‘Novel’ Summarizer and Keyword identifier using text rank with sentence farm detection

Authors: Rohan Phadke(phadke.r@husky.neu.edu)
Rushabh Shah(shah.rushabh1@husky.neu.edu)

Introduction

The task at hand is an extension and application of classic text summarization problem in language processing. Summarization is the task of getting an input in the form of text and generating a concise and holistic summary as the output which conveys the same meaning as the original text, but in fewer words. Natural Language Processing primarily deals with two types of summary generation, namely Extractive and Abstractive.

Extractive type of summarization deals with generating a summary purely using the input words only and it does not rephrase the input text in any way. On the other hand, abstractive type of summary is a general type of summary which rephrases the words in input text to generate a comprehensive summary. It is not bounded by the input text.

We will be providing two new modifications to the conventional text rank algorithm used for extractive text summarization and then analyse their effectiveness as compared to the text rank algorithm, in this paper.

Task explanation

We are analysing two novel series and generating summaries on a chapter-by-chapter basis. The importance of studying this problem lies in the fact that a human mind can possibly remember only 10% of what it reads. So, it becomes important to summarise text and present in such a way that it conveys the holistic meaning of the prose in fewer words. According to another study, “Summaries as short as 17% of the full text length speed up decision making twice, with no significant degradation in accuracy.” So, summaries in general are a very powerful tool and if used properly, can speed up a variety of tasks. Solution to this problem can be extended to other types of literature and can be generalized for subject text-books too.

System Input and Output

The input for this problem is books from two famous novels, namely The Harry Potter series and The Hunger games series. This input is in the form of text files which has first been cleaned to remove any textual anomalies. These cleaned files served as input for the text summarization algorithm.
The output is a file containing the summary of the input chapter. It is an extractive summary consisting of only the most important sentences of the chapter generated by summarization algorithm. There is no training data for the method we are proposing as modification to text rank, as it is an unsupervised technique of text summarization.

Sample Example

Sample example is provided in form of 4 different files:

One input file of The Harry Potter series Book 1, Chapter 9, namely, input.txt
Three output files each of the summaries provided by three different methods, namely, summary1.txt, summary2.txt and summary.txt

Related Work

There has been some relevant work done already in the field of text summarization which is as follows:

- **Extraction Based Text Summarization** ([https://stanford.edu/~sxie/cs221-final-report.pdf](https://stanford.edu/~sxie/cs221-final-report.pdf))
  This paper mainly addresses the extractive type of summarization technique. It focuses on features and methodologies used to extract a summary from an article and maintaining the time stamp to achieve high coherency.

  This paper talks in detail about abstractive as well as extractive summarization techniques. It delves into topics like tf-idf, cluster based method, LSA method, text summarization using neural networks, automatic TS based on fuzzy logic and much more.

  This paper focuses on generic text summarization by ranking and extracting sentences from the original documents. It uses IR text-ranking method and latent semantic analysis technique to identify semantically similar sentences and use to create summaries.

  This paper articulates the extractive text summarization technique for documents that are highly redundant in nature. For example, user reviews, product reviews, feedback, etc. It talks in detail about Path Redundancy Scores and how it is used to score sentences and place them as a part of summary.
Our improvements and modifications

The problem that we studied in different from all the work that has been already done previously in this field. It has the following differences:

- **Input**
  Most of the relevant work done is either done on short passages or articles. Some of the papers deal only with top 100 words of an article for summary generation. In general, shorter the article/text, more accurate(easy) it is to generate a convincing summary. Our problem deals with huge amount of data in form of book series and each series containing many chapters. Pre-processing step also differs as compared to what has been already done previously.

- **Methodologies**
  We started out by proposing many techniques that can be used to generate the summary. We have incorporated 3 main methodologies which can be compared against each other to determine the winning summary produced by each method. Previous work dealt with separate independent methods with no(limited) comparison of the summaries generated by each individual method.
  The text rank algorithm will be modified in two separate ways: 1) by boosting importance of certain characters and events in the story, and 2) using this boosted importance of characters to generate action graphs for every chapter and using a technique like ‘identification of link farm spam pages’ that is used on page rank.

- **Output**
  The output generated in many of the previous works doesn’t consider the coherency and time stamping for summary generation. Our problem deals with series of novels and so maintaining the same occurrence of events as found in original text becomes critical for accurate summary generation.

Pre-processing of Data

We worked on Harry Potter and Hunger Games series for implementation of our text summarization on chapter-by-chapter basis. We obtained these books from various sources and have cleaned them to standardize the text that we use as the input for our project. The text in the books obtained was too unstructured for use in the project and so it required extensive cleaning and parsing.

In cleaning phase, we did away with all unnecessary details like page numbers, preface, index, reviews etc. and generated plaintext files of the books having only kept the chapter name and chapter content for each chapter. $N^{th}$ chapter has its title in upper case on the $(2N-1)^{th}$ line and the chapter is on the $2N^{th}$ line.

Input data varies in size as per book’s content but roughly each book is about ~ 1 to 1.5 MB in size. Size of data is reasonable enough to generate summary in just seconds and it doesn’t serve as a bottleneck for the algorithm to process summary.
Features Used

We started off by using syntax trees to extract features on every keyword such as the context of each scene, the locations and the actions performed by the character. Also, we divided a set of proper nouns into places, people and other objects. But we realized that this feature extraction isn’t that useful for summary generation and so we decided to drop it. We came with another feature generation which deemed more appropriate and helpful for accurate summary generation.

The most important, and the only feature that we are currently using in the project is defining relations with the important characters by generating graph where each node is a character and each edge is a count of interactions between these nodes. These weights help in updating weights on the corresponding nodes and result in iterative weight updates.

Methods Used

We are using the following methods to implement the text summarization (with improvements):

- **TextRank**

  TextRank algorithm used the same intuition as of PageRank algorithm. Instead of working with web pages, TextRank works with sentences by assigning weights to each sentence based on the content (proper nouns in our case). This is a simple technique to implement as it is unsupervised and requires no training data for summary generation. This technique works very well for generating extractive type of summary.

- **TextRank with important character identification**

  We have extended the TextRank algorithm for generate better quality summaries by identifying the keywords among the sentences. By doing this, we are increasing the weights of keyword nodes that are more important than other words. This is done in such a way that the total weight at the start and at the end remains the same. With this, we can generate better summaries because we are considering the ‘importance’ of keywords. The characters in a story are the agents that drive the entire story ahead. Their actions set the course of the events that occur in their world. It is only fair that these words have higher importance than other stop words as well as non-stop words occurring frequently throughout the story. To boost their importance, their frequency counts need to be modified in a reasonable way. To do that, we reduce the count of non-important words and assign this deficit count to the important words according to their count. We calculate ratio of important to unimportant words, and use that ratio to decide on how much the count is to be boosted. The result is that non-important words, which outnumber the important words by a huge margin are awarded a marginal decrease in their count which doesn’t affect the text rank algorithm much, but the corresponding deficit is divided up into a much smaller set of important words, which considerably increases their count, boosting their importance.

- **TextRank with important character identification and character farm spam detection**

  We have implemented ‘character farm detection’ which shares the same ideology as ‘link farm detection’ in case of PageRank. This farm detection technique is an iterative algorithm which iterates on the graph and reduces importance of characters that are less important. Consequently, this results in the identification of farm of sentences that contain the characters as suggested by farm detection. In each iteration, we propagate the ‘badness’ or ‘goodness’ of each node to every other node it interacts with. Ultimately we are combining this propagated weight of ‘badness’ with the normal weight we get using Text Rank
generated graphs. This results in better summary generation as we are removing spam sentences.

We begin with assigning initial weights to the list of possible important characters in the range of +1 to -1 based on their current calculated importance. In each iteration, their actions and interactions with any other agent affect the weight of both agents. The idea behind this is that some agent who interacts heavily with the protagonist is most certainly an important character to the story in some way, whereas the extras or characters that appear a few times here and there are usually not significant to the overall story unless there is considerable interaction with other main characters. Agents that do not interact with any other agents are mostly not part of the important summary of the story. Hence their weight value remains the same at the end of an iteration. However, an agent which interacts a few times and performs certain actions with important characters, but has frequency importance less than another character which doesn’t interact with others, will be gradually pulled up in importance after every iteration. Thus, in a sense, we stretch the importance scale from the range (+1, -1) to a higher positive value and a lower negative value to get to the more apt importance coefficient of characters.

**Dataset Size**

There is no explicit data for training/development as we have implemented an Unsupervised approach for text summary generation. All the data that is available with us is assigned for testing purpose only. The dataset size depends on the book series under consideration. Separately, each book in The Harry Potter series is ~ 1 MB to 1.5 MB. This size varies with the chapter content and number of chapters in each book.

**Evaluation Metrics**

The evaluation task for the generated summary mainly considers a comparison between the summary that is generated using the algorithm with that generated manually be a human. The following two metrics are the considered the best for performing this type of evaluation:

- **Rouge-N**
  Rouge-N is N-gram word evaluation metric between the system generated summary and the gold summary, that is, human generated. More precisely, it is the ratio of the count of N-gram phrases which occur in system generated and human generated summaries. Intuitively, it translates to the recall value which measures the number of common N-gram phrases. Generally, for text summarization evaluation task we consider Rouge-1 or Rouge-2 metrics. We consider Rouge-3 only if we have long gold summaries.

  For our project, we are considering Rouge-(sentence length) which basically means that we are looking for the ratio of common number of sentences in the summaries generated by us and that in the human generated summaries.

- **BLEU**
  BLEU metric is a modified representation of precision which is extensively used in machine translation evaluation. Just like the Rouge metric, it considers the similarity quotient between the system generated summary and human generated summary. In general, BLEU metric for our model is defined as the ratio on number of common sentences between the
system generated summaries and human generated summaries divided by the number of sentences in the system generated summaries. This metric is nothing but precision which is modified to avoid any problem when a model generated summary contains repeated relevant information.

Both these metrics are the most powerful tools to evaluate a text summarization task as they consider the common information present in model generated and human generated summaries.

Baseline for the Project

As mentioned in the section of ‘Methods Used’, the baseline for text summarization task the TextRank Algorithm. We are building on that further to generate summaries with keyword identification and character farm identification improvement. These results in generating summaries of better quality.

Each method produces a slightly different summary.

For summary generated by each method for The Harry Potter series Book 1 Chapter 4:
The size of summary generated has been set to a default of 20% of original text size. Thus, all 3 methods will give a summary with same number of sentences. For this chapter, we get a summary of 67 sentences. When we compare the summaries generated by method 1 vs that by method 2, we find that there are no changes in the summaries. The boosting isn't overwhelming enough to change the summary much, suggesting that text rank algorithm gives a satisfactory result. However, on comparing summaries generated by method 1 vs that by method 3, we find that there are four sentences out of 67 that these summaries differ in.
The respective numbers of evaluation are as shown below.

Confusion matrices for The Harry Potter Book 1 Chapter 4:

1) TextRank

<table>
<thead>
<tr>
<th></th>
<th>in model summary</th>
<th>not in model summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>in human summary</td>
<td>40</td>
<td>27</td>
</tr>
<tr>
<td>not in human summary</td>
<td>27</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Rouge-(sentence length) score = Recall = 40/67

BLEU score = Precision = 40/67

2) TextRank with important character identification

<table>
<thead>
<tr>
<th></th>
<th>in model summary</th>
<th>not in model summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>in human summary</td>
<td>41</td>
<td>26</td>
</tr>
<tr>
<td>not in human summary</td>
<td>26</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Rouge-(sentence length) score = Recall = 41/67

BLEU score = Precision = 41/67

3) TextRank with important character identification and character farmspam detection

<table>
<thead>
<tr>
<th></th>
<th>in model summary</th>
<th>not in model summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>in human summary</td>
<td>43</td>
<td>24</td>
</tr>
<tr>
<td>not in human summary</td>
<td>24</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Rouge-(sentence length) score = Recall = 43/67

BLEU score = Precision = 43/67

References

- NLTK documentation (http://www.nltk.org/)
- Networkx (https://networkx.github.io/)
- Identifying Link Farm Spam Pages by Wu and Davison (http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.87.5445&rep=rep1&type=pdf)