### *Book Summarization*

### *Abstract*

Summarization is the process of shortening a text document with software, in order to create a summary with the major points of the original document. In this document I have tried to explain various extractive methods to construct a summarizer and methods to evaluate summaries created using summarizer. At the end I have performed summarization of a book and have evaluated it against gold summary.

Keywords: Extractive summary, Evaluation, LSA, LexRank.

### *Types of Summaries:*

Extractaction based Summary:

In this summarization task, the system extracts objects from the entire document, without modifying the objects themselves. Extraction techniques copy the information deemed most important by the system to the summary (for example, key clauses, sentences or paragraphs), Examples of this include key phrase extraction, where the goal is to select individual words or phrases to "tag" a document, and document summarization, where the goal is to select whole sentences (without modifying them) to create a short paragraph summary.

***Abstraction-based summarization:***

Abstraction involves paraphrasing sections of the source document. In general, abstraction can condense a text more strongly than extraction.

In this document we focus on developing an extractive summarizer with the following properties:

**Informative:** we aim for a summary that provides enough information to provide the need for reading the source documents.

**Generic:** we want the summary to capture generic information from input document. This is in contrast to query-focused summaries, where the summary must be sensitive to a query that captures the information need of the user.

***General Architecture of a Text Summarizer:***

As a general approach, we adopt the following architecture to model the task of a summarizer:

**Text representation:** capture the significant aspects of the input text in an abstract text representation. Tokenization of words using module from NLTK , for morphological inflections we use Stemming module in NLTK. Also all words taken into account are only content words i.e. all stop words are filtered from text .

**Ranking:** We calculate word frequencies and sort the elements in the text representation in terms of information significance.

**Extraction:** extract from the sorted representation the elements that will be rendered in the summary. Coherence and cohesion constraints may force us to expand the selection to more items than dictated by significance only. In addition, the requirement to avoid redundancy (select the same content more than once) may also force us to not only select items only based on significance.

**Realization:** render the selected content into fluent text.

***Extractive Methods :***

**Luhn:**

Step 1 : Vectorization of the document.

Step 2: Determine the top Words by calculating the word frequency in the document..

Step 3: Select top sentences based on term frequency of the word in a sentence.

Step 4: Write top sentences to the output file in order of the occurrence of the sentences.

**LSA (Latent Semantic Analysis):**

In Latent Semantic Analysis first step is to create the word by title matrix. In this matrix, each index word is a row and each title is a column. Each cell contains the number of times that word occurs in that title. The parse method takes a document, splits it into words, removes the ignored characters and turns everything into lowercase so the words can be compared to the stop words. If the word is a stop word, it is ignored and we move on to the next word. If it is not a stop word, we put the word in the dictionary, and also append the current sentence number to keep track of which sentence the word appears in. Once all sentences are parsed, all the words (dictionary keys) that are in more than 1 sentence are extracted and sorted, and a matrix is built with the number of rows equal to the number of words (keys), and the number of columns equal to the sentence count. The raw matrix counts are modified so that rare words are weighted more heavily than common words using TF-IDF (Term Frequency – Inverse Document Frequency) weighting method. Once the matrix is built we use Singular Value Decomposition or SVD to analyze the matrix that can capture and represent word combination patterns which are recurring in the corpus. The magnitude of the singular value indicates the importance of the pattern in a document.

**LexRank:**

[LexRank](https://www.cs.cmu.edu/afs/cs/project/jair/pub/volume22/erkan04a-html/erkan04a.html) is an unsupervised graph based approach similar to TextRank. LexRank uses IDF (Inverse Document Frequency)-modified Cosine as the similarity measure between two sentences. This similarity is used as weight of the graph edge between two sentences. LexRank also incorporates an intelligent post-processing step which makes sure that top sentences chosen for the summary are not too similar to each other.

*Centroid based method*

The document is split into sentences and construct a graph where the nodes correspond to the sentences in the input document and the edges indicate sentence similarity.

To assess sentence similarity, we consider a term vector representation of the sentences: consider all the words that appear in all the sentences of the input document set as (w). Each sentence is represented as a vector X where the coordinate xi is 1 if the word wi appears in the sentence, 0 otherwise. Compute dot product of two vector (Boolean vectors values are 0 or 1) cosine of two vector to check similarity of two sentences.

The centroid of a cluster of sentences: the centroid is a fictional sentence, which is encoded by the vector equal to the "average" of the vectors representing the sentences in the cluster.

*Centrality based method:*

The first step of the model is to compute a sentence similarity matrix - for each sentence, using cosine distance to each other sentence in the cluster we compute di,j (diagonal values of matrix is 1). We can then apply a threshold on matrix to filter the similarity matrix. Given the filtered similarity matrix for a given threshold, assessing sentence centrality is to count the number of similar sentences for each sentence. We define degree centrality of a sentence as the degree of the corresponding node in the similarity graph. This definition of degree centrality, considers all neighbors of a sentence as equally significant. The idea of Lexrank is to rank the neighbors in terms of prestige: a sentence is more central if it is close to many high-prestige sentences. Recursively, the prestige of a sentence depends on the prestige of its neighbors. Markovian process , ensures aperiodic and irreducible scoring of the sentences in the matrix.

**TextRank:**

TextRank is a unsupervised algorithm based on weighted-graph.

* Pre-process the text: remove stop words and stem the remaining words
* Create a graph where vertices are sentences.
* Connect every sentence to every other sentence by an edge. The weight of the edge is how similar the two sentences are. The weight of an edge between two sentences is the percentage of words appearing in both of them.
* Run the PageRank algorithm on the graph.
* Pick the vertices(sentences) with the highest PageRank score

**SumBasic:**

Text representation: we represent the input text as a [bag of words](http://en.wikipedia.org/wiki/Bag_of_words_model) - that is, the document is represented as vectors of stemmed content words with their frequency.

Ranking: We compute a significance score for input sentences based on the frequency of the words they contain.

Extraction: extract from the sorted representation the elements that will be rendered in the summary.

Pick the sentence S\* with the highest score and append it to the output summary. Re-compute the scores of the remaining sentences and iterate until the summary has reached the maximum allowed length.

Realization: output the sentences in the order they have been selected.

**KL:**

Introduced for content model Multi Document Summarization

KL-Sum also a sentence selection algorithm, where a target length for the summary is fixed (L words). The objective of the summarizer is to find a set of sentences whose length is less than L words and whose unigram distribution is as similar as possible to the source document set.

***Evaluating Summaries***



Figure : Taxonomy of summary evaluation measures

There are mainly two categories of evaluation techniques: Intrinsic and Extrinsic.

***Intrinsic method*** uses Human generated summaries for comparisons as they are considered to be intellectual summaries.

Intrinsic method has two approaches: Text-quality evaluation and Content evaluation.

*Content evaluation* has two approaches: content-based and co selection-based.

The co-selection measures sentence extracts. It counts how many reference summary sentences the candidate summary contains i.e. summary generated by automatic summarizer. Co-selection measures exactly the same sentences. It ignores the fact that two sentences can contain the same information even if they are written differently. Furthermore, summaries written by two different annotators expressing the same content do not in general share identical sentences. The main assessment metrics of co-selection measures are precision, recall and F-score.

*Precision, Recall and F-measure:*

Precision (P) is computed as no. of sentences occurring in both candidate and reference summaries divided by the no. of sentences in the candidate summary.

Recall (R) is the no. of matched sentences in both candidate and reference summaries divided by the no. of sentences in the reference summary.

F-score is harmonic average of both precision and recall.

*Content-Based Measures:*

The Content-based measures actually compare the words in a sentence, rather than the entire sentence.

*Cosine Similarity:*

We can calculate the similarity between pairs of the documents using ‘[cosine similarity](https://en.wikipedia.org/wiki/Cosine_similarity)’ algorithm, which measures the [cosine](https://en.wikipedia.org/wiki/Trigonometric_functions#cosine) of the angle between two vectors. In this case, each document can be presented as a vector whose direction is determined on a set of the TF-IDF values in the space. Where X and Y represent system summary and reference summary respectively based on the vector space model. If the two vectors are pointing to similar directions then we can say these two documents are similar. So in order to measure and compare the similarity we want to calculate the cosine of the angle between the two vectors.

*Unit Overlap:*

Another similarity measure is Unit Overlap, where representations of X and Y is based on sets of words or lemmas.

*The ROUGE (Recall-Oriented Understudy for Gisting Evaluation):*

ROUGE evaluation is an automatic (and therefore cost-effective) evaluation method. It compares a summary with a gold-standard summary (one generated manually). For LexRank, Luhn and LSA methods I have made use of the [Sumy](https://github.com/miso-belica/sumy) summarization library which implements these algorithms. We used the ROUGE-1 metric to compare the discussed techniques. Specifically, it is the ratio of the count of N-gram phrases which occur in both the model and gold summary, to the count of all N-gram phrases that are present in the gold summary. Another way to interpret it is as the recall value which measures how many N-grams from the gold summaries appeared in the model summaries.

Generally for summarization evaluation, only ROUGE-1, ROUGE-2 metrics are used, rationale being that as we increase N, we increase the length of the N-gram word phrase that needs to be matched completely in both the gold and model summary.

Example:

Gold Summary: a good diet must have apples and bananas.  
Model Summary: apples and bananas are must for a good diet.

If we use the ROUGE-1, the score is 7/8 = 0.875.

For ROUGE-2, it is 4/7 = ~0.57.

The above ratios can be interpreted as the amount of relevant information that our algorithm managed to extract from the set of all the relevant information, which is exactly the definition of recall, and hence Rouge is recall based.

***Extrinsic method*** measures the performance based on how the summaries generated are useful for a certain task.

**Problem Statement :**

Read a PDF file of any book and write a python 2.7 program to identify summarize each chapter in 1 page.

In previous task I had created a PDF summarizer, using the Sumy 0.7.0 Module for automatic summarization of text documents and HTML pages in python.

***Performance evaluation :***

In this task I studied the working of six algorithms mentioned in extractive summary and has created a python program to evaluate the score of each generated summary against the gold summary (ideally written by human). The generated summary is in plaintext format encoded in “utf-8”. The following evaluation is done on the book *“Harry Potter and the Sorcerer's Stone by J. K. Rowling”* and the two reference summaries are created using content from web (links are given below).The summary program generates summary using Luhn, LSA, Lex-Rank, Text-Rank, Sum—Basic, KL of every chapter in 30 sentences. The reference summaries are also saved as plaintext and encoded in “utf-8” otherwise program will not be able to read them. For comparison sentence count is set to 25 sentences. The evaluation is done on the basis of unigram, bigram and Longest Common Subsequence methods using ROUGE.

Reference summary 1 is created from <http://www.sparknotes.com/lit/harrypotter/> and is used as a gold summary to compare with the system generated summaries.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Luhn | Lsa | Lex-Rank | Text-Rank | Sum-Basic | KL |
| Recall | 0.170068 | 0.260417 | 0.123153 | 0.168919 | 0.099602 | 0.148810 |
| F-score | 0.290698 | 0.413223 | 0.219298 | 0.289017 | 0.181159 | 0.259067 |
| Cosine Similarity | 0.882150 | 0.872330 | 0.867152 | 0.893626 | 0.751828 | 0.819018 |
| Cosine Similarity (document) | 0.988480 | 0.993830 | 0.964957 | 0.985602 | 0.890247 | 0.985447 |
| Unit overlap | 0.246773 | 0.236676 | 0.233766 | 0.228147 | 0.138868 | 0.214581 |
| Unit overlap (document) | 0.597216 | 0.622683 | 0.513037 | 0.648123 | 0.379866 | 0.554303 |
| Rouge-1 | 0.379258 | 0.424678 | 0.284633 | 0.308857 | 0.137017 | 0.309614 |
| Rouge 2 | 0.102457 | 0.120882 | 0.070352 | 0.091290 | 0.022055 | 0.078169 |
| Rouge-L(Summary Level) | 0.007159 | 0.005394 | 0.008874 | 0.006660 | 0.013903 | 0.008081 |

Table 1: Scores of summary generated using given algorithm against reference summary 1.

Figure 1 Graphical representation of data in Table 1

Reference summary 2 is taken from <https://www.shmoop.com/harry-potter-sorcerers-stone/> and is used as a gold summary to compare with the system generated summaries.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Luhn | Lsa | Lex-Rank | Text-Rank | Sum-Basic | KL |
| Recall | 0.170068 | 0.260417 | 0.123153 | 0.168919 | 0.099602 | 0.148810 |
| F-score | 0.290698 | 0.413223 | 0.219298 | 0.289017 | 0.181159 | 0.259067 |
| Cosine Similarity | 0.880963 | 0.887603 | 0.879807 | 0.880897 | 0.734741 | 0.840535 |
| Cosine Similarity (document) | 0.988480 | 0.993830 | 0.964957 | 0.985602 | 0.890247 | 0.985447 |
| Unit overlap | 0.272799 | 0.283760 | 0.215986 | 0.235639 | 0.114229 | 0.211713 |
| Unit overlap (document) | 0.597216 | 0.622683 | 0.513037 | 0.648123 | 0.379866 | 0.554303 |
| Rouge-1 | 0.338491 | 0.396518 | 0.233075 | 0.272727 | 0.112186 | 0.264023 |
| Rouge 2 | 0.071605 | 0.087921 | 0.048480 | 0.064796 | 0.016669 | 0.054466 |
| Rouge-L(Summary Level) | 0.006145 | 0.005796 | 0.007318 | 0.005649 | 0.012398 | 0.006447 |

Table: Scores of summary generated using given algorithms against reference summary 2.

Figure 2 Graphical representation of Table 2

***Comparative analysis:***

Table 1 and Table 2 contain the Recall, F-score, Cosine similarity, Unit overlap, Rouge-1,Rouge-2, Rouge-L scores of method of text summarization. As evident from both the table the recall score and the F-score of the LSA summarizer is the best followed by Luhn and Text-Rank. The high values of Cosine similarity tell us that both the reference summary and generated summary are similar.

***Graphical Analysis:***

Here we observe that ROUGE-1 gives the highest score in respect to Rouge-2 score. It is due to fact that, if there is a matching term in ROUGE-2, then it will also be present in ROUGE-1; however reverse is not always true. Score of Rouge-L (ROUGE- Longest Common Subsequence)(Summary Level) is also very low which shows that there are very few subsequences found in reference and generated summary.

***Conclusion:***

In this report , we have discussed important extractive summarization methods. The experimental result on a book show that LSA(Latent Semantic Analysis) performs better than others. So LSA summaries are more concise and give essential information. The result however can vary on different data. So it cannot be concluded that the LSA will always give the best results. We have taken summary of 25 sentences to compare , change in number may also vary the results.