Document Summarization with TextRank and Enhanced Rouge

Executive Summary: In this project, a document summary is produced for every document by selecting the most relevant sentences. The TextRank algorithm is used to select these sentences. This algorithm creates a weighted non-directed graph from all the sentences in a document. The edge weights indicate the similarity between the two connecting sentences. Then a formula similar to the PageRank formula used to rank webpages is used to rank the sentences. To produce a summary of k-sentences, the top-k sentences are used. This algorithm is run on a dataset of scientific papers.

The document summaries produced will be scored for similarity with the abstracts of this paper and this similarity score will be reported, along with the summaries. For scoring, an enhanced version of the ROUGE (Recall-Oriented Understudy for Gisting Evaluation) measure is used. Instead of considering only the n-grams present in the reference summary and the machine-generated summary for Rouge (as is usual), this project proposes an 'Enhanced' Rouge which also considers other relevant n-grams. This relevance is indicated by parts of speech tags of the words present in the n-grams. These words are present in both the summary and the abstract, though the n-grams themselves are only present in the machine-generated summary.

Goal: The goal of this project is to produce document summaries using TextRank and experiment with a new evaluation measure for document summarization.

Background: The task of document summarization involves producing automatic summaries of documents by a computer program. The target is to produce a short summary paragraph from a long article spanning several paragraphs. The two most important ways to produce such a summary is **extraction** and **abstraction**. Extraction involves using key phrases or sentences from the document itself to produce a summary. Abstraction involves paraphrasing the document. The latter involves Natural Language Generation which is a hard research problem on its own.

Extraction could be of two types, supervised and unsupervised. Supervised techniques involve giving a set of documents (training set) to the learning algorithm along with their ideal summaries. Then this classifier produces summaries on a new set of documents (test set). Unfortunately, the performance of this technique depends highly on whether the test set and training set are from the same/similar domains.

Unsupervised algorithms, on the other hand, produce document summaries without any training. Thus their performance is not domain dependant. That's why in this project, an unsupervised approach to document summarization will be implemented.

The standard evaluation measure for document summarization is a bag of words technique - ROUGE(Recall-Oriented Understudy for Gisting Evaluation). The system is recall-based to encourage the

summarization algorithm to include as much relevant information as possible. For one reference summary, this is the ratio of the number of the candidate n-grams in the system-produced summary to the total number of candidate n-grams. These candidate n-grams are usually all the n-grams (including n-grams with only stopwords) in the reference summary. To improve this, an 'Enhanced' ROUGE is proposed in this project.

Description of TextRank: The target document is first tokenized into sentences. Then each sentence is tokenized into a collection of its words. A weighted undirected graph is formed among the sentences. The weight of an edge between two sentences is the similarity between these two sentences. The similarity score is given below:

$$Sij = |Si \cap Sj| / (\log|Si| + \log|Sj|)$$

Si = Sentence i

Sj = Sentence j

|Si| = Number of words in Si

|Sj| = Number of words in Sj

|Si ∩ Sj| = Number of common words between Si and Sj

|Si ∪ Sj| = Number of words in the union of Si and Sj

In this project, other similarity measures like longest common subsequence, similarity score with only nouns and adjective have been explored.

After the graph is produced, the scores of the vertices (individual sentences) are obtained by the following formula:

$$WS(Vi) = (1-d) + d*(\Sigma(wji/\Sigma Wj)*WS(Vj)))$$

WS(Vi) = Weight of Vertex i

WS(Vj) = Weight of Vertex j

d = Randomness parameter

wji = Weight of edge between Vertex i and Vertex j

ΣWj = Sum of weights of edges from Vertex j

The first summation sums over all the edges to vertex I from vertices j.

Evaluation Using 'Enhanced' ROUGE: After this, the sentences are reverse sorted according to their scores. The top-k sentences are used to produce a summary of k documents. This summary is then

evaluated against a gold standard (an abstract in case of a paper) using the 'Enhanced' ROUGE (Recall-Oriented Understudy for Gisting Evaluation) measure. This new evaluation measure looks for the relevant words present in the n-grams of both the machine-generated summary and the abstract. The relevance of a word is indicated by its parts of speech tag. In this project, words with the simplified POS tags – ADJ, N, NP have been considered relevant. Then the ngrams in the machine-generated summary containing the relevant words but absent in the reference summary have been included in the numerator of Enhanced Rouge as a relevant n-gram adjusted by n of the n-gram.

Enhanced Rouge = (#Common n-grams + (#Relevant n-grams/n)) / (#n-grams in reference summary)

→ number of items

Explanation with Example:

Reference Summary: *Calcification is the process in which calcium builds up in body tissue where there normally isn't any calcium.*

Machine Summary: *Build up of calcium in the body where it is normally absent is calcification.*

Though both these summaries are defining the same process - calcification, there are no common tri-grams between the two. Hence,

Plain 3-Rouge score= #common trigrams/#tri-grams in reference = 0/16 = 0

Now, let's consider

Trigrams containing calcification in the machine summary – "absent is calcification"

Trigrams containing calcium in the machine summary – "up of calcium", "of calcium in", "calcium in the"

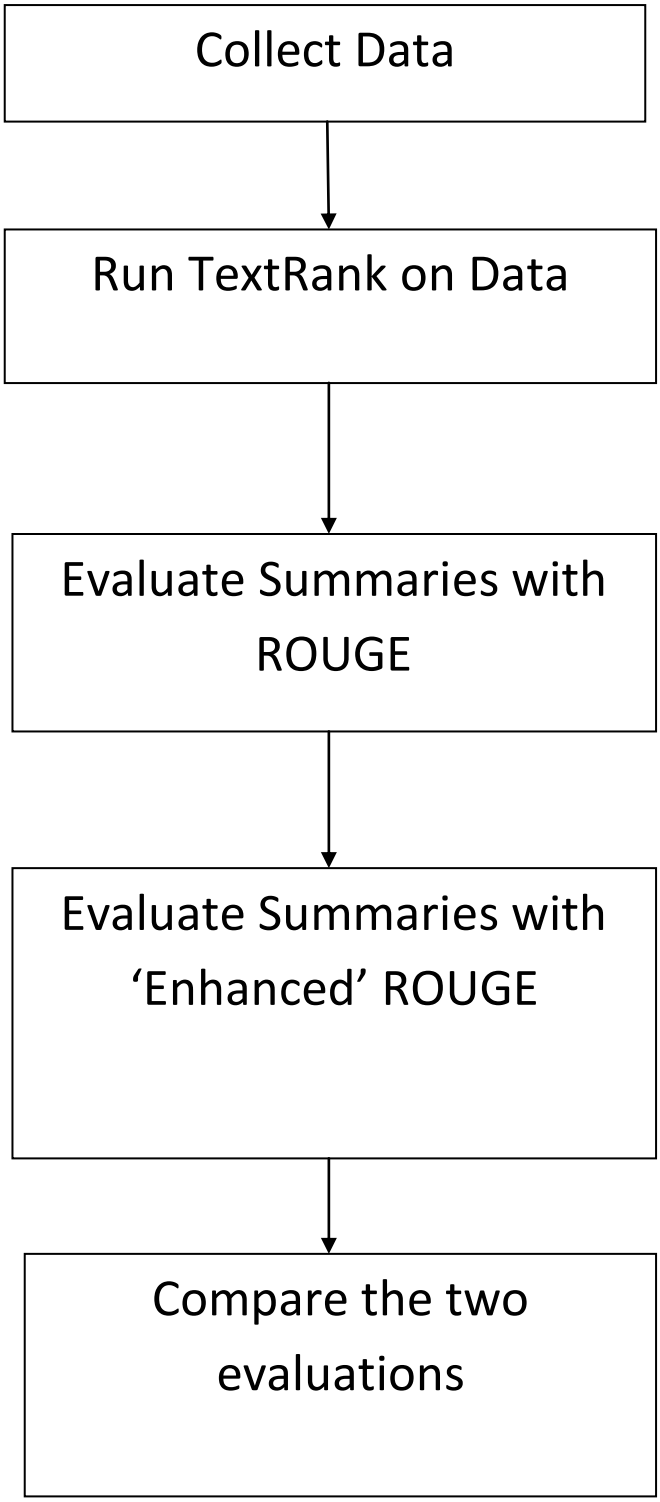
Trigrams containing body in the machine summary – "in the body", "the body where", "body where it"

These 3 words – calcification, calcium, body are meaningful words (words with POS tags - ADJ, N, NP) present in both the summaries. Therefore, it is logical to include the trigrams in the machine summary which contain these words as relevant trigrams. However, since every relevant word would be present in 3 trigrams (unless they are at the beginning or end), we divide the number of these trigrams obtained by n(3 in this case). Note that in this example, there are 7 such tri-grams.

Enhanced 3-Rouge score = (#Common n-grams + (#Relevant n-grams/n)) / (#n-grams in reference summary) = (0 + 7/3)/16 = 7/48 = .1458

It is still low, considering that both the sentences define the same concept, but a considerable improvement over zero in the usual 3-Rouge case.

Project Flow-Chart:



Methodology: The dataset is a plain text dataset of scientific papers of KDD Cup 2003 (website). It can be downloaded here. A dataset of papers has been used because for a paper, there is a readily available reference summary – its abstract. The python programming language has been used in this project, due to its brevity and ease of use. The code can be divided into the following parts:

a) Conversions from and to various Natural Language tokens. Data cleaning has been included in this part. The functions defined are:
   - clean_sents(sents_li): Given a list of sentences, this function excludes sentences which contain @,*,!,_,+,/,\\,= and returns a new list
   - text_to_sent(text): Given a string of text, this function tokenizes it into a list of sentences. Regular expression from python's re package is used for this purpose. It calls clean_sents(sents_li) to send only meaningful sentences.
   - sent_to_words(sent): Given a sentence string, this function converts into lower case, splits it to return the list of constituent words.
   - extract_abstract_text(file_n): Given a file name, it reads the file and returns the abstract text and main text.
   - valid_ref_n_grams(reference_sents): Given a list of abstract sentences, it indicates if there are any meaningful n-grams in them. If not, we don't include the paper in the analysis.

b) Creating the TextRank graph. This includes forming the graph from the sentences and finding the TextRank scores of the sentences. However, this does not include finding similarity between sentences. That warrants its own part. The functions defined in this part are
   - sents_to_graph(sents_li):Given the list of sentences, this function creates the graph using a similarity measure(described in the next part). The graph is a dictionary (also known as associate array, map or table). Every sentence is indicated by its position in the dictionary. The key is a sentence position while its value is a list of (sentence position, similarity score between the 2 sentences) pairs. For example, a key value pair 1: [(2,.1),(3,.09),(7,.9)] indicate that the sentence in position 1 is connected to the sentences in positions 2,3,7 and the similarity scores between 1-2, 1-3, 1-7 are .1,.09 and .9 respectively. Note that though this graph is fully connected, some of the edge weights could be zero. Also, it is an undirected graph, since it is a similarity graph. Another interesting point – there are no self-edges.
   - get_newscores(current_scores,graph): Given the graph and the TextRank score of each sentence in the graph, this function calculates the new TextRank score of each sentence.
   - difference(new_scores,current_scores): Given the new TextRank scores and the current TextRank scores, this function finds the differences between the old and new TextRank score of every sentence. It then returns the sum of these differences.
   - compute(graph,epsilon): After initializing the initial TextRank scores of every sentence, this function repeatedly calls get_newscores(current_scores,graph) until the scores are no longer changing as indicated by a difference value lower than epsilon.

c) Similarity scores between two sentences. The functions defined are

- sent_sim_score_num_words(sent1,sent2): Given two sentences, this function returns the similarity score between the two, based on the formula given in the description of TextRank. This is the similarity measure which has been used in this project.
- sent_sim_score_num_nn_words(sent1,sent2): Similar to the previous function. However, words are only included in the count of the numerator if they have an 'interesting' parts of speech tag – noun or adjective (N, NP, ADJ). This produces extremely low similarity scores and hence has not been used in this project.
- sent_sim_score_lcs(sent1,sent2): Given two sentences, it divides the length of the longest common subsequence between the two by the same denominator as in the last two functions. This has not been used because calculating lcs takes considerable time. Since the similarity score calculation is done for every pair of sentences, this becomes a bottleneck.
- lcs(X,Y,m,n): Given two strings X and Y and their lengths m and n, this function returns the length of the longest common subsequence in the two strings.

d) Calculation of rouge and finding parts of speech tags. The functions defined are

- rouge(n,reference_sents,machine_sents,counter): Given the sentence list of the reference summary(abstract) and the machine-generated summary(TextRank summary), this function returns the usual Rouge score.
- enhanced_rouge(n,reference_sents,machine_sents,counter): Given the sentence list of the reference summary(abstract) and the machine-generated summary(TextRank summary), this function returns the 'Enahnced' Rouge score.
- fn_assign_POS_tags(words_list): Given a list of words in sequence, this function predicts the POS tags of the words. The Viterbi algorithm with bigram HMM taggers have been used to train a classifier on the news section of the Brown corpus. This function uses that trained classifier to make predictions.

Results: The average Rouge scores obtained for n=1,2,3,4&5 and with top 3 sentences chosen are given below in the following table:

N	Rouge Score	Enhanced Rouge Score
1	0.33069	0.33069
2	0.10590	0.27477
3	0.03861	0.23377
4	0.01737	0.22107
5	0.00930	0.22970

Results with top 4 sentences:

N	Rouge Score	Enhanced Rouge Score
1	0.36944	0.36944
2	0.11965	0.33215
3	0.04381	0.29110
4	0.01903	0.27706
5	0.00981	0.28681

Results with top 5 sentences:

N	Rouge Score	Enhanced Rouge Score
1	0.39925	0.39925
2	0.13439	0.38515
3	0.04960	0.34465
4	0.02131	0.33128
5	0.01068	0.34206

As might be expected, both the Rouge scores increase with increasing k.

The main difference between the two scores is that in the usual case, the score reduces drastically as n increases. However, since the same summary is being evaluated, that is not intuitive. The 'Enhanced' Rouge score, on the other hand, decreases much more slowly. Notably, the scores for n=1 and n=5 are of the same order of magnitude.

Challenges Faced:

The following challenges were faced during the project:

1. Messy data – The papers have been converted from LATEX to plain text. However, the symbols used in LATEX have not been removed. This has resulted in arbitrary whitespace and special characters, which made any tokenization and finding similarity between tokens difficult. Sentences containing some of these special characters have been removed from analysis. This also resulted in the failure of the NLTK functions for the common tasks like sentence tokenization, word tokenization. For example, the PunktSentenceTokenizer of NLTK failed to tokenize the text to sentences. Custom code had to be developed for all this usual operations.
2. Unclear indication of paper sections – This point is related to the previous point. While abstracts have often been indicated by the word Abstract, other sections of the papers have not always been similarly indicated. Papers where the word Abstract is absent or whose abstracts produce zero n-grams after cleaning have not been included in the analysis. Without such messy data, both the Rouge scores is likely to be higher.
3. More involved methods to enhance Rouge require more data – Instead of considering POS tags, named entity tags of words in the summaries could have been considered. Since the most

pertinent entity for papers is a concept, an approach to identify concepts was required. The paper "An Unsupervised Method for General Named Entity Recognition and Automated Concept Discovery" describes such an approach. However, this approach requires considerable data and is not feasible for a short document like a summary.

4. Infeasibility of computationally intensive methods for similarity computation between sentences – Since the similarity score has to be computed for all pairs of sentences in the body of the paper, even a moderate-length text of 200 sentences requires $^{200}C_2 = 19900$ computations of similarity. And that is only to find the similarity scores between sentences for 1 document. Hence, it becomes prohibitively long to run TextRank with a complex similarity formula.

Future Work:

Any or all of the following steps can be performed to build on the current work.

- Calculation of word similarity – To measure number of common words, instead of checking if one word is identical to another, their cosine similarity based on their word-vector representation can be checked. Only words with very high similarity would be considered identical. Similarly, words with very low distance between them in the WordNet graph can be considered identical. The caveat is that these calculations would increase the time required for calculation of similarity. This can also be used for finding common and relevant n-grams for Rouge.
- Alternate algorithm for Summary generation – TextRank relies on a bag of words approach to generate summaries. Algorithms which take into account word features could potentially produce better summaries. For example, the approach described in the paper "An Unsupervised Method for General Named Entity Recognition and Automated Concept Discovery" could be used to select summarizing sentences which are most 'related' to the discovered concepts. Quantifying this 'relation' would be the subject of future research.
- Stemming – Stemming could be used in two places. First, while comparing words to calculate sentence similarity. Words which only differ due to tense, singular/plural would then be considered identical, thus meaningfully boosting word similarity. Secondly, it could used to find the n-grams for both the Rouge scores. Thus phrases like '*builds up*' and '*build up*' in the example would be included, at least for 2-Rouge.
- Summary Similarity Score based on Meaning Representation – The crux of Rouge is still a bag of words approach. But instead of counting common or relevant n-grams, the meaning representations of the reference and machine-generated summaries could be obtained. A similarity score between the 2 meaning representations could then be used as a summary evaluation metric. Developing good meaning representations and a sensible similarity score would be the topics of future research.

Sources:

1. Automatic Evaluation of Summaries Using N-gram Co-Occurrence Statistics, by Chin-Yew Lin and Eduard Hovy
2. An Unsupervised Method for General Named Entity Recognition and Automated Concept Discovery, by Enrique Alfonseca and Suresh Manandhar
3. TextRank: Bringing Order to Texts, by Rada Mihalcea and Paul Tarau
4. BLEU: A Method For Automatic Evaluation of Machine Translation, by Kishore Papineni, Salim Roukos, Todd Ward and Wei-Jing Zhu
5. KDD Cup Dataset, 2003. Download link
6. Definition of Calcification: http://www.healthline.com/health/calcification#Overview1
7. Longest Common Subsequence Implementation: http://www.geeksforgeeks.org/dynamic-programming-set-4-longest-common-subsequence/
8. Mining of Massive Datasets – Chapter 5, by Jure Leskovec, Anand Rajaraman, Jeffrey D Ullman
9. POS Tagging and Syntactic Parsing – Lecture 2, by Heng Ji, NLP course at RPI
10. Tagging with HMMs, Lecture by Michael Collins