Introduction

Text summarization is a problem that has been studied for many years and various solutions and systems have been proposed and implemented. the task can be described as building a machine that is capable of making a summary when given a text. The key idea is to find a “smaller” set of data that contains the information of the entire data set.

There are two main approaches to this problem: Extraction and Abstraction. Extraction methods build the summary by selecting sentences having rich information in the given text. Meanwhile, abstractive methods aim at creating a summary using key ideas from the text by building an internal semantic representation and then use natural language generation techniques. Beside these methods, machine learning techniques in information retrieval and text mining have also been used to develop systems that aid users with the task of summarization. Examples for such systems include MAHS = Machine Aided Human Summarization, which highlights candidates to be included in the summary, and systems that depend on post-processing by a human (HAMS = Human Aided Machine Summarization).

Proposed method

We propose an extractive method using TextRank with [similarity measure adjustments]. We prefer this extractive approach to an abstractive one due to following reasons:

_The idea of abstractive method sounds appealing, however, it requires deep-learning algorithms and natural language processing, making it difficult to develop the model within the one-week time constraint.

_TextRank is a well-known and accurate extractive text summarization method whose foundation is the PageRank algorithm. The PageRank algorithm was developed by the Google Team and implemented in the Google Search.

_The performance of the algorithm depends strongly on how similarity measure is defined, which could be an area of improvement.

_The algorithm is an unsupervised learning method which doesn’t a training process.

Description of the proposed method

TextRank

A given text is considered as a graph having vertices which are connected by edges.

Each sentence in a text is a vertex.

We compute the score of each vertex and then rank them to select high-score sentences forming the summary.

The score is computed based on these principles:

_The importance score of a vertex is measured by the number of vertices to which it links and the importance of those vertices.

_Links between two sentences are defined by similarity measures. This measure will then be integrated into computing the importance score as weights.

The formula for computing the score of a vertex

$$Importance\ Score(V_i) = (1-d) + d \times \sum_{V_j \in In(V_i)} \frac{w_{ij}}{\sum_{V_k \in Out(V_j)} w_{jk}} \times Importance\ Score(V_j)$$

V_i: *the vertex i*

d: *the damping factor*

$In(V_i)$: *the set of vertices that link to* V_i

$Out(V_j)$: *the set of vertices that* V_j *links to*

w_{ij}: *the similarity measure between vertex i and vertex j*

The damping factor is the probability of jumping from one vertex to another random vertex in the graph. d could be set between 0 and 1. In our case, we choose d to be 0.85.

The original TextRank algorithm use this similarity measure to compute the weights

$$w_{ij} = weight(V_i, V_j) = \frac{|\{w_k | w_k \in V_i\ and\ w_k \in Vj\}|}{\log|V_i| + \log\ |V_j|}$$

w_k: *words in the vertex*

Starting from arbitrary values assigned to each vertex in the graph, the computation iterates until convergence below a given threshold is achieved. After running the algorithm, an importance score is associated with each vertex in the graph. The final values obtained after TextRank runs to completion are not affected by the choice of the initial value.

We also found some similarity measures that have shown to gain significant improvements:

Longest Common Substring From two sentences: we identify the longest common substring and report the similarity to be its strengths.

Cosine Distance The cosine similarity is a metric widely used to compare texts represented as vectors. We used a classical TF-IDF model to represent the documents as vectors and computed the cosine between vectors as a measure of similarity. Since the vectors are defined to be positive, the cosine results in values in the range [0,1] where a value of 1 represents identical vectors and 0 represents orthogonal vectors.

BM25 is known to be a ranking function as state of the art of information retrieval techniques. It is a variation of a TF-IDF model using probabilistic model

Given two sentences R and S, BM25 is defined as:

$$BM25(R,S) = \sum_{i=1}^{n} IDF(si).\frac{f(s_i,R).(k+1)}{f(s_i,R) + k(1-b+b.\frac{|R|}{avgDL})}$$

where k and b are parameters and avgDL is the average length of the sentences in our text. For our implementation, we choose k=1.2 and b=0.75.

IDF(s_i) is the inverse document frequency. This term will be negative when s_i appears more than half of the sentences in the text, which could cause some problems later on, therefore we made some adjustment to the formula:

$$IDF(s_i) = \begin{cases} \log(N - n(s_i) + 0.5) - \log(n(s_i) + 0.5) \ if \ n(s_i) < \frac{N}{2} \\ \varepsilon.avgIDF \ if \ n(s_i) \geq \frac{N}{2} \end{cases}$$

epsilon is between 0.3 and 0.5 and avgIDF is the average IDF of all of the words in the text.

Background

Extractive text summarisation includes ranking words and sentences and use them to generate a summary. Therefore, these summaries include the most important sentences in the input in order. Some algorithms are supervised learning, which requires a corpus to identify descriptive words in a document to be summarized. To identify those words, statistical approach such as likelihood or TF-IDF. An unsupervised approach for text summarization is more robust because it doesn't require corpus.

A graph-based ranking algorithm like Google's PageRank is called TextRank. The TextRank model decides the importance of a vertex within a graph. The importance of a vertex is evaluated by the number of votes that are casted on that vertex, and for each vertex, the weight of its vote is influenced by its importance. Hence, with the same number of votes, the vertex that has more votes from important vertices will be considered more important.

Traditionally, directed graph is applied to TextRank model. In case of loosely connected graphs, undirected graph can be considered. Let $G = (V, E)$ be a directed graph with the set of vertices V and set of edges E:

- E will be a subset of $V \times V$
- For a vertice V_i, predecessors are all the vertices $In(V_i)$ that point to V_i
- V_i will point to $Out(V_i)$ call successors

Unlike web surfing context where a page is hardly include multiple or partial links to another page, graph-based ranking in the context of natural language texts, may exist multiple or partial links between

the vertices. Therefore, it is useful to indicate the strength of the connection between two vertices V_i and V_j as weight w_{ij} .

Evaluation

Method	File reference	ROUGE-1	ROUGE-2	ROUGE-L
Textrank	ref0	0.3533	0.0638	0.1490
	ref1	0.3703	0.0645	0.1533
BM25	ref0	0.3755	0.0674	0.1560
	ref1	0.3890	0.0687	0.1591

For testing the proposed variations, we used the database of the 2003 Document Understanding Conference (DUC). To evaluate results we used version 0.1.5 of the Sumeval package (https://github.com/chakki-works/sumeval), where ROUGE-1, ROUGE2 and ROUGE-L were used as metrics. The final result is an average of these two scores which we test from two test file. We found the resulting scores of the original algorithm 2.3% improvement over the baseline.