Indian Institute of Technology,
Kanpur

CS671: Introduction to Natural Language Processing

Interim Report and Early Results

Automatic Abstract Generation for Scientific Papers

Authors:
Gaurav (13274)
Palash Chauhan (13455)
Shubham Agarwal (13674)

Supervisor:
Prof. Harish Karnick

September 28, 2016

Abstract

There are two approaches to automatic summarization: extraction and abstraction. Extractive methods work by selecting a subset of existing words, phrases, or sentences in the original text to form the summary. In contrast, abstractive methods build an internal semantic representation and then use natural language generation techniques to create a summary that is closer to what a human might generate. In this early report, we propose an approach for extractive summarization where relevance of words, phrases, or sentences is based on a topic model fit to the corpus.

1 Dataset

The dataset that we chose to work on was the nips-2015-papers available on Kaggle. The main csv file of the dataset contains the following:

- Id - unique identifier for the paper (equivalent to the one in NIPS's system)
- Title - title of the paper
- EventType - whether it's a poster, oral, or spotlight presentation
- PdfName - filename for the PDF document
- Abstract - text for the abstract (scraped from the NIPS website)
- PaperText - raw text from the PDF document (created using the tool pdftotext)

Our main approach is based on a topic-model like LDA [1] which is fit on the entire corpus created by the text from the given papers. The main dataset available on Kaggle provides only around 400 papers. We felt that this would not be sufficient for a good fit of the topic model. Therefore, we went ahead and downloaded all NIPS papers in text format given here. These were obtained by running OCR on the original pdf files. This dataset contains a total of 1740 papers from the period 2000-2012, resulting in a total dataset of around 2100 papers for us!

2 Pre-Processing

A scientific paper involves a lot of things like text, maths, symbols, images etc. Not all of that is relevant for the generation of an abstracts. It is mainly the text which is required. Hence it was needed that the papers are pre-processed. The following steps were involved in the pre-processing procedure:

1. References were removed from the body of the papers.
2. All non-ascii characters were removed from the papers.
3. The abstracts of a paper i was stored in a file ai.txt and the body of the paper was stored in a file called pi.txt
4. A naive method was used to separate the abstract and the body. The dataset does not provide the two separately so python's *find()* method was used to find occurrences of the words *Abstracts* and *Introduction*. A few papers did not have this separation so were ignored.
5. All the abstracts file were appended to form *Abstracts.txt* and all the papers were appended to form *Papers.txt*

After this pre-processing we obtained 1886 paper-abstracts pairs and the corresponding files with papers and abstracts appended.

3 Initial Approach

The first step in our main pipeline is *Sentence Extraction* or *Extractive Summarization*. It involves identifying the most salient sentences of a text. The major downside of applying sentence-extraction techniques to the task of summarization is the loss of coherence in the resulting summary. Nevertheless, sentence extraction summaries can give valuable clues to the main points of a document. We experimented with the following 2 approaches for this task:

3.1 Word-Count based approach

The Sentence Extraction task is basically ranking the sentences in a document and then selecting the top k sentences to form the summary. In this approach, sentences were ranked in the following way:

Algorithm 1 Word-Count-Based-Extractor

procedure GENERATE_SUMMARY
    for <every paper P in corpus> **do**
        $word2freq \leftarrow frequency\ of\ each\ word\ in\ P$
        $score(sentence) \leftarrow sum([word2freq(word)\ for\ word\ in\ sentence])$
        $score(sentence) \leftarrow score(sentence)/length(sentence)$
        return Top K sentences such that $K < maxSummarySize$

3.2 Topic-Model (LDA) based approach

For the purpose of ranking the sentences in a document, a relevance score of the sentence is required. In the previous approach, it was calculated using the frequencies of words in the sentences. This approach uses the similarity between the topic distribution of the entire paper and the topic distribution of an individual sentence (**number of topics = 20**). Higher this similarity, more is the relevance of the sentence.

Algorithm 2 Topic-Model-Based-Extractor

procedure GENERATE_SUMMARY
    $corpus \leftarrow read(Papers.txt)$
    $M \leftarrow LDA(corpus, nTopics = 20)$
    for <every paper P in corpus> **do**
        $paperTopicVector \leftarrow M(P.text)$
        $score(sentence) \leftarrow 1 - cosineDistance(paperTopicVector, M(sentence))$
        return Top K sentences such that $K < maxSummarySize$

Topic 0	networks layer neural image deep layers convolutional hidden sequence recurrent lstm
Topic 1	theorem algorithm proof result appendix analysis properties lemma conditions assumptions
Topic 2	distribution posterior gaussian inference likelihood sampling variational latent prior markov random
Topic 8	figure results performance data table experiments test datasets error methods accuracy training
Topic 18	2010 2011 2012 2013 2014 computer science machine learning conference journal ieee proceedings

Figure 1: Top Keywords

3.2.1 TOPICS

Figure 1 shows the top keywords for few of the topics obtained. It can be seen that Topic 0 corresponds to neural networks, Topic 1 to theorems and proofs, Topic2 to Bayesian Statistics, Topic 8 to results and experiments and Topic 18 to References given in papers.

4 Results and Conclusions

4.0.1 Evaluation

For evaluating the results of these two approaches, we use the ROUGE metric[2]. Under the ROUGE package, 5 metrics are available:

- ROUGE-N: N-gram based co-occurrence statistics.
- ROUGE-L: Longest Common Subsequence (LCS) based statistics. Longest common subsequence problem takes into account sentence level structure similarity naturally and identifies longest co-occurring in sequence n-grams automatically.
- ROUGE-W: Weighted LCS-based statistics that favors consecutive LCSes .
- ROUGE-S: Skip-bigram based co-occurrence statistics. Skip-bigram is any pair of words in their sentence order.
- ROUGE-SU: Skip-bigram plus unigram-based co-occurrence statistics.

We evaluated the abstracts generated from the two approaches using the actual abstracts as reference summaries. For ROUGE-N, $N = 1, 2, 3, 4$ were considered.

	Word-Count			Topic-Model		
	Recall	Precision	F-Score	Recall	Precision	F-Score
ROUGE-1	0.76	0.45	0.55	0.38	0.63	0.44
ROUGE-2	0.58	0.34	0.42	0.27	0.44	0.31
ROUGE-3	0.50	0.29	0.36	0.23	0.39	0.27
ROUGE-4	0.44	0.26	0.32	0.20	0.35	0.24
ROUGE-L	0.75	0.45	0.54	0.36	0.60	0.42
ROUGE-W	0.24	0.25	0.24	0.12	0.37	0.17
ROUGE-S	0.57	0.22	0.29	0.17	0.40	0.20
ROUGE-SU	0.57	0.21	0.29	0.17	0.40	0.21
Average	**0.55**	**0.31**	**0.37**	**0.24**	**0.45**	**0.29**

4.0.2 Observations

- Word-Count based approach has a better recall than the Topic-Model approach. We feel this is due to the the fact that the topic vector inferred for any individual sentence using the trained model does not provide a comprehensive representation of the sentence. Also the text of the papers needs to pre-processed more and only sections relevant to the summary should be kept. That however would be a subjective decision.
- The Topic-Model approach, however, performs better on precision.

5 Further Plan

We plan to take the following steps in the remainder of the project:

1. Pre-process the available text in a better manner so as to obtain a better topic model. We intuitively feel that the topic model approach should perform better. We will experiment with other Topic Models like Hierarchical LDA [3].
2. In case we do not obtain better performance, we will move ahead with the Word-Count approach.
3. Use a pre-trained RNN-based abstractive sentence summarizer, fine tune it on relevant data somehow and then run it on every sentence in the extractive summary obtained from previous steps.

References

[1] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[2] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*, volume 8. Barcelona, Spain, 2004.

[3] Yee Whye Teh, Michael I Jordan, Matthew J Beal, and David M Blei. Hierarchical dirichlet processes. *Journal of the american statistical association*, 2012.