CS6120 Natural Language Processing Course Project Report

# Text Summarization with LSA and BART

Roger He  Yiying Wang  Yixin Fang

**Abstract**

In this project, we explore how to apply two different text summarization models, LSA and BART, to generate summaries on DUC2004 database corpus. We successfully made summarizations for each article and then evaluated the ROUGE scores of the generated summary to the reference summary for each model. We conclude that while current extractive and abstractive summarization methods are promising, they are still far from perfect to summarize complex subject matters in a meaningful way.

## Introduction

With the rise of internet, we now have tremendous information readily available to us. We are bombarded with it from many resource — social media, emails, news, advertisements, ebooks, blog posts, and so on. How cool it would be if instead of reading huge chunks of text data in the form of any type text we could grab the most important information with just a few meaningful sentences! Automatic text summarization still needs more research to reach a higher level of fluency for human readers.

Summarization is the process of shortening a set of data computationally, to create a subset that represents the most important or relevant information within the original content [1]. There are multiple types of summarizations — based on input type, based on desired output or purpose [2]. In our project, we focused on the tasks based on output type. In this kind of text summarization, the task can be either abstractive [3] or extractive [4]. The approach involved in extractive summarization focuses on identifying key passages from the source text and then using it to form a summary. In contrast, abstractive summarization involves understanding the intent and rewriting it.

To implement extractive summarization, we decide to implement LSA [5]. LSA (Latent semantic analysis) extracts semantically significant sentences by information such as words that are used together and which common words are seen in different sentences. A high number of common words among sentences indicates that the sentences are semantically related. Then these semantically significant sentences are then listed together as a summary.

Although extractive summaries are more intuitive in machine level and helpful in highlight the key sentences from a long article, some of these summaries might be harder to interpret without adequate context. To generate more concise and meaning summaries, we could continue with abstractive summarization.

In abstractive summarization task, we have adopted the BART [6] Transformers [7] models. BART (Bidirectional and Auto-Regressive Transformers) model is a de-noising autocoder for pre-training sequence-to-sequence models, which could be thought of as a generalized BERT model [8]. It is implemented as a sequence-to-sequence model with a bidirectional encoder over corrupted text and a left-to-right autoregressive decoder. When properly fine-tuned, BART is effective for text generation and text comprehension tasks. We sourced and adopted the open-source NLP framework developed by HuggingFace, Inc. [9]

**Dataset and Evaluation Metric**

Besides the models which would be implemented in our experiment, choosing appropriate dataset to train the model is the another challenging in text summarization when we encounter with large dataset. Dataset here mentioned about the source texts coming with the corresponding reasonable summary. It should be careful since creating meaningful summaries requires good command of the subject and of natural language which can both be difficult for a machine, while it is unrealistic to manually do summary for each text. For this reason, we chose DUC2004 dataset [10], which provides reference summary along with the dataset. There are 5 tasks in this dataset, including 50 TDT English document clusters (for tasks 1 & 2 respectively), 25 TDT Arabic document clusters (for tasks 3 & 4 respectively), 50 TREC English document clusters (for task 5). As the suggestion provided by NIST, ROUGE [11] is the standard and basic evaluation metrics for DUC2004. ROUGE score highlights the word overlap between the summarized and the source text. ROUGE 1 — measures single word overlap between source and summarized text whereas ROUGE 2 measure bi gram overlap between source and summary. We need to notice that since ROUGE score metric only looks at word overlap and not readability for the text, it is not a perfect metric as text with high rouge score can be a badly written summary.

**Summary methodology**

We aim to follow the guidelines for task 1 and task 2 of DUC 2004. First, a brief overview of the dataset that we acquired: the whole dataset consists of 500 articles in total, categorized into 50 event topics chose by NIST (10 documents per cluster). Task 1 is to generate a very short summary (<= 75 characters) of each document, and task 2 is to generate a short summary (<= 665 characters) of each document cluster. References are given per cluster, we have reference1 and reference2 for each cluster of documents, corresponding to task 1 and task 2 respectively.

In practice, during task 1, it is hard to strictly limit the summarization output to a number of characters without disrupting the result greatly, so for both LSA and BART methods, we are generating a summary for each document in each cluster, combine all the summaries in one cluster together (~10 summaries per cluster), and do another layer of summarization on top of the generated summaries in one cluster to have the final summary for cluster. This way, although the summaries for each document can be long, we still manage to have a general and relatively short summary for the whole cluster. And for task 2, the BART tokenizer's maximum input length is 1024 tokens, which is less than the number of tokens of whole cluster (~10 docs

in the cluster merged into one document), so I send the input document in batches of 1024 tokens to BART tokenizer, details of parameters will be discussed in BART section.

**LSA model**

All languages have their own intricacies and nuances which are difficult for a machine to understand. This can include different words that mean the same thing, and the words with multiple meanings. Humans can distinguish between these words due to being able to understand the context behind these words. However, a machine would not be able to capture the concept since it cannot understand the content in which the words have been used in a new environment. Simply mapping words to documents will not really help. Figuring out the hidden concepts behind the words is the key to solving this issue. This is where LSA comes into play as it attempts to leverage the context around the words to capture the hidden concepts. It is an unsupervised approach that doesn't need training and only uses the context in the input document itself such as word embeddings and term frequency of words used throughout the entire document.

The basic mechanics of LSA involve the following steps. Create the input matrix where columns represent sentences and rows represent words. Then using singular-value decomposition (SVD) to decompose the matrix into three other matrices with the relationship, $A = U \sum V^T$. A is the input matrix of size number of words by number of sentences (m x n). U is the matrix of words by extracted concepts (m x n). The sum represents the scaling values diagonal of the matrix. $V^T$ is a matrix of sentences by extracted concepts. There are multiple ways of ranking and selecting sentences based off these values, but the one chosen was based off the sigma matrix and $V^T$matrix. This means the selected sentences will be based off sentence length, extracted concepts, and how many sentences for each extracted concept are used. [13]

The LSA summarizer created takes in a document to summarize and a second argument which is the number of sentences to reduce the document to. LSA is a lighter weight approach to text summarization which has both benefits and draw backs. It is faster than models that use more data, but often times results are worse in terms of ROUGE scores.

**BART model**

BART is a denoising autoencoder for pretraining sequence-to-sequence models that is trained by corrupting text with an arbitrary noising function and learning a model to reconstruct the original text [6].

This architecture is mainly combined with a bidirectional encoder and a left-to-right decoder, so BART is often seen as generalizing BERT, GPT and many other recent pretraining schemes. The pretrained model that we are using in summarization task is the large model pretrained on CNN news articles, which has 12 layers in encoder and decoder. The architecture is similar to that of BERT with some differences: (1) each layer of decoder additionally performs cross-attention

over the final hidden layer of the encoder, and (2) BART does not have additional feed-forward network before word-prediction. Intuitively, from the first difference with BERT, we can see that the design of architecture of BART focused more on understanding the general meaning of the input, hence in every layer of decoder, a cross-attention with the final representation (hidden layer) of the encoder is added.

Another important aspect of why BART can generalize well the meaning of document is because of its novel in-filling scheme, and the randomly shuffling of sentences during training. The in-filling scheme is letting an arbitrary transformation applied to the original text, including changing its length by allowing arbitrary length of spans of text replaced with a single mask token, the length of span could be 0 as well as the whole sentence. This approach generalizes the original word masking and next sentence prediction objectives in BERT by forcing the model to reason more about overall sentence length and make longer range transformations to the input [6].

In DUC task1, we noticed that if setting “max_length” and “min_length” parameters to generate method of BartForConditionalGeneration, the output will not be complete sentences, most of the times sentences are directly cut in order to meet the maximum number of words parameter (max_length), incomplete sentences most definitely will affect the semantic meaning of summary, so we decided to add another summarization layer on top of the summaries (with no limited length) generated for each document, so that the final summary for each cluster can be relatively short and still generalize well. In the second-layer summarization, I set the “max_length” to 200 and “min_length” to 100 to limit the final summary, otherwise if the summary is too long, the recall for ROUGE may be close to 1, but precision would be very low, which is not ideal.

For DUC task2, the maximum length of tokens for BART tokenizer is 1024 tokens, so in order to treat all documents under one cluster as a single cluster and pass it to tokenizer, I need to pass the inputs in batches and generate summaries in batches as well, because a single pass of the whole cluster will result in indexing error as the model simply does not support input of that length. When generating summaries, I set the max_length to 40 for each batch, so that the final summary (combining summaries of batches together) will not be extremely long, but still a lot larger than 665 characters. This should not affect much since the propose of this project is to compare results generated by LSA and BART.

**Results:**

| | **ROUGE-1** | | | **ROUGE-2** | | | **ROUGE-L** | | |
|---|---|---|---|---|---|---|---|---|---|
| | Precision | Recall | F-1 | Precision | Recall | F-1 | Precision | Recall | F-1 |
| **BART(TASK1)** | 26.51 | 23.02 | 24.58 | 4.92 | 3.99 | 4.39 | 24.89 | 21.61 | 23.07 |
| **BART(TASK2)** | 20.62 | 36.61 | 25.80 | 4.46 | 8.42 | 5.66 | 18.97 | 33.77 | 23.75 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **LSA(TASK1)** | 20.73 | 40.65 | 27.31 | 3.79 | 7.42 | 4.99 | 18.00 | 29.95 | 22.28 |
| **LSA(TASK2)** | 20.03 | 40.03 | 26.58 | 4.91 | 7.32 | 4.91 | 17.62 | 29.26 | 21.76 |

Precision is the ratio of overlapping words compared to the total number of words in the reference summary. Recall is the ratio of overlapping words compared to the total number of words in the model generated summary. F1 score tries to find a balance between the two. A summary with high precision means that a lot of the words in the generated summary are in the reference summary. For example, BART task 1 has a 26.51 precision score. This means 26.51% of the words overlapped are found in the reference summary. A summary with high recall means a lot of the words in the generate summary are also found in the reference summary. For example, LSA task 1 has a 40.65 recall score. This means 40.65 % of the generated summary contains words found in the reference summary. A high precision and low recall mean the summary generated has a lot of matching words with the reference summary but has a lot of unnecessary data that isn't found in the reference summary. A low precision and high recall mean the data in the generated summary is important, but it is missing a lot of words found in the reference summary.

From the results we can see that for BART summaries, ROUGE-1 and ROUGE-L are much higher than ROUGE-2, this is partially because in ROUGE-2, 2-grams are compared which has a lower chance of occurring. However, it is noticeable that Recall scores for both ROUGE-1 and ROUGE-L are much larger than the Precision, this is because our BART generated summaries did not strictly adhere to the restrict of characters for summaries, for most document clusters, the generated summaries are much larger than the references provided, so intuitively, words in references are more likely to appear in generated summaries due to greater word counts, resulting in higher recall values. Judging from the results above, we can see that there are not much difference between the results from task 1 and task 2, however, if we did not add another summarization layer on top of small summaries of task 1, the precisions for task 1 would be very low and resulting in decreased performances.

Compared to BART, LSA has a higher recall score, but lower precision score. This means for an individual phrase or word in the LSA summary, there were higher chances of overlap with the reference summary, but for BART had more matching words with the reference summary.

**Future experiments:**

In this project, we evaluated the generated summaries using pretrained BART model and LSA method. If given more time, we could download DUC2003 datasets and use it as training set to fine-tune the BART model. I suppose we could unfreeze last one or two layers of decoder in BART and train on it. The reason that we could keep the encoder untouched is because after training on extremely large CNN news corpus, the generalization of encoder should be good

enough, we would just need to fine-tune the model with respect to our downstream task, in this case, summarization.

**References:**

[1] Automatic Summarization (Source: https://en.wikipedia.org/wiki/Automatic_summarization)

[2] Types of Summarization (Source: https://medium.com/jatana/unsupervised-text-summarization-using-sentence-embeddings-adb15ce83db1)

[3] Som Gupta, S. KGupta. Abstractive summarization: An overview of the state of the art. Expert Systems with Applications, Pages 49-65, 2019.

[4] Towards Automatic Text Summarization: Extractive Methods (Source: https://medium.com/sciforce/towards-automatic-text-summarization-extractive-methods-e8439cd54715)

[5] Susan T. Dumais. ”Latent Semantic Analysis". *Annual Review of Information Science and Technology*. **38**: 188–230, 2005.

[6] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. arXiv preprint arXiv:1910.13461, 2019

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Advances in neural information processing systems, pages 5998–6008, 2017.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018.

[9] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R’emi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface’s transformers: State-of-the-art natural language processing. ArXiv, abs/1910.03771, 2019.

[10] https://duc.nist.gov/duc2004/

[11] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In Text Summarization Branches Out, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[12] Ozsoy, Makbule & Alpaslan, Ferda & Cicekli, Ilyas. (2011). Text summarization using Latent Semantic Analysis. J. Information Science. 37. 405-417. 10.1177/0165551511408848.

[13] Steinberger, Josef & Jezek, Karel. (2004). Using Latent Semantic Analysis in Text Summarization and Summary Evaluation.