learning algorithm to generate vectors for documents that are an alternative to the bag of words vector [24].
   - Determine the average length of a sentence (after removal of the stop-words) and let this number be r.
   - Choose r-most frequent words (excluding stop words) as a query vector. Train a doc2vec model on all sentences in the corpus and extract Doc2vec embedding features for each sentence and the query vector using trained model. For each sentence s, perform a cosine-similarity between the Doc2vec embedding of s and the query. Use this cosine-similarity weight as the score for each sentence.

xv. *Word2Vec*: *Word2vec* is used to make word embeddings by grouping similar words together in vector space without human intervention. It captures the linguistic contexts of the words using two-layer neural network architecture. Its input is a text corpus and produces a vector space, i.e., feature vectors for words in the corpus of documents. *Word2vec* generates the exact meaning of a word based on the usage and content of the dataset assigned to it. These guesses are very useful to us as it can be used to establish associations between words (e.g, 'woman' is to 'queen', 'man' is to 'king') or cluster the documents and classify them. The vectors formed by *Word2vec* are called *neural word embeddings*. *Word2vec* makes word embeddings in two ways, namely *skip-gram* and *continuous bag of words (CBOW)* models [25]. The required steps to select important sentences using Word2vec (CBOW model) are shown below.
   - Determine the average length of a sentence (after removal of the stop-words). Let this number be q.
   - Select q-most frequent words (excluding stop words) as a query vector. Train a word2vec model on all sentences in the corpus and extract Word2vec embedding features for each sentence and the query vector using trained model. For each of the sentence s, perform a cosine-similarity between the Word2vec embedding of s and the query which will generate the score for each s.

3. *Reducing 2n/X sentences to n/X sentences*:
After performing the topic modeling with the appropriate number of topics *k*, and selecting $\frac{2n}{Xk}$ sentences from each topic, we are left with a total of $2n/X$ sentences. The mapping of each topic to sentences is shown below:

$$Topic_1 \rightarrow \text{(set of } 2n/Xk \text{ sentences)}$$
$$Topic_2 \rightarrow \text{(set of } 2n/Xk \text{ sentences)}$$
$$\vdots$$
$$Topic_k \rightarrow \text{(set of } 2n/Xk \text{ sentences)}$$

For each topic, the following steps are used:
   i. All fifteen word and sentence level features for each sentence (as discussed in Step 2) are extracted in the set of $2n/Xk$ sentences.
   ii. An aggregate score for each sentence is calculated using Equation 6.

$$\frac{\text{sum of all normalized feature scores}}{\text{total number of features}} \tag{6}$$

   iii. All the sentences are ranked based on their aggregate scores and the top n/Xk sentences are selected among them.
   iv. Steps (i-iii) are carried out for all k topics, hence we now have a total of n/X sentences. However, since some of the sentences might be chosen for more than one time as discussed in Step 1, therefore we will always have less than or equal to n/X sentences. The mapping between topics and sentences are now reduce as shown below:

$$Topic_1 \rightarrow \text{(set of } n/Xk \text{ sentences)}$$
$$Topic_2 \rightarrow \text{(set of } n/Xk \text{ sentences)}$$
$$\vdots$$
$$Topic_k \rightarrow \text{(set of } n/Xk \text{ sentences)}$$

4. *Ordering of the sentences*:
Now we have a list of n/X sentences, but these sentences have not yet been ordered. To ensure that the generated summaries are coherent, the following steps are followed:
   i. Since the corpus C contains a total of i documents, hence we have a total of i opening lines (opening line indicates the first sentence of a document), i.e., one line for each document.
   ii. After traversing through the list of n/X sentences, if we encounter an opening line OL during this traversal, then OL will be chosen as the summary's first sentence. Unlikely, when none of the n/X sentences contain any opening line then a sentence is selected randomly from this list of n/X sentences as the opening line.
   iii. Next, a sentence-sentence similarity matrix is generated using WordNet based similarity as shown in Table 3. We created a list which will store the sentences in order. The first entry in this list is the starting sentence which is identified as described in Step ii. We then go to the r^{th} row of the sentence-sentence

similarity matrix where r is the starting index. We traverse this row and find the maximum element in the row apart from the diagonal elements. Consider that this element is found in the j^{th} column. We now add j to our list of ordered sentences and then jump to the j^{th} row of the matrix. Then the element having the maximum score from the j^{th} row excluding the previous selected sentence and the j^{th} column is selected.

iv. In this way, we keep selecting sentences that are closest in similarity to the previously chosen sentence. This will ensure that the summary sentences will be coherent. Table 3 shows the matrix where $m = n/X$ and s_{ij} denotes the similarity between sentence i and j. Implementation details are shown in Algorithm 3.

Table 3: Sentence-Sentence matrix

	s_1	s_2	s_3	...	s_m
s_1	s_{11}	s_{12}	s_{13}	...	s_{1m}
s_2	s_{21}	s_{22}	s_{23}	...	s_{2m}
s_3	s_{31}	s_{32}	s_{33}	...	s_{3m}
.	.	.	.	...	.
.	.	.	.	...	.
.	.	.	.	...	.
s_m	s_{m1}	s_{m2}	s_{m3}	...	s_{mm}

Algorithm 3: Sentence ordering

```
Input: unordered extracted sentences of the summary
Output: order_list
num_sets ← number of unordered sentences
sim_matrix[num_sents][num_sents]← WordNet based similarity matrix
order_list[num_sents] ← φ
sent ← starting sentence index
for all i ∈ (0, num_sents -1) do
    append sent into the order list
    sent ← index of sentence having maximum similarity with sent
end for
return order_list
```

2.1 Generating Extractive gold summaries from Human written summaries

Python Natural Language Toolkit[9] has been used to tokenize the documents into sentences. The way extractive gold summaries are created from the human written gold summaries (available in DUC datasets) are explained below and generalized in Algorithm 4.

i) Parse each document d of C sentence by sentence. For each sentence $s \in d$, calculate the number of keywords it has in common with each of the four human written gold summaries of the corpus C. The number of common words between each s and four human written gold summaries gives the score for the sentence s. This way the scores for all the sentences of a document are calculated.

[9] http://www.nltk.org/

ii) Now rank all the sentences of a document d using the scores received by them in Step i). Top-m important sentences are selected from the ranked sentences in order to form the extractive gold summary of d. For experimental purpose, we have taken m = 5, i.e., each document has an extractive gold summary of five sentences. Repeat steps i) and ii) for all documents of the corpus C.

Algorithm 4: Extractive gold summary generation

Input: four human written gold summaries of the corpus C
Output: extractive gold summaries $s_{extract-summary}[d]$
$hws[4][] \leftarrow$ four human written gold summaries of DUC dataset
$s_{count} \leftarrow \phi$
$s_{extractive}[d][s] \leftarrow \phi$
$s_{extract-summary}[d] \leftarrow \phi$ // stores the extractive gold summaries for training the classifiers

for all *hws*[4][] **do**
    remove the stop-words
    $L[] \leftarrow$ all keywords of *hws*[4][]
end for
for all $d \in C$ **do**
    for all $s \in d$ **do**
        for all word $w_1 \in s$ **do**
            for all word $w_2 \in L$ **do**
                if $w_1 = w_2$ **then**
                $s_{count} = s_{count} + 1$
                end if
            end for
        end for
        $s_{extractive}[d][s] \leftarrow s_{count}$
        $s_{count} \leftarrow \phi$
    end for
end for
for all $d \in s_{extractive}[d][s]$ **do**
    rank s according their s_{count} values
    //rank all sentences of d
    $s_{extract-summary}[d] \leftarrow$ top-k sentences from d
    //select top k sentences which represent the extractive gold summaries of d
end for

3 Experimental Analysis

For experimental purpose, Document Understanding Conference (DUC)[10] datasets are used, where each dataset having four human written summaries. DUC-2001 dataset has 30 document sets and each set has an average of 10 documents having a total of 296 documents. Similarly, DUC-2005 has 50 document sets and each set having 25-50 documents constitutes a total of 1503 documents in all. DUC-2007 contains 45 document sets each containing 25 documents and a total of 1125 documents in all.

[10] http://www.duc.nist.gov

3.1 ROUGE-N scores evaluation

ROUGE or Recall-Oriented Understudy for Gisting Evaluation score [26] is generally used to measure the performance of text summarization. ROUGE-N measures the unigram, bigram, trigram, and higher order n-gram overlap between the summaries generated by the system (i.e., by the proposed approach) and the gold summary (either human written or extractive). ROUGE-1 and ROUGE-2 are used to compute the F-measure value for unigram and bigram matching respectively. ROUGE toolkit takes as input both system generated summary *sgs* and gold summary *gs* and evaluates the different scores based on the n-gram matching. The recall and precision of extractive and human written summaries are computed using Equations 7 and 8 respectively.

$$recall = \frac{\sum_{s \in gs} \sum_{gram_n \in s} Count_{match}(gram_n)}{\sum_{s \in gs} \sum_{gram_n \in s} Count(gram_n)} \tag{7}$$

where n stands for the length of the n-gram i.e., $gram_n$, $Count_{match}(gram_n)$ is the maximum number of n-grams overlap between *sgs* and the *gs*, and $Count(gram_n)$ is the number of n-grams in the gs. Equation 7 shows that ROUGE-N relies on n-gram recall because the denominator of the equation denotes the total sum of the number of n-grams present in the human generated gold summary. It can also be noted that the denominator of the equation 7 can be increased when one adds more summaries because there might exist more human generated gold summaries. The numerator of the Equation 7 finds the sum over the human written gold summaries. This eventually gives more importance to those n-grams which are common to both the system and human written gold summaries. Hence, the system generated summaries that contain words shared by more sentences in human written gold summary will be preferred by ROUGE-N measure.

$$precision = \frac{\sum_{s \in gs} \sum_{gram_n \in s} Count_{match}(gram_n)}{\sum_{s \in sgs} \sum_{gram_n \in s} Count(gram_n)} \tag{8}$$

All the symbols have the same meaning as in equation 7. Here the denominator denotes the total sum of the number of n-grams present in the system generated summary.

3.2 Discussion

For experimental purpose, the stop-words are removed from all DUC-datasets while computing the ROUGE score. 'n' sentences are extracted from all the documents of each DUC dataset and are kept in a corpus. LDA is run on each sentence of the corpus to get optimum number of k independent topics as shown in Figures 1-3 for different DUC datasets. After getting k independent topics, for each topic, a set of n/X important sentences are identified based on word and sentence level features which are part of the final summary where X is a user defined value and greater than 2. Finally, all these n/X sentences are ranked based on the sentence-sentence similarity scores. LDA is run on each sentence of the corpus to get optimum number of k independent topics as shown in Figures 1-3 for different DUC datasets. Tables 4 - 9 show the performance of ROUGE-1 score of extractive and human written gold summary on different DUC datasets. Similarly, Tables 10 - 15 show the performance of ROUGE-2 score of

extractive and human written gold summary on different DUC datasets. For demonstration purpose, the aggregate scores of six example sentences of document 'D0720E' of DUC-2007 dataset are shown in Table 16. From the results of these tables, the following points can be observed

i. The ROUGE-N scores of human written summaries are less than the extractive summaries and this is because as in the original document (i.e., the document of the DUC dataset), people try to use different words and paraphrase the documents in their own way, hence most of the words of the human written summaries do not match with the original documents. However, the extractive gold summaries takes care of the words common between the sentence of the original document and the four human gold summaries as discussed in Section 2.1.
ii. If the stop-words are removed from the corpus, then for every concise summary, ROUGE-1 score alone may suffice [26] which are reflected when we compared the obtained ROUGE-1 with the ROUGE-2 scores.

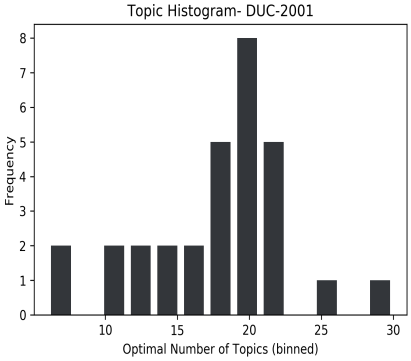

Fig. 1: Optimal number of topics on DUC-2001

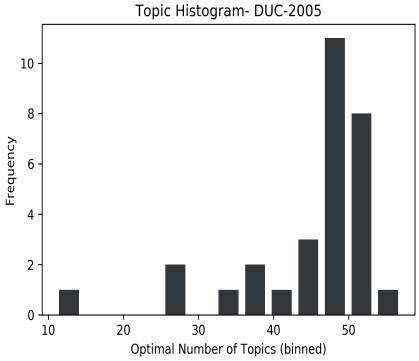

Fig. 2: Optimal number of topics on DUC-2005

4 Conclusion

The paper developed an approach that generates a concise summary using multi-document text summarization. Initially, the topic modeling using LDA is run on each single sentence of a corpus and a subset of sentences are selected per topic. Next, fifteen important word and sentence level features are identified and using these features, the aggregate score of each sentence is calculated. Some important sentences are selected based on the aggregate scores and finally those important sentences are ordered to form a coherent summary. Unlike most of the other techniques, we can assured that the summaries generated by the proposed approach will not only contain sentences that have the best structural features, but also cover all the key topics of the corpus. Future work includes further improvement of the two heuristic methods i.e., identifying a suitable number of topics and reordering of the sentences. This work can also be extended by using abstractive text summarization technique.

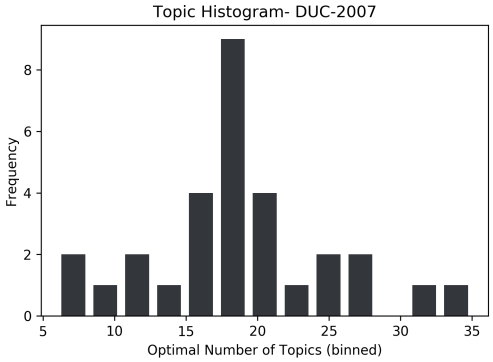

Fig. 3: Optimal number of topics on DUC-2007

Table 4: ROUGE-1 Extractive (DUC-2001)

Doc-Id	Recall	Precision	F-Score
D01A	0.67470	0.16120	0.26022
D02A	0.77724	0.10197	0.18029
D03A	0.59951	0.29545	0.39583
D07B	0.50481	0.34426	0.40936
D09B	0.74463	0.08446	0.15171
D10B	0.53012	0.25611	0.34537
D16C	0.49412	0.39179	0.43704
D17C	0.54726	0.25287	0.34591
D18C	0.61071	0.32138	0.42114
D20D	0.55847	0.34412	0.42584
D21D	0.38517	0.44722	0.41388
D23D	0.56782	0.34306	0.42771
D25E	0.59268	0.15831	0.24987
D26E	0.42410	0.32001	0.36477
D29E	0.57518	0.33241	0.42133
Average	0.57243	0.27697	0.35002

Table 5: ROUGE-1 Human written (DUC-2001)

Doc-Id	Recall	Precision	F-Score
D01A	0.44363	0.20841	0.28359
D02A	0.67545	0.23602	0.34981
D03A	0.49339	0.26794	0.34729
D07B	0.44161	0.19836	0.27376
D09B	0.66033	0.23525	0.34691
D10B	0.44467	0.25728	0.32596
D16C	0.46725	0.19963	0.27974
D17C	0.47535	0.15517	0.23397
D18C	0.59443	0.24584	0.34783
D20D	0.40270	0.21912	0.28381
D21D	0.31047	0.23889	0.27002
D23D	0.32176	0.19306	0.24132
D25E	0.67138	0.27818	0.39337
D26E	0.54741	0.23091	0.32481
D29E	0.50806	0.26069	0.34458
Average	0.49719	0.22831	0.30978

Table 6: ROUGE-1 Extractive (DUC-2005)

Doc-Id	Recall	Precision	F-Score
D301	0.43273	0.74537	0.54757
D307	0.44990	0.77817	0.57016
D311	0.37160	0.86547	0.51995
D313	0.47493	0.79914	0.59578
D321	0.55001	0.75214	0.63538
D324	0.20625	0.89593	0.33531
D331	0.71323	0.72054	0.71687
D332	0.74888	0.68653	0.71635
D343	0.40890	0.85114	0.55242
D345	0.33215	0.86331	0.47973
D346	0.47539	0.74505	0.58043
D347	0.78038	0.71554	0.74656
D350	0.45230	0.75985	0.56706
D354	0.47950	0.78655	0.59579
D391	0.68774	0.68914	0.68844
Average	0.50425	0.77692	0.58985

Table 7: ROUGE-1 Human written (DUC-2005)

Doc-Id	Recall	Precision	F-Score
D301	0.46291	0.22785	0.30538
D307	0.51105	0.26882	0.35232
D311	0.47226	0.60594	0.53081
D313	0.68563	0.32934	0.44495
D321	0.59350	0.22408	0.32533
D324	0.19080	0.64404	0.29438
D331	0.76016	0.17862	0.28927
D332	0.58571	0.13851	0.22404
D343	0.54839	0.33601	0.41670
D345	0.35294	0.57365	0.43701
D346	0.51151	0.27370	0.35659
D347	0.61161	0.13117	0.21602
D350	0.54472	0.28926	0.37786
D354	0.52239	0.22173	0.31132
D391	0.57046	0.32261	0.41214
Average	0.52826	0.31768	0.35294

Table 8: ROUGE-1 Extractive (DUC-2007)

Doc-Id	Recall	Precision	F-Score
D0701A	0.54533	0.11414	0.18877
D0702A	0.58369	0.20606	0.30459
D0703A	0.51667	0.23308	0.32124
D0705A	0.55703	0.21627	0.31157
D0706B	0.49161	0.17402	0.25705
D0707B	0.30196	0.24290	0.26923
D0708B	0.51429	0.25862	0.34417
D0709B	0.79960	0.15011	0.25277
D0710C	0.38978	0.29897	0.33839
D0711C	0.67520	0.29781	0.41332
D0712C	0.46970	0.26007	0.33477
D0713C	0.47486	0.20706	0.28838
D0714D	0.44698	0.24968	0.32039
D0715D	0.59544	0.27756	0.37862
D0716D	0.54277	0.22412	0.31724
Average	0.52699	0.22736	0.30936

Table 9: ROUGE-1 Human written (DUC-2007)

Doc-Id	Recall	Precision	F-Score
D0701A	0.73705	0.05485	0.10210
D0702A	0.67704	0.08788	0.15557
D0703A	0.44531	0.28571	0.34809
D0705A	0.71311	0.17920	0.28642
D0706B	0.61508	0.13158	0.21678
D0707B	0.48606	0.38486	0.42958
D0708B	0.49407	0.35920	0.41597
D0709B	0.76384	0.07788	0.14135
D0710C	0.60079	0.31340	0.41192
D0711C	0.76378	0.13691	0.2322
D0712C	0.63878	0.28188	0.39115
D0713C	0.62451	0.19245	0.29423
D0714D	0.67969	0.11083	0.19058
D0715D	0.58120	0.18061	0.27558
D0716D	0.61089	0.19123	0.29128
Average	0.62874	0.19789	0.27885

Table 10: ROUGE-2 Extractive (DUC-2001)

Doc Id	Recall	Precision	F-Score
D01A	0.12270	0.05760	0.07840
D02A	0.27298	0.09533	0.14131
D03A	0.16556	0.08982	0.11646
D07B	0.13553	0.06076	0.08390
D09B	0.20380	0.07257	0.10703
D10B	0.17137	0.09907	0.12555
D16C	0.14474	0.06168	0.08650
D17C	0.13428	0.04373	0.06597
D18C	0.21429	0.08846	0.12523
D20D	0.14363	0.07806	0.10115
D21D	0.09420	0.07242	0.08189
D23D	0.10441	0.06259	0.07826
D25E	0.26299	0.10887	0.15399
D26E	0.22944	0.09654	0.13590
D29E	0.17790	0.09116	0.12055
Average	0.17185	0.07857	0.10681

Table 11: ROUGE-2 Human written (DUC-2001)

Doc-Id	Recall	Precision	F-Score
D01A	0.15676	0.03341	0.05508
D02A	0.27224	0.03209	0.05742
D03A	0.17473	0.07784	0.10771
D07B	0.08356	0.05090	0.06327
D09B	0.26738	0.02708	0.04918
D10B	0.14054	0.06061	0.08469
D16C	0.12533	0.08972	0.10458
D17C	0.13333	0.05524	0.07811
D18C	0.22554	0.10641	0.14460
D20D	0.17439	0.09426	0.12237
D21D	0.07219	0.07521	0.07367
D23D	0.14139	0.07650	0.09928
D25E	0.18733	0.04433	0.07169
D26E	0.06648	0.04372	0.05275
D29E	0.16757	0.08564	0.11335
Average	0.15925	0.06353	0.08518

References

1. G. A. Miller, "Wordnet: a lexical database for english," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.
2. L. Suanmali, N. Salim, and M. S. Binwahlan, "Feature-based sentence extraction using fuzzy inference rules," in *2009 International Conference on Signal Processing Systems*. IEEE, 2009, pp. 511–515.
3. H. P. Luhn, "The automatic creation of literature abstracts," *IBM Journal of research and development*, vol. 2, no. 2, pp. 159–165, 1958.
4. K. Ganesan, C. Zhai, and J. Han, "Opinosis: a graph-based approach to abstractive summarization of highly redundant opinions," in *Proceedings of the 23rd International Conference on Computational Linguistics*. Association for Computational Linguistics, 2010, pp. 340–348.
5. R. Nallapati, B. Zhou, C. dos Santos, Ç. glar Gulçehre, and B. Xiang, "Abstractive text summarization using sequence-to-sequence rnns and beyond," *CoNLL 2016*, p. 280, 2016.

Table 12: ROUGE-2 Extractive (DUC-2005)

Doc-Id	Recall	Precision	F-Score
D301	0.26548	0.45738	0.33596
D307	0.30684	0.53084	0.38889
D311	0.26914	0.62703	0.37662
D313	0.34112	0.57410	0.42796
D321	0.37891	0.51822	0.43775
D324	0.17576	0.76435	0.28579
D331	0.46161	0.46634	0.46396
D332	0.50026	0.45860	0.47853
D343	0.31701	0.66007	0.42831
D345	0.26956	0.70092	0.38937
D346	0.29128	0.45659	0.35567
D347	0.57447	0.52673	0.54957
D350	0.29049	0.48812	0.36422
D354	0.32333	0.53047	0.40177
D391	0.41050	0.41134	0.41092
Average	0.34505	0.54474	0.40635

Table 13: ROUGE-2 Human written (DUC-2005)

Doc-Id	Recall	Precision	F-Score
D301	0.11554	0.05650	0.07589
D307	0.12379	0.06471	0.08499
D311	0.13556	0.17274	0.15191
D313	0.22111	0.10552	0.14286
D321	0.17939	0.06729	0.09786
D324	0.07609	0.25529	0.11724
D331	0.30040	0.07011	0.11369
D332	0.16444	0.03862	0.06255
D343	0.15736	0.09580	0.11909
D345	0.07858	0.12686	0.09705
D346	0.09778	0.05198	0.06788
D347	0.13407	0.02856	0.04709
D350	0.11668	0.06156	0.08059
D354	0.07639	0.03218	0.04528
D391	0.12335	0.06924	0.08869
Average	0.14003	0.08646	0.09284

Table 14: ROUGE-2 Extractive (DUC-2007)

Doc-Id	Recall	Precision	F-Score
D0701A	0.15177	0.03173	0.05249
D0702A	0.17049	0.06013	0.08891
D0703A	0.18436	0.08291	0.11438
D0705A	0.18883	0.0732	0.10550
D0706B	0.20433	0.07222	0.10672
D0707B	0.07480	0.06013	0.06667
D0708B	0.18966	0.09510	0.12668
D0709B	0.36747	0.06887	0.11601
D0710C	0.13208	0.10124	0.11462
D0711C	0.25001	0.11017	0.15294
D0712C	0.18845	0.10420	0.13420
D0713C	0.12605	0.05488	0.07647
D0714D	0.13470	0.07521	0.09652
D0715D	0.23429	0.10904	0.14882
D0716D	0.18343	0.07561	0.10708
Average	0.18538	0.07831	0.10720

Table 15: ROUGE-2 Human written (DUC-2007)

Doc-Id	Recall	Precision	F-Score
D0701A	0.28151	0.01987	0.03712
D0702A	0.16667	0.02072	0.03685
D0703A	0.17501	0.10553	0.13166
D0705A	0.22944	0.05464	0.08826
D0706B	0.16309	0.03229	0.05390
D0707B	0.12971	0.09810	0.11171
D0708B	0.14815	0.10375	0.12203
D0709B	0.22780	0.02221	0.04047
D0710C	0.18930	0.09504	0.12655
D0711C	0.24473	0.04096	0.07018
D0712C	0.22984	0.09580	0.13523
D0713C	0.21992	0.06463	0.09991
D0714D	0.22449	0.03505	0.06064
D0715D	0.21364	0.06250	0.09671
D0716D	0.23770	0.07073	0.10902
Average	0.20539	0.06145	0.08801

Table 16: DUC-2007 (D0720E)

Agg_score	Sentence
0.367	Representatives of the heavy metal band Metallica went Wednesday to the headquarters of Napster Inc. with 13 boxes of papers said to list more than 300,000 computer users who, without the band's permission, had recently downloaded Metallica music from other users listed on Napster's Internet site.
0.355	NEW HAVEN, Conn. (AP) The heavy metal band Metallica on Wednesday dropped legal action against Yale University after the school agreed to block access from their computer network to an Internet site that allows people to trade copyrighted music.
0.352	SAN FRANCISCO_The battle over music in cyberspace spilled onto the streets of San Mateo, Calif., Wednesday as Metallica drummer Lars Ulrich jawed with fans of Napster, the popular and controversial music sharing software.
0.278	Another subplot features Limp Bizkit's singerrapper, Fred Durst, who also is a senior vice president of Interscope Records, whose parent label, Universal Music, joined the RIAA in a separate lawsuit against Napster.
0.164	'I create my music.'
0.015	The blush deepened.

6. N. Moratanch and S. Chitrakala, "A survey on extractive text summarization," in *Computer, Communication and Signal Processing (ICCCSP), 2017 International Conference on*. IEEE, 2017, pp. 1–6.
7. V. Gupta and G. S. Lehal, "A survey of text summarization extractive techniques," *Journal of emerging technologies in web intelligence*, vol. 2, no. 3, pp. 258–268, 2010.
8. M. Mendoza, S. Bonilla, C. Noguera, C. Cobos, and E. León, "Extractive single-document summarization based on genetic operators and guided local search," *Expert Systems with Applications*, vol. 41, no. 9, pp. 4158–4169, 2014.
9. N. Chatterjee and P. K. Sahoo, "Random indexing and modified random indexing based approach for extractive text summarization," *Computer Speech & Language*, vol. 29, no. 1, pp. 32–44, 2015.
10. G. Yang, D. Wen, N.-S. Chen, E. Sutinen *et al.*, "A novel contextual topic model for multi-document summarization," *Expert Systems with Applications*, vol. 42, no. 3, pp. 1340–1352, 2015.
11. Y.-H. Hu, Y.-L. Chen, and H.-L. Chou, "Opinion mining from online hotel reviews–a text summarization approach," *Information Processing & Management*, vol. 53, no. 2, pp. 436–449, 2017.
12. M. Yousefi-Azar and L. Hamey, "Text summarization using unsupervised deep learning," *Expert Systems with Applications*, vol. 68, pp. 93–105, 2017.
13. C. Fang, D. Mu, Z. Deng, and Z. Wu, "Word-sentence co-ranking for automatic extractive text summarization," *Expert Systems with Applications*, vol. 72, pp. 189–195, 2017.
14. R. Nallapati, F. Zhai, and B. Zhou, "Summarunner: A recurrent neural network based sequence model for extractive summarization of documents." in *AAAI*, 2017, pp. 3075–3081.
15. R. K. Roul, J. K. Sahoo, and R. Goel, "Deep learning in the domain of multi-document text summarization," in *International Conference on Pattern Recognition and Machine Intelligence*. Springer, 2017, pp. 575–581.
16. F. Liu, J. Flanigan, S. Thomson, N. Sadeh, and N. A. Smith, "Toward abstractive summarization using semantic representations," *arXiv preprint arXiv:1805.10399*, 2018.
17. S. Narayan, S. B. Cohen, and M. Lapata, "Ranking sentences for extractive summarization with reinforcement learning," *arXiv preprint arXiv:1802.08636*, 2018.
18. J. Zhang, J. Tan, and X. Wan, "Towards a neural network approach to abstractive multi-document summarization," *arXiv preprint arXiv:1804.09010*, 2018.
19. D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.
20. N. Aletras and M. Stevenson, "Measuring the similarity between automatically generated topics," in *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, volume 2: Short Papers*, 2014, pp. 22–27.
21. S. Kullback and R. A. Leibler, "On information and sufficiency," *The annals of mathematical statistics*, vol. 22, no. 1, pp. 79–86, 1951.
22. B. Fuglede and F. Topsoe, "Jensen-shannon divergence and hilbert space embedding," in *Information Theory, 2004. ISIT 2004. Proceedings. International Symposium on*. IEEE, 2004, p. 31.
23. J. H. Lau and T. Baldwin, "An empirical evaluation of doc2vec with practical insights into document embedding generation," *ACL 2016*, pp. 78–86, 2016.
24. O. Levy, Y. Goldberg, and I. Dagan, "Improving distributional similarity with lessons learned from word embeddings," *Transactions of the Association for Computational Linguistics*, vol. 3, pp. 211–225, 2015.
25. T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.
26. C.-Y. Lin, "Rouge: A package for automatic evaluation of summaries," in *Text summarization branches out: Proceedings of the ACL-04 workshop*, vol. 8, pp. 74–81.