CMPSCI 585 --- Fall 2016 --- Progress Report

Pankaj Bhambhani, SPIRE ID - 30566626

Title - Content Summarization/Extraction of a Book using its Index

Collaborators - NONE

Objective

With the globalisation of internet in today's world, we have an explosion of digital information all over the globe. A lot more people are now interested and curious in getting knowledge and information about various fields than previously, thanks to openly available content (for e.g. via Massive Open Online Courses - MOOCs as they call them). Though books have always been the key source of crucial information (and continue to be), these new generation of explorers prefer newer methods of learning, such as online blog posts and YouTube video tutorials and they find books pretty long to read (no wonder they've invented a new term for long explanation texts - TL;DR "Too Long; Didn't Read"). Also, in general the increasing social media trend is reducing our concentration time period, so a lot of people end up starting books enthusiastically but never finishing them. We want to generate a summary of the contents of a book using its Bibliographical Index as an attempt to bridge the gap between the two options. We want to allow people to be able to read (or at least attempt to read) books in a way that works best for them. The goal is to be able to generate a **short extract** that allows the user to "gain a sense" of a 400-page book with a good index and a good table of contents "in an hour". The reader can then use this extract along with the preface/introduction and the index itself to lookup specific topics in the book, making him/her feel that he has already gone through the book once. **The emphasis is more on covering the breadth of the book with a limited set of sentences than on the sentences themselves being heavily correlated to each other.**

Scope

Automatic Text Summarization is one of the actively-researched fields in Natural Language Processing. The most commonly used methods for this can be roughly divided in three stages - 1) Extraction of relevant and important keywords from the text, 2) Extracting relevant phrases/sentences based on these keywords and other factors, such as its relevance to the title/subtitle and to the core-idea of the text, and 3) Generate natural language sentences which represent the extracted phrases and produce a summary of the text. In our case, we are dealing with a specific collection of document, i.e. books with table of contents and an index. We plan to use the index as the relevant keywords instead of deriving them as in step 1), and then we use steps 2) and 3) as described above.

Current Progress

Data set

I have a dataset of book contents and index for 5 books. However, this set still requires some preprocessing, depending on how the author has specified the index. For example, names of persons are usually specified differently than the way they may have been used in the book (for instance the name Brendan O'Connor would be written in index as O'Connor, Brendan, i.e last name followed by first name). Multiple words could be combined into a single phrase(for instance, in a book about the history of a country, the word "economy" might have different usages during different periods of time (ancient, medieval, modern) and each of these usages might have separate mentions in the index), or one word could be referenced in the index of other words (for instance, an index mention of painting, could also have a reference to the index mentions of the word "art"). I plan to experiment with the way I preprocess the data and see the results. I plan to see whether it is beneficial to split a phrase into its constituent words or not, whether different usages of a word should be treated as different words or a single word. This annotation requires some time, and I have only been able to generate a dataset for a single book (including its index and content). However, I have a set of regular expressions and scripts available with me now that I have done it once, and preprocessing for subsequent books is expected to be quicker. Also, for the preliminary results, I have the entire content of a book as one single text instead of separating them by page number. This means I am only using the index themselves for now, instead of associating them with the numbers. This means that probability of a sentence being in the summary only depends on whether the word contains an index words. I will soon include page numbers, which would then mean that only the sentences in those pages specified by the index will have a probability of being in the summary. For convenience, I am sorting this list of index words alphabetically and removing duplicate mentions of words.

My source is Project Glutenberg and the ebooks available in UMass Amherst Libraries. I am using pdfminer library (https://github.com/euske/pdfminer) to extract index out of the book as well as the book contents itself. The index as well as the content for the book I have processed are included in the accompanying folder. The book is named "The History of India". Note that generating fool-proof list of sentences from a pdf book is a difficult task and I have only run basic regular expressions for the same, so it is bound to contain some errors, most being an abbreviation being treated as end of sentence. I plan to focus on correcting this if I get some time at the end.

Algorithm/Experiment

As mentioned above, for the preliminary I case I consider the entire content of the book as a single text instead of dividing it into pages. This text is then split into a list of sentences, which is then used in the experiment.
Once preprocessing of the data is complete, I then iterate through every sentence and check to see if it contains any of the index words. If it does, I added it to a resulting list of sentences (if it's not already present). Later, I will separate the contents by page numbers, which would

allow to choose a sentence containing an index only it's in a specified list of page mentions for that index.
After getting a filtered list of sentences which are "more relevant than the rest", I then use a content summarizer to produce a final summary. I plan to use two options for text summarization.
1) Extractive Summarization - where sentences are ranked according to a metric and 2) Abstractive Summarization, which is more like the way humans summarize text.

I am looking at following libraries for summarization of my filtered data

1) Sumy - https://github.com/miso-belica/sumy for extractive summarization
2) Tensorflow Textsum - https://github.com/tensorflow/models/tree/master/textsum for abstractive summarization.

Preliminary Results

As mentioned in the previous sections, the preliminary experiment has been performed on the content and index of a single book, titled "The History of India". For this index, I have considered the entire content of the book as a single text instead of dividing it into pages. This text is then split into a list of sentences, which is then used in the experiment. The summarization library used in this case is **sumy**, which internally uses nltk for tokenisation and other tasks. Sumy is able to parse HTML and plain text with various available algorithms. For my case, I have used the LexRank algorithm, which is a variant of the PageRank algorithm mentioned in the reference. This algorithm constructs a graph with sentences. It then uses a similarity metric to define the similarity between two sentences, and creates an edge between two sentence vertices every time their similarity crosses a predefined threshold. It then uses the PageRank algorithm to choose the top sentences for summarization.

The mentioned book has a list of 3667 sentences. Initially I tried to separate phrases)like "Government of India", "Economic liberalization", etc) into separate words. However, that approach filtered out very few sentences as some of the individual words like "India", "Indian", "Act", etc. were very common and many sentences have these indices. I then retained the phrases as is and instead focused on other preprocessing tasks. I changed the names of people as mentioned in the previous section. For the preliminary case, I removed and references of an index word in other index words, since right now I am only concerned with the presence of an index in a sentence. Later, I will add page numbers which will help extracting sentences as mentioned in the previous sections.

The latter approach produced a filter list of 1866 sentences. I then used the LexRank summarization to summarize this list of sentences, keep the number of sentences in the summary to 100. The resulted summary is in the accompanying folder. This is obviously a preliminary result and can be improved upon by the approaches I mentioned in the previous sections. I am still looking at available approaches for evaluating this extract, since it's not necessary that it be a summary. It should only convey the broader meaning of the text, and for now it's not necessary that the sentences are highly correlated with each other. Any suggestions in this regard are highly welcome.

Reference

I plan to use the following work for the project:

1) Al-Hashemi, Rafeeq. "Text Summarization Extraction System (TSES) Using Extracted Keywords." *Int. Arab J. e-Technol.* 1.4 (2010): 164-168. This paper summarizes the key steps of summarizing texts using the steps we described above, namely parsing the text, extracting keywords, ranking sentences and then generating summary.
2) Erkan, Günes, and Dragomir R. Radev. "LexRank: Graph-based lexical centrality as salience in text summarization." *Journal of Artificial Intelligence Research* 22 (2004): 457-479.
3) Text Summarization using Tensorflow https://research.googleblog.com/2016/08/text-summarization-with-tensorflow.html I am considering using this library for text summarization though I am considering alternatives.