Graph Summarization Using Word Correlation Analysis on Large Set of Documents

Putu Y. Kusmawan[1(✉)] and Joonho Kwon[2]

[1] Department of Electrical and Computer Engineering, Pusan National University, Busan, South Korea
[2] Institute of Logistic Information and Technology, Pusan National University, Busan, South Korea
{putuyuwono,jhkwon}@pusan.ac.kr

Abstract. As there are a lot of available documents in the Internet, it is impossible to manually extract their important information. In this paper, we propose a system for extracting important information automatically from huge volume of documents using word correlation analysis. Our system analyzes words' occurrence and co-occurrence frequencies on several levels: sentence, paragraph, and document. And then, it performs three different analysis steps: occurrence frequency, adjacent correlation, and importance score analysis, to calculate the importance score of each word. Finally, it can extract keywords and store them in a graph structure. The benefits of using a graph structure were twofold. We could effectively manage the keywords and their connections; and it assisted us with the retrieval of relevant documents. Our preliminary experiment shows that our technique can be used for analyzing large set of documents well.

Keywords: Word correlation analysis · Large set of document · Graph summary

1 Introduction

In this digital era, people might want to grab important information from documents quickly, but it is difficult to be done since there are too many documents to read. This problem is often called information overload. For an instance, in digital forensic field, investigators are facing difficulties in finding suspicious relationships among individuals during a crime scene investigation [1] as there are so many documents or files to be analyzed. The similar problem occurs in a domain of online news broadcasting. Because there exist enormous amount of rapidly updated information, consumers do not have much time to read all of them. Thus, several online news aggregators [2–4] help consumers staying in touch with the latest news easily. However, they ignore connections among information.

W.-S. Han et al. (Eds.): DASFAA 2014, LNCS 8505, pp. 61–74, 2014.
DOI: 10.1007/978-3-662-43984-5_5,

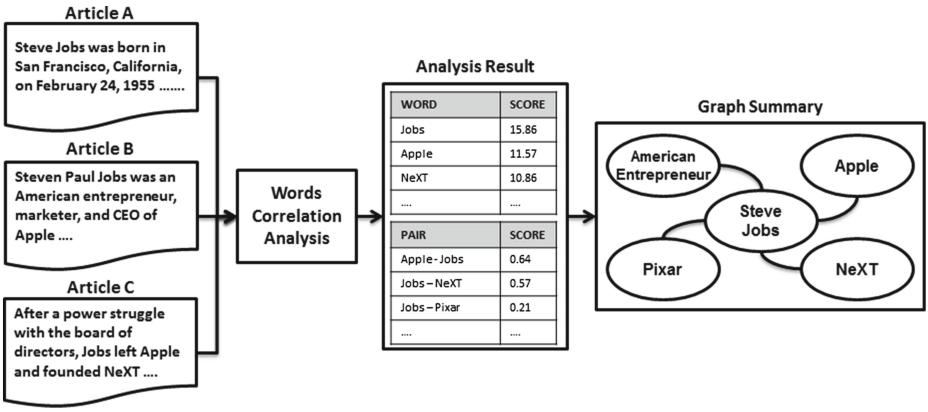


Fig. 1. Example of correlation analysis

Example 1. Let us describe the necessity of extracting information by an example. Assume that there are several articles about Steve Jobs. An article A tells much more about his childhood life, and less about his career. Another article B tells about the time when he founded a company called Apple. The others mention his past activities in several companies such as NeXT and Pixar which he had founded and had worked for. After reading all of those articles one by one, readers might discover that Steve Jobs is very related to the most successful American Entrepreneur.

It will be great, if we can build a system that can analyze the relationships among information scattered across different sources. Figure 1 illustrates the system of graph summarization using word correlation analysis on several articles about Steve Jobs. As we can see in the graph summary, a vertex labeled Steve Jobs is connected to several vertices labeled as Apple, NeXT, and Pixar. This helps us to understand why he is one of the most famous "American Entrepreneur".

An automatic document analysis has been an interesting topic as several researchers conduct their work [5–7] in this field. However, to our knowledge, none of them try to analyze a large collection of documents. Most of them only consider how their approach can extract keywords or summarize documents well by evaluating their approach on a small number of documents. Wartena and Matsuo [5,6] try to extract keywords from short texts. During an extraction process, they consider connections between words by checking a single level of co-occurrence. However, when the size of document grows very large, analyzing a single level of co-occurrence is not enough since there might be long distance correlation between words. Hu [7] breaks down an input text into different types of levels, and assigns a fixed weighting value for each of them. However, if we analyze large set of documents, the number of sentences and paragraph within documents may vary. This will impact the distribution of word co-occurrence

on each level, thus we should assign a proper weighting value for each level of documents regarding to its portion.

In this paper, we introduce a new technique for extracting meaningful relations among words to produce keywords as well as creating graph summary from large set of documents. Our contributions are summarized as follows:

- We propose a technique to create a graph summary by analyzing word correlation on three different levels: sentence, paragraph, and document.
- We assign a dynamic weighting value to every level by considering its portion. For example, we divide the number of sentences by the total number of levels to obtain the weighting value of sentence level. These weighting values might also be configured based on experimental evaluation.
- We describe the usage of the resulting graph summary for storing important information and retrieving relevant documents.
- We implement our technique in Map-Reduce framework and conduct a preliminary evaluation that shows the performance of our technique in analyzing large set of documents.

The rest of this paper is organized as follows: Sect. 2 presents the overview of our system. Section 3 explains about how we can compute the importance score of words using our technique. Section 4 describes about the process of storing the analysis result into a graph structure and how we can take advantage of the resulting graph structure in retrieving relevant documents. To evaluate the performance of our method, we have conducted a preliminary experiment and describe it in Sect. 5. Finally, we conclude our work in Sect. 6.

2 System Overview

In this section, we shall explain the architectural overview of our system as shown in Fig. 2. Our system consists of five modules: a preprocessing module, three correlation analysis modules implemented by using Map-Reduce framework, and a graph construction module. We briefly explain the functionalities of five modules here and describe the details in the next section.

- **Document Preprocessor** (PRE): This module takes input documents and creates level maps by parsing them into sentences, paragraphs and documents. It also removes stop words such as the, is, are, at, which, in, and on.
- **Occurrence Frequency Analyzer** (OFA): This module reads the preprocessed documents and checks the occurrence frequency of each word on each level. The output will be a list of words containing their set of frequencies.
- **Adjacent Correlation Analyzer** (ACA): This module analyzes the strength of connections among words by using the preprocessed content and the level statistics. The result of this analysis step is a list of pairs with their correlation scores.
- **Importance Score Analyzer** (ISA): This module combines the result of two previous steps to calculate the importance scores of all words. The result of step is a list of words with their importance scores.

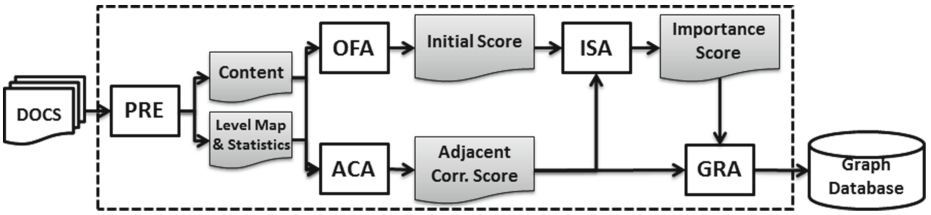


Fig. 2. Architectural overview

– **Graph Constructor** (GRA): This module takes the analysis results and stores them in a graph structure. The output of ISA will be stored as vertices and the output of ACA will be stored as edges. In addition, we also perform an entity resolution process to identify people, companies, organizations, cities, geographic features, and other types of entities.

3 Importance Score Analysis

In this section, we shall describe the detailed steps of calculating the importance score of all words. We divide and explain all of them in several subsections.

3.1 Document Preprocessing

As there might be a lot of meaningless words and irrelevant characters, we cannot directly perform our analysis steps on the input documents. We should perform several perprocessing steps to prepare the documents before being analyzed. There are several tasks that must be done in this step:

– Extracting sentence from input documents.
– Eliminating stop words.
– Assigning document ID, paragraph ID, and sentence ID (Level Map) to each sentence
– Checking the number of each level (Level Statistics).
– Partitioning documents into several blocks which are equal before loading into HDFS.

The outputs of this step are level maps and statistics as well as preprocessed documents' content. Due to the tiny size of a single document, we concatenate and partition the preprocessed documents into several equal-sized files to fit HDFS block size.

Example 2. For providing a running example, we use two documents (D1 and D2) about Steve Jobs' biography [8,9] in which each document has four sentences and two paragraphs. Figure 3 illustrates an example of document preprocessing. We use an existing sentence parser [10] to extract sentences and paragraphs. For assigning level map, we use an incremental integer number on each level. By doing so, we can also obtain the level statistics of the input documents.

DOC	ORIGINAL CONTENT
D1	Steve Jobs was an American entrepreneur, marketer, and Inventor, who was the co-founder, chairman... After a power struggle with the board of directors in 1985 ...
D2	Steven Paul Jobs was born on February 24, 1955, in San Francisco, California... As an infant, Steven was adopted by Clara and Paul Jobs and named Steven Paul Jobs...

LEVEL MAP			PREPROCESSED CONTENT
DOC	PAR	SEN	
D1	P1	S1	Steve Jobs American entrepreneur marketer Inventor co-founder chairman CEO Apple Inc.
D1	P1	S2	Apple, Jobs widely recognized charismatic pioneer personal computer revolution.
D1	P2	S3	power struggle Board directors 1985, Jobs left Apple founded NeXT.
D1	P2	S4	1986 Jobs acquired computer graphics division Lucasfilm spun off Pixar.
D2	P3	S5	Steven Paul Jobs born February 24 1955 San Francisco California February 24 1955.
D2	P3	S6	Smart directionless Jobs experimented different pursuits starting Apple Computers Steve Wozniak
D2	P4	S7	Infant Steven adopted Clara Paul Jobs named Steven Paul Jobs.
D2	P4	S8	Clara worked accountant Paul Coast Guard veteran machinist.

LEVEL STATISTICS		
2	4	8

Fig. 3. Example of document preprocessing step

3.2 Occurrence Frequency Analysis

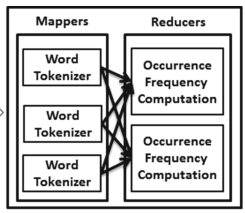
As an initial score, we analyze words occurrence frequency on each level: sentence, paragraph, and document and implement it in Map-Reduce framework. First, mappers read the preprocessed input documents to emit word as key, and level map as value. And then, reducers summarize the frequency of each word regarding its occurrence location. The final output is a list of word including its occurrence frequency on each level and the overall frequency summation. Figure 4 shows an example output of analyzing word occurrence frequency.

3.3 Adjacent Correlation Analysis

Next, we analyze the strength of connection between two words by calculating adjacent correlation scores. To calculate these scores, we need to check the co-occurrence frequency and location of every pair of words. Then, we compute the correlation scores level by level, formalized as Level-specific correlation (LC) analysis. For performing this step, we have observed the following characteristics:

1. Important pair of words must appear at least once in the same sentence.
2. Words may not stand alone to deliver information, they must work together to form at least a sentence.
3. After forming a sentence, words can form larger structure called paragraph and document to deliver more complete information.

LEVEL MAP			PREPROCESSED CONTENT
DOC	PAR	SEN	
D1	P1	S1	Steve Jobs American entrepreneur ...
D1	P1	S2	Apple, Jobs widely recognized charismatic ...
D1	P2	S3	power struggle Board directors 1985 ...
D2	P3	S5	Steven Paul Jobs born February 24 1955 ...
D2	P3	S6	Smart directionless Jobs experimented ...
D2	P4	S7	Infant Steven adopted Clara Paul Jobs ...
D2	P4	S8	Clara worked accountant Paul Coast Guard ...



WORD	SET OF FREQUENCIES			ALL FREQ.
	SEN	PAR	DOC	
Jobs	7	4	2	13
Apple	4	3	2	9
NeXT	1	1	1	3
Pixar	1	1	1	3
American	1	1	1	3
....	....	....	....	....

Fig. 4. Example of occurrence frequency analysis

Table 1. Example of calculating adjacent correlation score

Pair	Level-specific Correlation (LC)			Adjacent correlation
	Sentence (N=8)	Paragraph N=4	Document N=2	
Jobs-Apple	Co-occur: 4 LC: 4/8 = 0.50	Co-occur: 3 LC: 3/4 = 0.75	Co-occur: 2 LC: 2/2 = 1	0.64
Jobs-NeXT	Co-occur: 1 LC: 1/8 = 0.125	Co-occur: 1 LC: 1/4 = 0.25	Co-occur: 1 LC: 1/2 = 0.50	0.21
Jobs-Pixar	Co-occur: 1 LC: 1/8 = 0.125	Co-occur: 1 LC: 1/4 = 0.25	Co-occur: 1 LC: 1/2 = 0.50	0.21
Jobs-American	Co-occur: 1 LC: 1/8 = 0.125	Co-occur: 1 LC: 1/4 = 0.25	Co-occur: 1 LC: 1/2 = 0.50	0.21

Example 3. Let us consider again the preprocessed content in Fig. 3. The words, "Jobs" and "Apple", co-occur in four sentences, and there are eight sentences in the whole input documents. We can calculate sentence-level correlation score as follows: 4/8 = 0.5. We perform the same operation for the upper level of co-occurrence, until we can get three different level-specific correlation scores (sentence, paragraph, and document).

We then summarize all level-specific correlation scores by considering the weighting values on each level as an adjacent correlation (AC) score. The AC score between two words (*a,b*) is formalized as follow:

$$AC_{a,b} = \sum_{x=s}^{d} \alpha_x . LC_{a,b}^{x} \tag{1}$$

where $LC_{a,b}^{x}$ is the level-specific correlation of word a and b, α_x is the weighting value for each level, and x is a set of levels containing sentence(s), paragraph(p), and document(d). To give a proper weighting value, we use the level statistics obtained from the document preprocessing step.

Example 4. Consider again the level statistics in Fig. 3. There are 8 sentences, 4 paragraphs, and 2 documents which produces 14 total levels of our dataset.

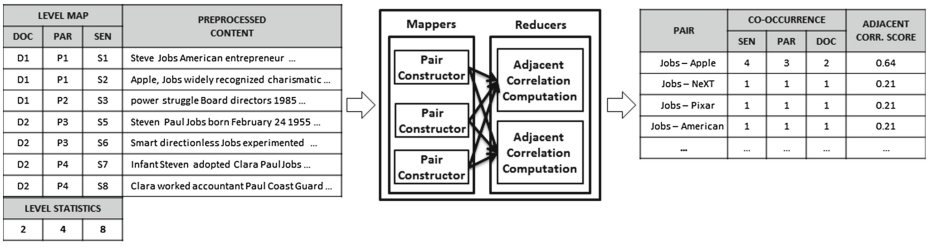


Fig. 5. Overview of adjacent correlation analysis in map-reduce

We will divide the number of corresponding level by the total number of levels in the whole documents. Therefore, we can calculate sentence level weighting value as follows: $8/14 = 0.57$. And then, we can calculate the adjacent correlation score for "Jobs-Apple": $(0.57 \times 0.50) + (0.29 \times 0.75) + (0.14 \times 1) = 0.64$. Table 1 shows the example of calculating adjacent correlation scores.

The higher adjacent correlation score means the stronger connections between two words. As we can see in Table 1, the pair "Jobs-Apple" has higher score than the pairs of "Jobs-NeXT" and "Jobs-Pixar". Thus, we can conclude that based on our dataset "Jobs" is more related to a company named "Apple" than "NeXT" or "Pixar". In addition, we can also use this value to prevent storing irrelevant pairs.

We also have implemented this analysis step in Map-Reduce to make it scalable. Mappers will read the preprocessed content as well as the level map to construct all pairs. After map task has finished, it will emit the pair of two words (separated by comma) as key and level map as value. Reducers will retrieve the intermediate result and check the level map to calculate word co-occurrence on each level. To compute the adjacent correlation score, they utilize the level statistics that has been loaded into Hadoop distributed cache before the job starts. As the result, reducers will emit the pair as key and its adjacent correlation score as value. Figure 5 shows the processing flow to calculate the adjacent correlation scores in Map-Reduce framework.

3.4 Importance Score Analysis

After performing occurrence frequency and adjacent correlation analysis, we can calculate the importance score of each word. There will be two mapper classes for reading two different data sources which are the result of the previous analysis steps, OFA and ACA. The first mapper class will read the result of OFA, then emit each word as key and its occurrence frequency as value with a character 'F' as prefix. The second mapper class will read each pair of words from ACA's output and split them into words. Then, it emits each word as key and emit its adjacent correlation score as well as its co-occurring word as value.

After map tasks have finished, reducers will begin their work by collecting all words occurrence frequencies and adjacent correlation scores. Prior to calculating the importance score, they will perform one additional step called transitive correlation analysis. This step is needed to measure the impact score that is shared between two co-occurring words. The impact must be proportional to the adjacent correlation score and the occurrence frequency difference between them. The bigger transitive correlation score a pair has, the bigger supporting score they share.

Example 5. Recall to the result of the two previous steps in Fig. 4 and Table 1, we can obtain the overall frequency difference between "Jobs" and "Apple" as follow $13 - 9 = 4$, and we have also computed the adjacent correlation score of "Jobs-Apple" which is 0.64. Thus, we can calculate the transitive correlation score of "Jobs-Apple" as follow $4 \times 0.64 = 2.57$.

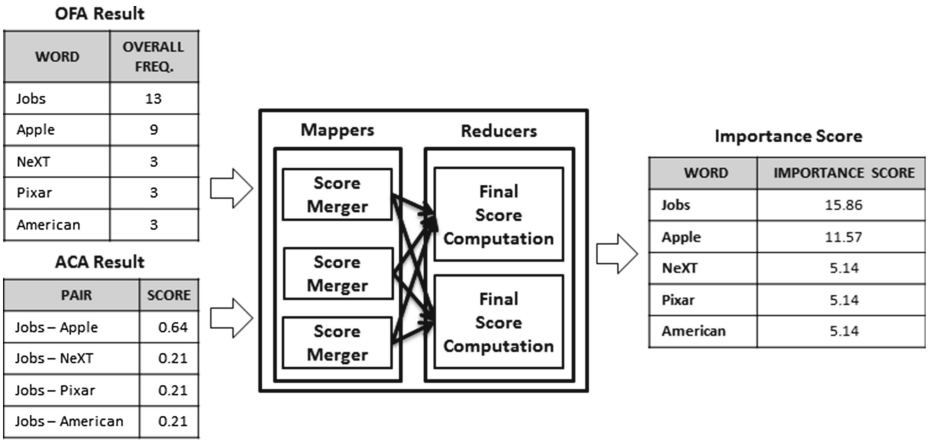


Fig. 6. Overview of importance score analysis in map-reduce

After calculating the transitive correlation of all pairs, reducers will calculate the importance score by adding the overall occurrence frequency of each word with the maximum transitive score (MAX(tra)) that it has. Figure 6 shows the Map-Reduce processing flow to calculate the importance scores.

Example 6. Consider again the result of OFA in Fig. 4, the overall occurrence frequency of "Jobs" is 13, and it is connected to several vertices via four different edges representing their transitive correlation scores. Thus, we can calculate the importance score of "Jobs" by adding its frequency to its maximum Transitive Correlation score: 13 + MAX(2.57, 2.14, 2.14, 2.14) = 15.57.

As the result of this step, reducers will emit word as key and its importance score as value. The higher importance score, the more likely we can regard a word as a good candidate for keyword. To make the process easier to understand, we describe this mechanism using a graph structure in Fig. 7.

4 Graph Structure

4.1 Graph Structure Construction

After performing all of the analysis steps, we construct a graph structure using the analysis results. As we know, a graph mainly consists of two main components: vertex and edge. Both vertex and edge may have several properties to store more detailed information. Figure 8 depicts the resulting graph structure that can be built by our system. There are 5 vertices in the graph, each of them has several properties. For an instance, a vertex labeled "Steve Jobs" stores the importance score, occurrence frequency and location. The edges also have several properties to store detailed information of all pairs of words.

We use the result of ISA and ACA to construct the graph structure. First we read all of the ISA's output and store them as vertices. And then, we create

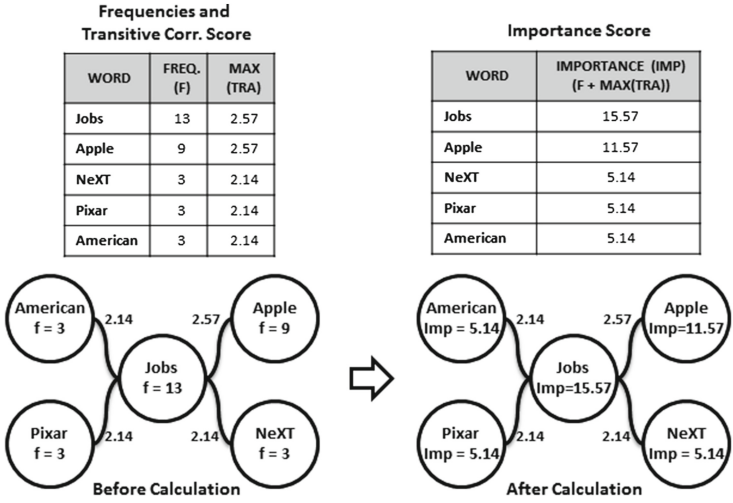


Fig. 7. Example of importance score calculation

edges using the output of ACA. After all vertices and edges have been stored in the graph, we perform an entity resolution using an existing NER library [11] to obtain the complete term form each vertex label.

4.2 The Usability of Graph Structure

In this subsection, we will discuss about the usage of the resulting graph structure produced by our system. By using the graph structure, we can manage all keywords and their connections well. In addition, it might also be useful for relevant document retrieval system.

Example 7. For this occasion, we will use the example documents that we have mentioned in Fig. 3. Suppose user wants to retrieve all documents which contain any information about "Jobs" and "Apple". If we represent keywords as a bag-of-words, we might be able to return all documents which contain both "Jobs" and

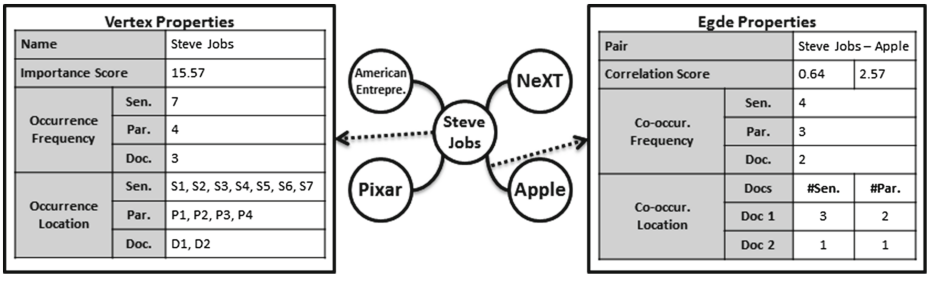


Fig. 8. Example graph structure

"Apple". However, such kind of representation does not store the connectivity between keywords. It will be difficult for us to rank the query answer based on the relevance of document's content. Finally, user must decide manually which document to read first. Using our approach, we can tackle this issue by analyzing the connection between "Jobs" and "Apple" in more detail. Even though they co-occur in both documents, the co-occurrence distribution in lower level is different. Doc.1 has more sentences mentioning about "Jobs" and "Apple" than Doc.2 does, thus we should recommend user to read Doc.1 prior to reading Doc.2. Thus, our system can quickly answer user's query using the following steps:

1. Locate "Jobs" vertex.
2. Find an edge that connects "Jobs" vertex with "Apple" vertex.
3. Check in which document they co-occur by analyzing the corresponding edge's properties.
4. Rank the result according to the number of sentence and paragraph co-occurrence on each document.

Finally, our system will returns Doc.1 at the top of the query answer.

We may also employ indexing system on the edge structure to improve the query performance. However such improvement approach will have a trade-off since it will also increase the complexity of graph construction.

5 Evaluation

In this section, we present experimental result to evaluate the accuracy and the performance of our system.

5.1 Experimental Environments

Hardware. We used two different hardware setups for evaluating our system. First, we used a single machine (Intel Core2Quad Q6600 @2.4 GHz, 64-bit, 4 GB RAM) running on Windows Server 2008 R2 Standard 64-bit for evaluating the accuracy of our system. Second, we used 15 commodity machines (Intel Core2Quad @2.66 GHz, 64-bit, 2 GB RAM, 500 GB HDD) running on Ubuntu 12.10 and configured them to work together on top of Hadoop version 1.2.1 for evaluating the performance of our system.

Dataset. We used two data sets: (1) IEEE dataset for an accuracy test and (2) digital books [12] for a performance test. The IEEE dataset consists of several papers which are randomly selected from IEEE Explore website. The overall statistics of our first dataset are as follows: 90,990 words, 20,511 sentences, 15,495 paragraphs, and 20 documents. The digital book dataset consists of 9,487,087 sentences, 2,873,700 paragraphs, and 2,620 documents. In total, the size of the digital book dataset is 1.1 GB with the average size of 2.7 MB.

5.2 Accuracy Test

In this evaluation, we tried to extract several important information in the form of keywords from several scientific paper documents. We combined the author's keyword and manually-selected keyword as the basis of forming relevant keyword list for each paper. Prior to this experiment, we broke down relevant keywords into words and eliminated duplicate words. As the result, each paper has at least 5 distinct words and at most 18 distinct words, with an average of 10 distinct words. Then, we extract keywords using our method and compute the precision-recall average score. As we know, precision and recall formula are defined as follows:

$$Precision(P) = \frac{CorrectWord}{ExtractedWord}; Recall(R) = \frac{CorrectWord}{RelevantKeyword} \quad (2)$$

For the first attempt, we want to extract as many correct keyword as possible; thus, we select Top-25 most important words from all documents. In this case, we manage to get a good recall score ($R = 0.73$), however the precision score is very low ($P = 0.29$). It makes sense since there are only 10 keywords on each paper in average. Therefore, we gradually decrease the number of extracted keyword for each document (Top-20,15,10,5) to achieve better precision score. Finally, we extract Top-5 most important words from all documents and we get $R = 0.32$ and $P = 0.63$. Figure 9 shows the result of our accuracy evaluation.

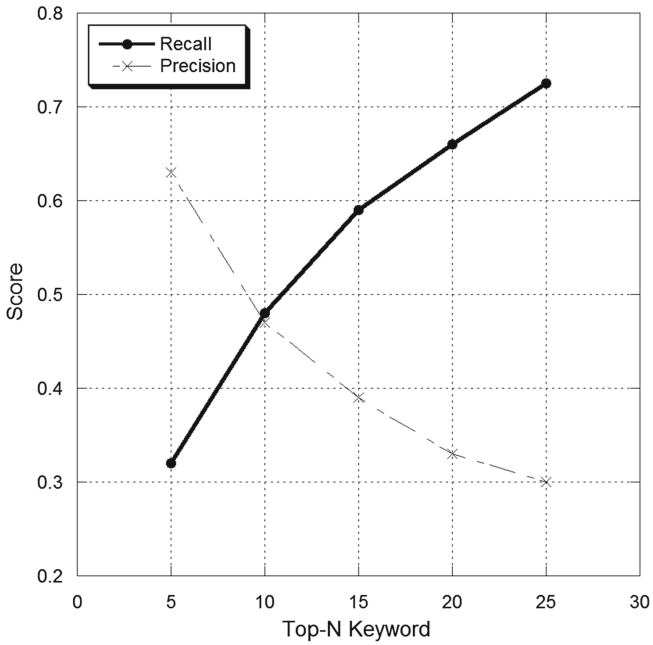


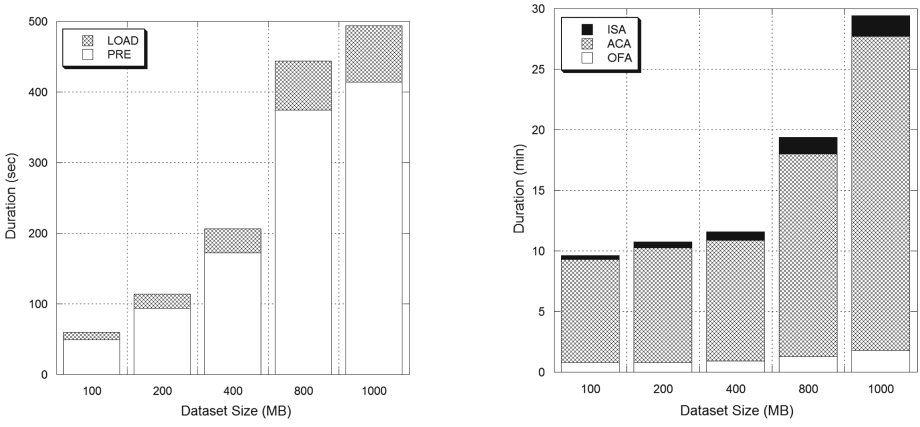
Fig. 9. Precision and recall result

5.3 Performance Test

To measure the performance of our technique, we have also conducted another experiment using the digital book dataset. We used 15 machines to process varying size of the datasets. Since the average size of a single book is only 2.7 MB, we concatenated the books to construct 50 MB input split size before running the Map-Reduce job. We used a single machine to preprocess the dataset, thus it takes several minutes to completely preprocess the dataset. Figure 10(a) shows the time needed for preprocessing the dataset(PRE) and for loading it into HDFS(LOAD).

During the first execution, our system requires around 10 min to completely analyze 100 MB of documents. When we gradually increase the dataset size up to 400 MB, our system requires considerably few more seconds to analyze them all. We get a significant performance degradation, when it comes to analyze 800 MB of documents. However, we can say that our system only requires less than a second to analyze a single book since it can finish processing 1 GB dataset in 37 min.

As shown in Fig. 10(b), the most time consuming step is adjacent correlation analysis (ACA). It consumes more than a half of the whole processing time. This is due to the very large amount of pairs that needs to be processed. For an instance, it calculates ACA score of 27 millions of pairs during the analysis of 1GB dataset. Our future works will focus on improving our technique especially in this step. It will be much faster, if we can eliminate some irrelevant pairs prior to this step.



(a) Preparation Time (b) Analysis Time

Fig. 10. Execution time

6 Conclusion

In this paper, we have proposed a system for summarizing a large set of document as a graph structure using word correlation analysis. Our system relies on the analysis of words occurrence, and co-occurrence statistics on each level of documents. First, it performs the occurrence frequency analysis to calculate word's initial importance score. Second, it analyzes the strength of connection among words by performing the adjacent correlation analysis. The adjacent correlation score is useful for calculating the impact score that a word gives or receives during the transitive correlation analysis. Then, our system performs importance score analysis to calculate the final score combining the initial importance score and the maximum transitive correlation score. Finally, a graph structure can be constructed from the importance scores and correlation scores. We have also described the usability of our graph structure for storing important information and retrieving relevant documents. Experimental results have shown that our system can give a considerably accurate result and perform well in analyzing large amount of documents.

Acknowledgement. This research was supported by the MKE(The Ministry of Knowledge Economy), Korea, under the ITRC(Information Technology Research Center) support program (NIPA-2013-(H0301-13-1012)) supervised by the NIPA(National IT Industry Promotion Agency).

References

1. Carrier, B.D.: Defining digital forensic examination and analysis tool Using abstraction layers. Int. J. Digital Evidence **1**(4) (2003). http://www.utica.edu/academic/institutes/ecii/publications/articles/A04C3F91-AFBB-FC13-4A2E0F13203BA980.pdf, DBLP, http://dblp.uni-trier.de
2. Chowdhury, S., Landoni, M.: News aggregator services: user expectations and experience. Online Inf. Rev. **30**(2), 100–115 (2006)
3. Summly: Summly news aggregator (2014). http://summly.com/
4. Inc., G.: Google news (2014). https://news.google.com/
5. Wartena, C., Brussee, R., Slakhorst, W.: Keyword extraction using word co-occurrence. In: Proceedings of Seventh International Workshop on Text-based Information Retrieval, Bilbao, Spain, pp. 54–58 (2010)
6. Matsuo, Y., Ishizuka, M.: Keyword extraction from a single document using word co-occurrence statistical information. Int. J. Artif. Intell. Tools **13**(1), 157–169 (2004)
7. Hu, X., Wu, B.: Automatic keyword extraction using linguistic features. In: Proceedings of 6th ICDM Workshops, pp. 19–23 (2006)
8. Wikipedia: Steve jobs (2013). http://en.wikipedia.org/wiki/SteveJobs
9. Website, T.B.C.: Steve jobs (2014). http://www.biography.com/people/steve-jobs-9354805
10. Group, T.S.N.: Stanford corenlp (2013). http://nlp.stanford.edu/software/corenlp.shtml

11. AlchemyAPI: Entity extraction api (2013). http://www.alchemyapi.com/api/entity-extraction/
12. Project Gutenberg Organization: Free ebooks - Project Gutenberg (2013). http://www.gutenberg.org/dirs/