# Obtaining Single Document Summaries Using Latent Dirichlet Allocation

**Conference Paper** · November 2012

DOI: 10.1007/978-3-642-34478-7_9



**2 authors:**

M. Narasimha Murty
Indian Institute of Science
**266** PUBLICATIONS **17,571** CITATIONS
SEE PROFILE

Karthik Nagesh
Indian Institute of Science
**1** PUBLICATION **0** CITATIONS
SEE PROFILE

# Obtaining Single Document Summaries Using Latent Dirichlet Allocation

Karthik N and M Narasimha Murty

Dept of Computer Science and automation, Indian Institute of Science,
Bangalore-560012, India
{karthik.n,mnm}@csa.iisc.ernet.in

**Abstract.** In this paper, we present a novel approach that makes use of topic models based on Latent Dirichlet allocation(LDA) for generating single document summaries. Our approach is distinguished from other LDA based approaches in that we identify the summary topics which best describe a given document and only extract sentences from those paragraphs within the document which are highly correlated given the summary topics. This ensures that our summaries always highlight the crux of the document without paying any attention to the grammar and the structure of the documents. Finally, we evaluate our summaries on the DUC 2002 Single document summarization data corpus using ROUGE measures. Our summaries had higher ROUGE values and better semantic similarity with the documents than the DUC summaries.

**Keywords:** Single Document Summaries, Latent Dirichlet Allocation, SVM, Naïve Bayes Classifier

## 1 Introduction

Generating summaries for large document corpora is an arduous task as it requires specialized human effort. The preciseness of a summary is limited to the understanding of a document or the document domain by the human summarizer. This has led to a large amount of research in the field of automatic text summarization to generate precise and concise summaries.

Text summarization can be classified into abstractive summarization and extractive summarization. Abstractive summarization involves understanding and identifying the main concepts of the document and generating the summary by using well formed sentences with this knowledge. Construction of such a system relies on the machine's ability to "understand" the natural language of the document with all its ambiguities.

An extractive summarization method consists of identifying important sentences or paragraphs in a document and concatenating them to form a meaningful summary. Extractive summarization techniques exploit statistical, linguistic and positional properties of sentences or paragraphs for ranking the same. In this paper we present a new extractive summarization technique on LDA(Latent Dirichlet allocation) to identify the summary sentences for generating single document summaries.

## 2 Related Work

The earliest known works on extractive summarization constructed summaries by identifying important sentences in the original document. Properties such as presence of high frequency words [4], location of sentences [3], etc were used to identify these important sentences. However such methods did not capture the semantics of the document. Thus, a different approach [5] using HMM's which considered the local dependencies of the extracted sentences tried to address this problem.

Our approach makes use of Latent Dirichlet Allocation(LDA) for extracting sentences which reflect the core "theme" of the document. According to Blei [1], the LDA topics are synonymous to the themes of the document. Most of the works on LDA based summarization are with regard to the multi-document summarization problem. To the best of our knowledge, only a few significant methods are available which deal with using LDA for single document summarization. Chang et al. [2] have proposed a new model for extractive summarization called SLDA (Sentence-based LDA) designed for query based summarization. This model assumes that each sentence in a document is influenced by a single theme or topic and ranking of sentences is according to their query likelihood calculated by the SLDA parameters.

Our approach exploits the class specific properties of the document namely the class specific topics. This is because a given document can be summarized in different ways based on the document class. We first do pre-processing on the data to select appropriate features for every class in the corpus, such that the vocabulary words of each class have least entropy in the class. Then we generate a document word matrix separately for every class and provide it as input to LDA for generating class-wise topics. These class-wise topics are post processed as explained in later sections and the summary topics for each document are obtained. Using these summary topics, the paragraphs are clustered and sentences extracted from such clusters are ranked based upon the importance of the paragraph of their origination, the importance of that cluster within the summary topic and the summary topic to which the cluster belongs to. Thus our summaries capture the central theme of the document.

## 3 Processing Data and LDA Topics for Summarization

The summary topics of a document D are the ones that highlight the main content of the document. For our summaries to effectively capture the semantics of the document, non-summary topics should have negligible probability within the documents. We choose appropriate asymmetric Dirichlet priors [7] for LDA such that only a few topics have high probabilities in the document-topic distribution.

### 3.1 Pre-processing Data

**Feature Selection using Naïve Bayes and SVM** Firstly, the stop words are removed by using the Zipf's law. Let this set of words be called $V_{wsw}$. Next

vocabulary words with high Mutual information[8] "I(U;C)" scores for a given class C are selected(where U depicts the presence or absence of the word in C). This ensures that our vocabulary words have least entropy for a given class. Next we employ Naïve Bayes and SVM methods for feature selection. The features generated by this step generate two different summaries by our algorithm. The Naïve Bayes method involves training a classifier on two classes at a time using features selected from previous steps and then, only selecting the high probability features within each class. In the SVM method, we learn the "w" vector for two classes at a time and then segregate features into positive class and negative class features based upon their sign in the "w" vector. We then rank the features based on their absolute weights and select the high ranking features for every class.

### 3.2 Post-Processing LDA Topics

Within a given topic $t_k$, the words are ranked on the basis of decreasing order of their probabilities in $t_k$ and the top 70% of the words are retained. Let this set of words in each topic $t_k$ be called $HPt_k$. Each word in $HPt_k$ is considered as a vector of its topic distributions. The words within each $HPt_k$ are clustered using this vector representation to generate semantically similar clusters. The set $HPt_k$ is replaced by the cluster representatives instead of the words where the probability of each cluster is the sum of the probabilities of its components. This proves to be useful in reducing the dimensionality of the topics by consolidating the probabilities of semantically similar words into clusters. Now, semantically similar words can be assigned the same probability(their cluster probability), for a given topic.

## 4 Summarization Algorithm

"A paragraph is a distinct section of a piece of writing, usually dealing with a single theme or topic."[10] Our approach is based on this fundamental definition of a paragraph. The semantics of a sentence is dependent on its local environment or nearby sentences, its meaning is ambiguous without context. Thus our approach differs from [2] by considering the paragraph as an entire document which defines a topic, whereas SLDA [2] considers the sentence as a single document defining a topic.

The steps of our algorithm are explained (in the same order) in detail in the following subsections.

### 4.1 Building Paragraph Similarity Matrix for Every Topic

The words in the set $V_{wsw}$ are arranged in the decreasing order of their frequencies in the corpus and partitioned into blocks of size L. Thus let vector G=$\{$ $V_{wsw}$ partitions of size L$\}$. We now define a vector W(G) which contains weights $\forall g \in G$ such that:

$$W(g) = \frac{1}{\sum_{words \in g} termfrequency} \tag{1}$$

**Algorithm 1** Get Similarity matrix

**Require:** Paras $\in D$, $t_k, W$
**Ensure:** Similarity matrix $\Rightarrow SMt_k$

**for** each $para_i \in$ Paras **do**
  Initialize $P_i$ vector of length($V_{t_k}$) to 0
  **for** x = 1 to length($V_{t_k}$) **do**
    $g' \leftarrow V_{t_k}$[x]
    **for** each w $\in$ words in $para_i$ **do**
      **if** w $\in g'$ **then**
        $P_i$[x]=W[x]
        break {Even if one word of block $g'$ is present in $P_i$, then that corresponding x entry is made non zero.}
      **end if**
    **end for**
  **end for**
**end for**{The vector $P_i$ is a modified representation of a paragraph such that it contains the W(g) of the corresponding g if even one of the words of g' are in $P_i$ }
**for** $i$=1 to length(P) **do**
  **for** $j$=1 to $j \leq i$ **do**
    $SMt_k[i][j]$=cosine-similarity($P[i]$,$P[j]$) {where, $P[i]$ and $P[j]$ are vectors}
  **end for**
**end for**{Two paragraphs are considered exactly same if they have atleast one word in $g'$, $\forall g' \in V_{t_k}$}

Now given a topic $t_k$, we construct a vector $V_{t_k}$ of same dimensions as G such that $V_{t_k}$={g'|g'=g$\bigcap HPt_k$,$\forall g \in G$}. The weight vector for $V_{t_k}$ is same as W(g). We make use of $V_{t_k}$ in Algorithm 1 which gives the paragraph similarity matrix $SMt_k$ for topic $t_k$. Paras is the set of all paragraphs in the document $D$, $t_k$ is the topic under consideration and W(G) is the weight vector.

### 4.2 Finding Summary Topics

For each topic $t_k$ in set of all topics, we sum the entries in the Similarity matrix $SMt_k$ to get the values $TW$[k] of a vector TW. We rank the topics based on the decreasing order of their $TW$[k] values and the top 70% of the topics are chosen for each document. These constitute the summary topics for document $D$.

### 4.3 Paragraph Clustering

Let the set of summary topics of document D be $SummaryTopics_D$. For every topic $t_i \in SummaryTopics_D$ we find the paragraph clusters by applying the Single-Link clustering algorithm by considering the entries in the corresponding $SMt_i$ as the distances between paragraphs $P_j$ and $P_k$. The clusters of topic $t_i$ are stored in the array $Clusterst_i$. The corresponding weights of the clusters in $Clusterst_i$ are stored in the array $CWt_i$. The cluster weights are got by adding the corresponding $SMt_i$[j][k] value into a cluster when a paragraph $P_j$ and $P_k$

are added into the same cluster. Each entry in the array $CWt_i$ is a measure of the closeness of the paragraphs within that cluster entry.

### 4.4 Sentence Ranking

For every $t_i \in SummaryTopics_D$ of document D, we can find the weight of the sentence $S_i$ by using (2)

$$P(S_i|t_j) = \sum_{Cl \in Clusterst_j} \sum_{\{Pa \in Cl\}} \prod_{\{w_{i_p} \in Pa \& w_{i_p} \in HPt_j\}} P(w_{i_p}|Pa)P(Pa|Cl)P(Cl|t_j)P(t_j), \tag{2}$$

where $w_{i_p}$ is the $p$th word of the sentence $S_i$. Equation (2) is a direct result of the application of the Bayes theorem. The value of the entities P($w_i$| Pa), P(Pa | Cl) and P(Cl |$t_j$) are direct. The value of P($t_j$) is got by normalizing the array $TW$ calculated in Sect.4.2. The entry $TW[j]$ is P($t_j$). All the candidate sentences are arranged in the decreasing order of their weights and the top sentences are extracted until we encounter our word limit. However one disadvantage of (2) is that long sentences are penalized due to the multiplication of the probabilities. So on the same lines as Arora et al [6], we also replace the multiplication of P($w_i$| Para) with summation. Now in order to ensure that length of the sentence does not play a significant role in determining the weight of the sentence, (2) is modified as (3).

$$P(S_i|t_j) = \frac{\sum_{Cl \in Clusterst_j} \sum_{\{Pa \in Cl\}} \sum_{\{w_{i_p} \in Pa \& w_{i_p} \in HPt_j\}} P(w_{i_p}|Pa)P(Pa|Cl)P(Cl|t_j)P(t_j)}{length(S_i)} \tag{3}$$

## 5 Evaluation and Results

We evaluated our algorithm on the DUC 2002 single document summarization corpus. There were a total of 60 sets of documents with 10 documents in each set. The corpus comprises of four classes distributed across the 60 sets.The Single Document Summarization task requires the generating of a summary not exceeding 100 words for every given document. To evaluate the automatic summaries, two human generated summaries are provided in the DUC corpus for every single document. We adopted the ROUGE-N [9] measure for evaluation. Specifically our summaries were evaluated using the ROUGE-1,2,3 and 4 measures. Given a document D, we shall refer to the summaries generated for D using our algorithm as **Auto_Summ(D)** and the DUC provided summaries as **DUC_Summ$_x$(D)**, where x $\in \{1, 2\}$ refers to either of the human summaries provided.

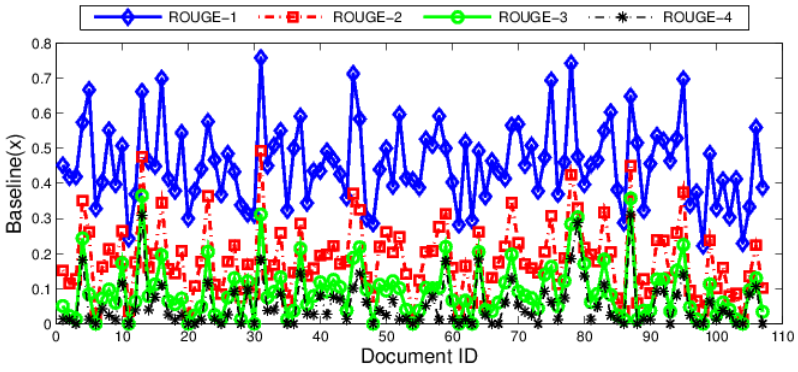


**Fig. 1.** Comparison of Baseline(D) values for ROUGE-1,2,3,4 values

For every document D, we fix the **baseline(D)** as maximum ROUGE value got by evaluating $DUC\_Summ_1(D)$ and $DUC\_Summ_2(D)$ against each other. This follows from the assumption that the baseline is the maximum ROGUE value that the human generated summaries can attain over one another. Figure 1 shows the comparison between the baseline ROUGE-1,2,3,4 values for documents of a single class. It was observed that the ROUGE-1 values are the highest and outscore the ROUGE-2 by a factor of 2. The corresponding graph for all four classes is more or less the same. This clearly shows that the DUC summaries are more word oriented than phrase oriented. Thus for our evaluation we only consider the ROUGE-1 values.

Our algorithm was run on documents pertaining to a single class at a time. The Auto_Summ(D) of every document is analysed using the ROGUE-1 measure against the two DUC_Summ(D) that are provided. The maximum ROUGE-1 value that is found for Auto_Summ(D) of the document D against the two DUC_Summ(D) is taken as the **ROUGE_score(Auto_Summ(D))** as in [9].

We define $\triangle$ROUGE_Gain(D)=ROUGE_score(Auto_Summ(D)) - baseline(D). If $\triangle$ROUGE_Gain(D) $\geq 0$, then we classify the corresponding Auto_Summ(D) as **Better summary** than the two DUC_Summ(D). Else if $\triangle$ROUGE_Gain(D) $< 0$, we classify such summaries as **Worse summaries**. Table 1 shows the statistics with respect to the evaluation of our Auto_Summ(D) for all classes of documents in the DUC data corpus. The column "Method" refers to the method used in feature selection. The results of the corpus clearly show that Auto_Summ(D) have consistently higher ROUGE values than the DUC_Summ(D) for both the SVM as well as the NBC feature selection methods. Another interesting observation is that for every class, NBC summaries have higher number of Better summaries and Mean(ROUGE_Score) than the SVM method. This could be because NBC focusses on features related to document clusters within the class. However SVM focusses on features of the support vectors at the boundaries. Thus NBC summaries tend to be more descriptive with higher ROUGE values than the discriminative summaries of SVM method.

**Table 1.** ROUGE-1 Statistics for Different Classes

| **Class** | **Method** | **Total docs** | No of **Better summaries** | No of **Worse summaries** | Mean (**baseline**) | Mean (**ROUGE_Score**) |
|---|---|---|---|---|---|---|
| 1 | SVM | 107 | 92 | 15 | 0.4911 | 0.5831 |
| 1 | NBC | 107 | **96** | 11 | 0.4911 | **0.6107** |
| 2 | SVM | 129 | 101 | 28 | 0.4316 | 0.5722 |
| 2 | NBC | 129 | **110** | 19 | 0.4316 | **0.6010** |
| 3 | SVM | 155 | 127 | 28 | 0.4662 | 0.5822 |
| 3 | NBC | 155 | **145** | 10 | 0.4662 | **0.6073** |
| 4 | SVM | 147 | 115 | 32 | 0.4504 | 0.5831 |
| 4 | NBC | 147 | **129** | 18 | 0.4504 | **0.6016** |

A major outcome of summarization based on labelled corpus is that, the summaries could be used to train a classifier instead of the documents. We motivate this interesting direction by looking at whether the Auto_Summ(D) are semantically more similar to D than the DUC_Summ(D). For the Auto_Summ(D) generated using SVM based feature selection, we train a SVM classifier on the set of all documents pertaining to two classes at a time. For every document D of the two classes, using the "w" and "b" value learnt by the SVM model, we calculate the value $Weight\mathbf{(X)}$=$w^T$X+b, where X can be Auto_Summ, DUC_Summ and D. Let the corresponding weights be Weight(D) for D, Weight(Auto_Summ(D)) for Auto_Summ(D) and Weight(DUC_Summ(D)) for DUC_Summ(D). To measure the semantic difference between the summaries and the document, we define Error(X) = abs(Weight(X) - Weight(D)), where X can be Auto_Summ(D) or DUC_Summ(D). In the case of DUC summaries, Weight(DUC_Summ(D)) is chosen as that summary from the two whose weight minimizes Error(DUC_Summ(D)).

To be able to substitute the summaries instead of D, it is required that Error(Summary) should be as small as possible. This would imply that the summary will behave in the same way as D. We try to verify that our Auto_Summ have lesser error compared to DUC_Summ. This would mean that Auto_Summ is semantically more similar to D than DUC_Summ. For evaluating the semantic similarities we define a quantity $\triangle SVM$ as

$$\Delta SVM(D) = Error(DUC\_Summ(D)) - Error(Auto\_Summ(D))$$

Similarly, for the Naïve Bayes approach we train a NBC model using the original documents. Now for every document, the posterior probability, given its parent class is found. Similary, for both Auto_Summ and DUC_Summ also the posterior probabilities are found. We define Error(X)=abs(P(Class|X) - P(Class|D)), where X is Auto_Summ(D) or DUC_Summ(D). As in the case above, the summary with the minimum Error is deemed more semantic. We define the value $\Delta$ NBC(D) for document (D) as

$$\Delta NBC(D) = Error(DUC\_Summ(D)) - Error(Auto\_Summ(D))$$

**Table 2.** Semantic similarity comparisons between DUC_Summ and Auto_Summ

| **Class** | **Method** | **Total docs** | No of **Semantically better** summaries | No of **Semantically worse** summaries | Mean (Error (DUC_Summ)) | Mean(Error (Auto_Summ)) |
|---|---|---|---|---|---|---|
| 1 | SVM | 107 | 73 | 34 | 0.4312 | **0.214** |
| 1 | NBC | 107 | 90 | 17 | 0.056 | **0.0139** |
| 2 | SVM | 129 | 93 | 36 | 0.4111 | **0.3648** |
| 2 | NBC | 129 | 92 | 37 | 0.0913 | **0.0237** |
| 3 | SVM | 155 | 103 | 52 | 0.5394 | **0.4041** |
| 3 | NBC | 155 | 132 | 23 | 0.0357 | **0.0154** |
| 4 | SVM | 147 | 105 | 42 | 0.5277 | **0.4033** |
| 4 | NBC | 147 | 115 | 32 | 0.0061 | **0.0030** |

Now for $\Delta NBC(D) \geq 0$ ($\Delta SVM(D) \geq 0$) we define the Auto_Summ(D) as **Semantically better**, else Auto_Summ(D) is defined as **Semantically worse** for the NBC based and SVM based methods respectively. The semantic similarity of the Auto_Summ(D) for all D in the DUC corpus is found out class wise and the statistics are presented in Table 2.

It is observed that the number of Semantically better summaries is higher in all classes of the DUC corpus. The Mean(Error(Auto_Summ)) are also consistently lesser than the corresponding Mean(Error(DUC_Summ)) for all the classes.

## 6 Conclusion and Future Work

The results of Table 1 prove that our summaries are better than the DUC summaries. Majority of the documents in every class have higher ROUGE_Scores than the baseline. The last column of the table gives the Mean(ROUGE_Score) which is consistently higher than the Mean(Baseline). This means our summaries in general have a higher rouge score than their individual baselines. The NBC method consistently outperforms the SVM method. This clearly explains our claim that NBC summaries are more descriptive that SVM summaries.

The results of Table 2 prove that our Auto_Summ are semantically more similar to the documents compared to the human generated summaries of the DUC corpus. This means that our algorithm is able to effectively capture the semantics of the document for a given class better than a human summarizer. Measures like $\Delta SVM$ and $\Delta NBC$ were chosen to compare the difference in the Error between DUC_Summ and Auto_Summ. This was done to discover whether our summaries could be substituted for the documents for learning the classifier. The Mean(Error(Auto_Summ)) was lesser than the Mean(Error(DUC_Summ)) for all classes. Thus, according to the results in Table 2 we can conclude that instead of the DUC_Summ our Auto_Summ could be used for training as they appear to be more semantically closer to the documents.

From this work, we get a few interesting directions in which this work could be extended. Firstly, LDA can be used for abstractive summarization or for phrase extraction from the documents, which is another task with the DUC 2002 data corpus. The usage of summaries in place of documents for classifier learning can be looked at in more detail with the help of topic models.

## References


1. Blei, D.: Probabilistic topic models. *Communications of the ACM*, 55(4):77–84, 2012.
2. Changm, Y., Chien, J.: Latent Dirichlet learning for document summarization. *ICASSP*, 2009.
3. Edmundson, H.P.: New Methods in Automatic Extracting. *J. ACM*, 16(2):264–285, 1969.
4. Luhn, H. P.: The automatic creation of literature abstracts. *Presented at IRE National Convention*, 1958.
5. Conroy, John M., O'leary, Dianne P.: Text summarization via hidden Markov models. *SIGIR*, 2001.
6. Arora, Rachit., Ravindran, Balaraman.: Latent dirichlet allocation based multi-document summarization. *AND*, 2008.
7. Hanna, Wallach., David, Mimno., Andrew, McCallum.: Rethinking LDA: Why Priors Matter. *Advances in Neural Information Processing Systems*, 2009.
8. Manning, C. D., Raghavan, P., Schütze, H.: *Introduction to Information Retrieval*. 1. ed. Cambridge University Press, July 2008.
9. Lin, C.Y.: Rouge: A package for automatic evaluation of summaries. In *Proceedings of the Workshop on Text Summarization Branches Out (WAS 2004)*, 2004.
10. *Oxford English Dictionary* . 2nd ed.Oxford University Press, 1989.