Graph-Based Algorithms for Text Summarization

Khushboo S. Thakkar
Department of Computer Science & Engineering
G. H. Raisoni College of Engineering, Nagpur, India
e-mail:khushboo.thakkar86@gmail.com

Dr. R. V. Dharaskar
Professor & Head, Dept. of Computer Science & Engineering
G. H. Raisoni College of Engineering, Nagpur, India
e-mail: rajiv.dharaskar@gmail.com

M. B. Chandak
HOD, Dept. of Computer Science & Engineering
Shri Ramdeobaba Kamla Nehru Engineering College, Nagpur, India
e-mail: chandakmb@gmail.com

Abstract-**Summarization is a brief and accurate representation of input text such that the output covers the most important concepts of the source in a condensed manner. Text Summarization is an emerging technique for understanding the main purpose of any kind of documents. To visualize a large text document within a short duration and small visible area like PDA screen, summarization provides a greater flexibility and convenience. This paper presents innovative unsupervised methods for automatic sentence extraction using graph-based ranking algorithms and shortest path algorithm.**

Keywords- Text Summarization,ranking algorithm, HITS, PageRank.

I. INTRODUCTION

Due to the rapid growth of the World Wide Web, information is much easier to disseminate and acquire than before. Finding useful and favored documents from the huge text repository creates significant challenges for users. Typical approaches to resolve such a problem are to employ information retrieval techniques. Information retrieval relies on the use of keywords to search for the desired information. Nevertheless, the amount of information obtained via information retrieval is still far greater than that a user can handle and manage. This in turn requires the user to analyze the searched results one by one until satisfied information is acquired, which is time-consuming and inefficient. It is therefore essential to develop tools to efficiently assist users in identifying desired documents.

One possible means is to utilize automatic text summarization. Automatic text summarization is a text-mining task that extracts essential sentences to cover almost all the concepts of a document. It is to reduce users' consuming time in document reading without losing the general issues for users' comprehension. With document summary available, users can easily decide its relevancy to their interests and acquire desired documents with much less mental loads involved.

II. GRAPH-BASED ALGORITHMS

A. Graph-based Ranking Algorithm

Graph-based ranking algorithms are essentially a way of deciding the importance of a vertex within a graph, based on information drawn from the graph structure. In this section, two graph-based ranking algorithms – previously found to be successful on a range of ranking problems are presented. These algorithms can be adapted to undirected or weighted graphs, which are particularly useful in the context of text-based ranking applications.

HITS

Hyperlink-Induced Topic Search (HITS) (also known as Hubs and authorities) is a link analysis algorithm that rates Web pages, developed by Jon Kleinberg. It determines two values for a page: its authority, which estimates the value of the content of the page, and its hub value, which estimates the value of its links to other pages.

In the HITS algorithm, the first step is to retrieve the set of results to the search query. The computation is performed only on this result set, not across all Web pages.

Authority and hub values are defined in terms of one another in a mutual recursion. An authority value is computed as the sum of the scaled hub values that point to that page. A hub value is the sum of the scaled authority values of the pages it points to. Some implementations also consider the relevance of the linked pages.

The Hub score and Authority score for a node is calculated with the following algorithm:

- Start with each node having a hub score and authority score of 1.
- Run the Authority Update Rule.
- Run the Hub Update Rule.
- Normalize the values by dividing each Hub score by the sum of the squares of all Hub scores, and dividing each Authority score by the sum of the squares of all Authority scores.
- Repeat from the second step as necessary.

HITS produces two sets of scores – an "authority" score, and a "hub" score:

$$HITS_A(Vi) = \sum_{Vj \in In(Vi)} HITS_H(Vj) \quad (1)$$

$$HITS_H(Vi) = \sum_{Vj \in Out(Vi)} HITS_A(Vj) \quad (2)$$

PageRank

PageRank is a link analysis algorithm, named after Larry Page, used by the Google Internet search engine that assigns a numerical weighting to each element of a hyperlinked set of documents, such as the World Wide Web,

978-0-7695-4246-1/10 $26.00 © 2010 IEEE
DOI 10.1109/ICETET.2010.104

with the purpose of "measuring" its relative importance within the set. The algorithm may be applied to any collection of entities with reciprocal quotations and references. The numerical weight that it assigns to any given element *E* is also called the *PageRank of E* and denoted by *PR*(*E*).

The name "PageRank" is a trademark of Google, and the PageRank process has been patented (U.S. Patent 6,285,999). However, the patent is assigned to Stanford University and not to Google. Google has exclusive license rights on the patent from Stanford University. The university received 1.8 million shares of Google in exchange for use of the patent; the shares were sold in 2005 for $336 million

Google describes PageRank:

> "PageRank relies on the uniquely democratic nature of the web by using its vast link structure as an indicator of an individual page's value. In essence, Google interprets a link from page A to page B as a vote, by page A, for page B. But, Google looks at more than the sheer volume of votes, or links a page receives; it also analyzes the page that casts the vote. Votes cast by pages that are themselves "important" weigh more heavily and help to make other pages "important"."

In other words, a PageRank results from a "ballot" among all the other pages on the World Wide Web about how important a page is. A hyperlink to a page counts as a vote of support. The PageRank of a page is defined recursively and depends on the number and PageRank metric of all pages that link to it ("incoming links"). A page that is linked to by many pages with high PageRank receives a high rank itself. If there are no links to a web page there is no support for that page.

PageRank is a probability distribution used to represent the likelihood that a person randomly clicking on links will arrive at any particular page. PageRank can be calculated for collections of documents of any size. It is assumed in several research papers that the distribution is evenly divided between all documents in the collection at the beginning of the computational process. The PageRank computations require several passes, called "iterations", through the collection to adjust approximate PageRank values to more closely reflect the theoretical true value.

In the general case, the PageRank value for any page **u** can be expressed as:

$$PR(\upsilon) = \sum_{v \in Bu} \frac{PR(v)}{L(v)} \qquad (3)$$

i.e. the PageRank value for a page u is dependent on the PageRank values for each page v out of the set B_u (this set contains all pages linking to page u), divided by the number $L(v)$ of links from page v.

The PageRank theory holds that even an imaginary surfer who is randomly clicking on links will eventually stop clicking. The probabilities, at any step, that the person will continue is a damping factor d. Various studies have tested different damping factors, but it is generally assumed that the damping factor will be set around 0.85.

The damping factor is subtracted from 1 (and in some variations of the algorithm, the result is divided by the number of documents in the collection) and this term is then added to the product of the damping factor and the sum of the incoming PageRank scores.

That is,

$$PR(A) = 1 - d + d \cdot \frac{PR(B)}{L(B)} + \frac{PR(C)}{L(C)} + \frac{PR(D)}{L(D)} + \ldots \qquad (4)$$

For each of these algorithms, starting from arbitrary values assigned to each node in the graph, the computation iterates until convergence below a given threshold is achieved. After running the algorithm, a score is associated with each vertex, which represents the "importance" or "power" of that vertex within the graph. Notice that the final values are not affected by the choice of the initial value, only the number of iterations to convergence may be different.

Text As Graph

To enable the application of graph-based ranking algorithms to natural language texts, we have to build a graph that represents the text, and interconnects words or other text entities with meaningful relations. Depending on the application at hand, text units of various sizes and characteristics can be added as vertices in the graph, e.g. words, collocations, entire sentences, or others. Similarly, it is the application that dictates the type of relations that are used to draw connections between any two such vertices, e.g. lexical or semantic relations, contextual overlap, etc.

Regardless of the type and characteristics of the elements added to the graph, the application of graph-based ranking algorithms to natural language texts consists of the following main steps:

1. Identify text units that best define the task at hand, and add them as vertices in the graph.
2. Identify relations that connect such text units, and use these relations to draw edges between vertices in the graph. Edges can be directed or undirected, weighted or unweighted.
3. Iterate the graph-based ranking algorithm until convergence.
4. Sort vertices based on their final score. Use the values attached to each vertex for ranking/selection decisions

Sentence Extraction

To apply TextRank, we first need to build a graph associated with the text, where the graph vertices are representative for the units to be ranked. For the task of sentence extraction, the goal is to rank entire sentences, and therefore a vertex is added to the graph for each sentence in the text.

Formally, given two sentences S_i and S_j, with a sentence being represented by the set of N_i words that appear in the sentence: $Si = W^i_1, W^i_2, \ldots, W^i_{Ni}$, the similarity of S_i and S_j is defined as:

$$\text{Similarity}(S_i, S_j) = \frac{|W_k | W_k \in S_i \& W_k \in S_j|}{\log(|S_i|) + \log(|S_j|)} \quad (5)$$

Other sentence similarity measures, such as string kernels, cosine similarity, longest common subsequence, etc. are also possible, and we are currently evaluating their impact on the summarization performance.

3: BC-HurricaineGilbert, 09-11 339
4: BC-Hurricaine Gilbert, 0348
5: Hurricaine Gilbert heads toward Dominican Coast
6: By Ruddy Gonzalez
7: Associated Press Writer
8: Santo Domingo, Dominican Republic (AP)
9: Hurricaine Gilbert Swept towrd the Dominican Republic Sunday, and the Civil Defense alerted its heavily populated south coast to prepare for high winds, heavy rains, and high seas.
10: The storm was approaching from the southeast with sustained winds of 75 mph gusting to 92 mph.
11: "There is no need for alarm," Civil Defense Director Eugenio Cabral said in a television alert shortly after midnight Saturday.
12: Cabral said residents of the province of Barahona should closely follow Gilbert's movement.
13: An estimated 100,000 people live in the province, including 70,000 in the city of Barahona, about 125 miles west of Santo Domingo.
14. Tropical storm Gilbert formed in the eastern Carribean and strenghtened into a hurricaine Saturday night.
15: The National Hurricaine Center in Miami reported its position at 2 a.m. Sunday at latitude 16.1 north, longitude 67.5 west, about 140 miles south of Ponce, Puerto Rico, and 200 miles southeast of Santo Domingo.
16: The National Weather Service in San Juan, Puerto Rico, said Gilbert was moving westard at 15 mph with a "broad area of cloudiness and heavy weather" rotating around the center of the storm.
17. The weather service issued a flash flood watch for Puerto Rico and the Virgin Islands until at least 6 p.m. Sunday.
18: Strong winds associated with the Gilbert brought coastal flooding, strong southeast winds, and up to 12 feet to Puerto Rico's south coast.
19: There were no reports on casualties.
20: San Juan, on the north coast, had heavy rains and gusts Saturday, but they subsided during the night.
21: On Saturday, Hurricane Florence was downgraded to a tropical storm, and its remnants pushed inland from the U.S. Gulf Coast.
22: Residents returned home, happy to find little damage from 90 mph winds and sheets of rain.
23: Florence, the sixth named storm of the 1988 Atlantic storm season, was the second hurricane.
24: The first, Debby, reached minimal hurricane strength briefly before hitting the Mexican coast last month.

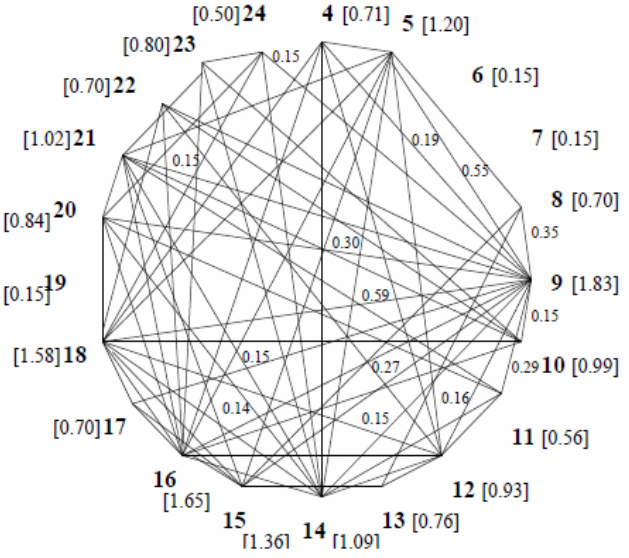

Figure 1. Sample graph build for sentence extraction

The resulting graph is highly connected, with a weight associated with each edge, indicating the strength of the connections established between various sentence pairs in the text. The text is therefore represented as a weighted graph.

After the ranking algorithm is run on the graph, sentences are sorted in reversed order of their score, and the top ranked sentences are selected for inclusion in the summary.

Fig 1 shows a text sample [6], and the associated weighted graph constructed for this text. The figure also shows sample weights attached to the edges connected to vertex 9, and the final TextRank score computed for each sentence. The sentences with the highest rank are selected for inclusion in the abstract. For this sample article, the sentences with id-s 9, 15, 16, 18 are extracted, resulting in a summary of about 100 words, which according to automatic evaluation measures, is ranked the second among summaries produced by 15 other systems.

B. Shortest-path Algorithm

Extraction based summarization normally produces summaries that are somewhat unappealing to read. There is a lack of flow in the text, since the extracted parts, usually sentences, are taken from different parts of the original text. This can for instance lead to very sudden topic shifts. The idea behind the presented method of extracting sentences that form a path where each sentence is similar to the previous one is that the resulting summaries hopefully have better flow. This quality is however quite hard to evaluate. Since the summaries are still extracts, high quality summaries should still not be expected [3].

Building the graph

When a text is to be summarized, it is first split into sentences and words. The sentences become the nodes of the graph. Sentences that are similar to each other have an edge between them. Here, similarity simply means word overlap, though other measures could also be used. Thus, if two sentences have at least one word in common, there will be an edge between them. Of course, many words are ambiguous, and having a matching word does not guarantee any kind of similarity. Since all sentences come from the same document, and words tend to be less ambiguous in a single text, this problem is somewhat mitigated.

All sentences also have an edge to the following sentence. Edges are given costs (or weights). The more similar two sentences are, the less the cost of the edge. The further apart the sentences are in the original text, the higher the cost of the edge. To favor inclusion of "interesting" sentences, all sentences that are deemed relevant to the document according to classical summarization methods have the costs of all the edges leading to them lowered.

The cost of an edge from the node representing sentence number i in the text, S_i, to the node for S_j is calculated as:

$$\text{cost}\, i, j = \frac{(i - j)^2}{\text{overlap}\, i,j \cdot \text{weight}\, j} \quad (6)$$

and the weight of a sentence is calculated as:

weight j = (1 + overlaptitle, j)

$$\cdot \frac{(1 + \sum_{w \in Sj} tf(w))}{(\sum_{w \in text} tf(w))}$$

$$\text{. early (j). } \sqrt{(1 + |\text{ edgej }|)} \quad (7)$$

Since similarity is based on the number of words in common between two sentences, long sentences have a greater chance of being similar to other sentences. Favoring long sentences is often good from a smoothness perspective. Summaries with many short sentences have a larger chance for abrupt changes, since there are more sentence breaks.

Constructing the summary

When the graph has been constructed, the summary is created by taking the shortest path that starts with the first sentence of the original text and ends with the last sentence. Since the original text also starts and ends in these positions, this will hopefully give a smooth but shorter set of sentences between these two points.

The N shortest paths are found by simply starting at the start node and adding all paths of length one to a priority queue, where the priority value is the total cost of a path. The currently cheapest path is then examined and if it does not end at the end node, all paths starting with this path and containing one more edge are also added to the priority queue. Paths with loops are discarded. Whenever the currently shortest path ends in the end node, another shortest path has been found, and the search is continued until the N shortest paths have been found.

III. COMPARISON

TextRank works well because it does not only rely on the local context of a text unit (vertex), but rather it takes into account information recursively drawn from the entire text (graph). Through the graphs it builds on texts, TextRank identifies connections between various entities in a text, and implements the concept of recommendation. A text unit recommends other related text units, and the strength of the recommendation is recursively computed based on the importance of the units making the recommendation. In the process of identifying important sentences in a text, a sentence recommends another sentence that addresses similar concepts as being useful for the overall understanding of the text. Sentences that are highly recommended by other sentences are likely to be more informative for the given text, and will be therefore given a higher score.

An important aspect of TextRank is that it does not require deep linguistic knowledge, nor domain or language specific annotated corpora, which makes it highly portable to other domains, genres, or languages.

The Shortest-path algorithm is easy to implement and should be relatively language independent, though it was only evaluated on English texts. The generated summaries, they are often somewhat "smooth" to read. This smoothness is hard to quantify objectively, though, and the extracts are by no means as smooth as a manually written summary.

When it comes to including the important facts from the original text, the weighting of sentences using traditional extraction weighting methods seems to be the most important part. Taking a path from the first to the last sentence does give a spread to the summary, making it more likely that most parts of the original text that are important will be included and making it unlikely that too much information is included from only one part of the original text.

IV. CONCLUSION

Automatic text summarization is now used synonymously that aim to generate summaries of texts. This area of NLP research is becoming more common in the web and digital library environment. In simple summarization systems, parts of text – sentences or paragraphs – are selected automatically based on some linguistic and/or statistical criteria to produce the abstract or summary.

Shortest-path algorithm is better because it generates smooth summaries as compared to ranking algorithms. Taking a path from the first to the last sentence does give a spread to the summary, making it more likely that most parts of the original text that are important will be included.

V. REFERENCES

[1] Satyajeet Raje, Sanket Tulangekar, Rajshekhar Waghe, Rohit Pathak, Parikshit Mahalle, "Extraction of Key Phrases from Document using Statistical and Linguistic analysis", 2009.

[2] Md. Nizam Uddin, Shakil Akter Khan, "A Study on Text Summarization Techniques and Implement Few of Them for Bangla Language", 1-4244-1551-9/07IEEE, 2007.

[3] Jonas Sj¨obergh, Kenji Araki, "Extraction based summarization using a shortest path algorithm", Proceedings of the Annual Meeting of the Association for Natural Language Processing, 2006.

[4] Massih R. Amini, Nicolas Usunier, and Patrick Gallinari, "Automatic Text Summarization Based onWord-Clusters and Ranking Algorithms", D.E. Losada and J.M. Fern´andez-Luna (Eds.): ECIR 2005, LNCS 3408, pp. 142–156, 2005.

[5] Rada Mihalcea, "Graph-based Ranking Algorithms for Sentence Extraction, Applied to Text Summarization". The Companion Volume to the Proceedings of 42st Annual Meeting of the Association for Computational Linguistics, pages 170–173, Barcelona, Spain, 2004.

[6] R. Mihalcea and P. Tarau, "TextRank – bringing order into texts", 2004.

[7] R. Mihalcea, P. Tarau, and E. Figa, "PageRank on semantic networks, with application to word sense disambiguation". In Proceedings of the 20st International Conference on Computational Linguistics, Geneva, Switzerland, August 2004.

[8] C.Y. Lin and E.H. Hovy, "The potential and limitations of sentence extraction for summarization". In Proceedings of the HLT/NAACL Workshop on Automatic Summarization, Edmonton, Canada, May 2003.

[9] Chin-Yew Lin and Eduard Hovy, "Automatic Evaluation of Summaries Using N-gram Cooccurrence Statistics". In Udo Hahn and Donna Harman, editors, Proceedings of the 2003 Human Language Technology Conference

[10] P.J. Herings, G. van der Laan, and D. Talman, "Measuring the power of nodes in digraphs". Technical report, Tinbergen Institute, 2001.

[11] S. Brin and L. Page, "The anatomy of a large-scale hypertextual Web search engine". Computer Networks and ISDN Systems 1998.