KeyGraph: Automatic Indexing by Co-occurrence Graph based on Building Construction Metaphor

Yukio Ohsawa, Nels E. Benson and Masahiko Yachida
Dept. Systems and Human Science,
Graduate School of Engineering Science Osaka University,
Toyonaka, Osaka 560-8531, Japan

Abstract

In this paper, we present an algorithm for extracting keywords representing the asserted main point in a document, without relying on external devices such as natural language processing tools or a document corpus. Our algorithm KeyGraph is based on the segmentation of a graph, representing the co-occurrence between terms in a document, into clusters. Each cluster corresponds to a concept on which author's idea is based, and top ranked terms by a statistic based on each term's relationship to these clusters are selected as keywords. This strategy comes from considering that a document is constructed like a building for expressing new ideas based on traditional concepts. The experimental results show that thus extracted terms match author's point quite accurately, even though KeyGraph does not use each term's average frequency in a corpus, i.e., KeyGraph is a content-sensitive, domain independent device of indexing.

1 Introduction

Advances in high-volume storage media have lead to an explosion in the amount of machine readable text available. This information appears in many forms such as electronic mails, news, libraries, and many others. The sheer quantity of text often prohibits dealing with complete documents themselves for some processing tasks, such as search and comparison of documents, due to the limited space and time considerations.

As a solution, a small set of terms (words and phrases) are employed to serve as representatives for a document. Searching, comparison, and other operations are performed on these terms, rather than the complete documents. Assigning representative terms to a document is a process called *indexing* and the terms assigned are known as *keywords*. Indexing helps in significantly reducing the human effort in sifting through vast amounts of information. The primary use of keywords is as the basis of various search techniques for information retrieval, but keywords also serve as the abstract for a document, i.e., keywords present a short summary of the information contained within a document.

Our goal is to extract keywords which works well for both purposes. That is, given a set of documents, we want to retrieve ones with content most accurately matching user's specific and unique interest, not only including terms he/she often uses than other people in the domain. For this purpose, we have to extract terms representing the author's point in the current document. For example, *camera* is not a keyword we want to extract from a document presenting an *omni-directional robot vision for looking at vertical edges all around the robot with a single lens* although *camera* may appear more frequently in the current document than in average documents in robotics domain, because *camera* does not represent the original contribution of the author. Rather, in order to summarize the originality of a document, or for investigating already presented documents overlapping a recent invention of the search engine user, it is important to extract the main and original point – use of a conical mirror for making a robot look at vertical edges all around itself, in the example above.

2 Previous Work and Our Aim

2.1 Previous work

Human experts can reliably select consistent keywords from documents by hand. Yet, it is becoming impractical to rely on human experts to choose keywords, because there are too many documents to be indexed. An effective approach for avoiding this problem may be to assign authors the task of indexing documents they wrote. However, there are documents in digital libraries, already stocked without being indexed - assigning authors the task of indexing all of their past work is not practical. In order to cope with such problems, there have been several approaches for automatic indexing of documents. Below, Let us show some previous approaches and their disadvantages.

• **Statistic indexing** The simplest method is to count the occurrences of each term in the document and select those with frequencies above a certain threshold [1]. However, some high frequency terms may have little importance to the document, while some low frequency terms may be important. Therefore, average frequencies of terms in a large-scale database [2, 3, 4] such as a corpus are used in order to exclude frequent terms in average documents (many criteria were invented as terms' importance [5]). Mutual information [6, 7] and Bayesian Network [8] were also used for obtaining typical keywords of documents

in each area, based on corpus-based document models.

However, a corpus based method is not always appropriate for our purpose of extracting the main idea, i.e. point of a document. In the example above, *camera* will be extracted if we use the corpus for the domain of robotics. If we miss *vertical edges*, by preferring *camera* or *vision*, then recall (reader not familiar with *recall* and *precision* will be referred to Section 5) will be poor for a user searching a document about generating a map of a room from visible vertical edges. Also, precision will be bad if the user wants good cameras for robot-eyes, because the document in the current example will be retrieved as relevant to the user.

• **Indexing by sections and titles** Editors of newspapers are trained to write the summary in the beginning of articles, because their aim is to be read by many people including busy ones. However, in some academic or technical papers, only previous works or the outline of the background domain appears in the introduction. That is, not every document has its own point in the beginning. Also, titles may be very abstract and meaningless - in extreme cases as in the Internet, a documents may have no title.

• **Indexing by natural language analysis** While natural language analysis may have the potential to yield the most accurate keywords for a document, the field has not yet reached the point where it can be applied to general problems. For example, a significant amount of background knowledge (syntactic, semantic, pragmatic knowledge, fonts for emphasis, and so on) is necessary [9]. This complexity leads to a large processing time requirement, and acquiring the knowledge is itself a bottleneck.

2.2 Our aim

Motivated by above problems of previous methods, we aim at automatic indexing which:

- expresses the main point of the author, not frequent terms he/she uses.
- uses only information in the text of documents, i.e. not external knowledge like a corpus, sections, or natural language processing.
- achieves a simple and fast algorithm.

3 KeyGraph: Indexing by a Co-occurrence Graph

3.1 A building construction metaphor

In this work, we deal with technical or academic documents to be indexed. Our method *KeyGraph* is based on the simple model that a document is written to carry a few original points, and that terms in that document are related for expressing these points.

Comparing a written document to a constructed building provides a useful metaphor to explain our model. This building has *foundations* (statements for preparing basic concepts), walls, doors and windows (ornamentation). But, after all, the *roofs* (main ideas in the document), without which the building's inhabitants cannot be protected against rains or sunshine, are the most important. These roofs are supported by *columns*. Simply put, KeyGraph finds the roofs, i.e. terms that hold the rest of the document together via columns. For extracting roofs, KeyGraph is composed of three major phases.

1) Extracting foundations, where basic and preparatory concepts are obtained as clusters, each of which are constructed by the co-occurrence among the terms in the document.

2) Extracting columns, where columns, i.e., the relationships between terms in the document and the basic concepts extracted in 1) are obtained.

3) Extracting roofs, where nodes (representing terms) at the cross of strong columns are regarded as $roofs$.

Making a co-occurrence graph in **1)** is a common approach for computing the relevance between terms, e.g., [10, 11]. Our new point is to regard the co-occurrence graph as the foundations of the document, on which the document is *built*. The roofs of the built document are obtained in **2)** and **3)**. In the following subsections, phases **1), 2)**, and **3)** are formally expressed. Candidates of foundations and roofs are prepared before all these phases, in the initial *document preparation phase* (see 3.2). Then, phases **1)**, **2)** and **3)** above run (see 3.3, 3.4 and 3.5 respectively).

3.2 Document preparation

A document (let us call this D) is composed of *sentences*, which are in turn composed of words.

Prior to processing D, a list of non-significant words that have little meaning is prepared, which is the only external knowledge used. This list is referred to as a *stop words list* and contains such words as "a", "and", "here", etc., similar to [9]. Along with such words to discard, our list also includes word stems to discard. Words in D are stemmed to reduce related words to the same root. For example "run", "running", and "runs" are all reduced to "run" by the method in [12].

Sequences of words bound by non-significant words an d stems become phrase candidates. For constructing phrases, each combination of multiple words in each phrase candidate is selected and added to a list, where the selected words are side by side in the phrase candidate and the length is greater than one word. For example, for words a, b, c, and d, phrase candidate $\{abcd\}$ generates phrases $\{abc\}$, $\{ab\}$, $\{bcd\}$, $\{bc\}$, and $\{cd\}$. The list of phrases is sorted by frequency and each phrase is processed in turn. If any phrase including the current phrase has a frequency equal to the current one, the current one is thrown out (there cannot be one with a greater frequency because the phrases are sorted). Otherwise, all the phrases including the current one are thrown out. This method pulls out longer phrases with higher frequency. After this *document preparation phase*, D is represented by D_{terms} including *unique* terms (i.e., each term is included once) $w_1, w_2, \ldots, w_j, \ldots, w_l$. Hereafter, a *term* means a word or a phrase in D_{terms}.

3.3 Phase 1) extracting foundations

Graph G for document D is made of nodes representing terms, and links representing the *co-occurrence* (term-pairs which frequently occur in same sentences throughout D). Nodes in G represent high-frequency terms in D, and form the candidates of foundation because terms may appear frequently for expressing typical basic concepts in the domain, i.e., frequently used in the background domain of the author. Then, subgraphs corresponding to the basic concepts are constructed. This construction is based on the idea that associations among terms in D reflect the semantic coherence or the underlying concept in a portion of D.

The idea that underlying concepts are implied by the association among terms in a document [13, 11] has been applied for segmenting a document into semantically coherent portions [14], based on the location of terms in D. However, selecting the most important portion has been tough yet. Also, few works used the co-occurrence information for indexing [15], but the search engine performance was not radically superior to classical methods as TFIDF. This means that using co-occurrences to compare queries with only local information (e.g. one sentence or one paragraph) in the document in hand is not very meaningful. Rather, the fact is that author's point is often expressed *globally* by all the sections in one document, based on local statements. From this standpoint, we use this idea to extract keywords.

In KeyGraph, associations between terms are defined by the co-occurrence among them, i.e., occurrences in the same sentences, and presented by the links in G (solid lines in Figure 1). A maximal connected subgraph, i.e. a connected graph which includes no connected proper subgraphs, in G, is regarded as a concept on which the author's idea is based. Hereafter, we say *clusters* to refer to these maximal connected subgraphs representing the *foundations*.

To embody above strategy, nodes and links in foundations are obtained as follows.

• **Nodes** Terms in D_{terms} are sorted by their frequencies in D. High frequency terms, i.e., the set of terms above the 30 th highest frequency when D_{terms} (we denote this set by HF, and 30 is determined empirically), are taken into G. Terms in HF become the nodes in G (black nodes in Figure 1).

• **Links** Nodes in HF are linked, if the association between the corresponding terms is strong. We define the association of terms, w_i and w_j in D, as

$$assoc(w_i, w_j) = \sum_{s \in D} min(|w_i|_s, |w_j|_s), \qquad (1)$$

where $|x|_s$ denotes the count of x in sentence s. This formulation comes from the assumption that each appearance of w_i is related to its nearest appearing w_j, not all w_j's in the sentence. Pairs of high-frequency terms in HF are sorted by $assoc$ and the pairs above the $(number\ of\ nodes\ in\ G) - 1$ th tightest association are represented in G by links between nodes. Here, $(number\ of\ nodes\ in\ G) - 1$ is the least number of links necessary for connecting all the nodes in G, i.e., G is singly-connected if G is a connected graph with $(number\ of\ nodes\ in\ G) - 1$ edges. Such a singly-connected graph means that the overall content of D flows without redundancy (term 3 is not derived from term 1 in two ways, i.e., directly from term 1 and through term 2 in Figure 1-a).

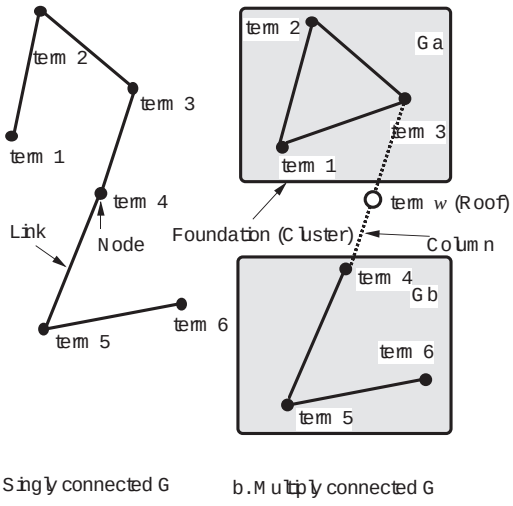

Figure 1: Extracted foundations as G_a and G_b - in b, roof (keyword) w is supported by columns (dotted lines)

If there are some redundancies, i.e., if multiple paths exist from term 1 to term 3 as in Figure 1-b, it means that the global relation between term 3 and term 4 is weaker than the local relation in cluster G_a so that segmenting G to foundations (G_a and G_b) is reasonable. In other words, $(number\ of\ nodes\ in\ G) - 1$ links is appropriate for segmenting the document to clusters.

3.4 Phase 2) extracting columns

Keywords that we want are terms that tie and hold clusters together, as depicted by the node of term w in Figure 1-b. For defining the tightness of term w's holding clusters, we assign the following value $key(w)$ to each term w in D. $key(w)$ is a real number between 0 and 1, defined as the probability of term w to appear if all the foundations in G are considered by the author, as follows.

First, let $|x|_s$ denote the count of x in sentence s. We also define $|g|_s$, the count of cluster g in sentence s, as the count of the terms in $s \cap g$. We define two auxiliary functions:

$$based(w, g) = \sum_{s \in D} |w|_s \, |g - w|_s \, . \qquad (2)$$

$$neighbors(g) = \sum_{s \in D} \sum_{w \in s} |w|_s \, |g - w|_s \, . \qquad (3)$$

where s and w represent sentences and terms in D.

$$|g-w|_s = \begin{cases} |g|_s - |w|_s & \text{if } w \in g, \\ |g|_s & \text{if } w \notin g. \end{cases} \quad (4)$$

$based(w,g)$ means how many times w appeared in D, based on the basic concept represented by cluster g. To be more specific, $based(w,g)$ denotes the co-occurrence between term w and cluster g, i.e., the count of w's appearance in the same sentences as terms in g. $neighors(g)$ means the count of terms in sentences including terms in g. The reasoning behind $|g-w|_s$ for $w \in g$ is to temporarily separate term w from cluster g that contains w.

Based on these functions, $key(w)$ denotes the conditional probability that w is used *if* the author has all the clusters (i.e., basic concepts) in mind. That is,

$$key(w) = probability(w| \bigcap_{g \subset G} g), \quad (5)$$

or equivalently

$$key(w) = probability(\bigcup_{g \subset G} (w|g)). \quad (6)$$

That is, we define $key(w)$ by

$$key(w) = 1 - \prod_{g \subset G} \left(1 - \frac{based(w,g)}{neighbors(g)}\right). \quad (7)$$

Here, division of $based(w,g)$ by the $neighbors(g)$ denotes the rate of the appearance of term w in the neighborhood of terms in g, which means the conditional probability that w is used if the author is interested in the fundamental concept of g.

Note that, in obtaining Eq.(7), we are on the assumption that basic concepts, i.e., clusters are independent of each other. This assumption corresponds to the idea that basic concepts are not derived from deeper reasoning of the author.

Sorting terms in D by keys produces a list of terms ranked by their association with cluster. The 12 top key terms are taken for *high key terms*, currently (12 is an empirically acquired number).

3.5 Phase 3) extracting roofs, i.e. keywords extraction

Next, we add all the high key terms as new nodes to G (if they are not in G yet). We do not sort terms simply by keys, because terms which represent foundations strongly related to roofs are also important for summarizing document D. These especially important foundations are of low keys but highly ranked by the sum of the strength of touching columns. The strength of the column between a high key term w_i and a high frequency term $w_j \in HF$ is expressed as

$$column(w_i, w_j) = \sum_{s \in D} min(|w_i|_s, |w_j|_s). \quad (8)$$

Columns touching w_i are sorted by $column(w_i, w_j)$, for each high key term w_i. Columns with the highest $column$ values connecting term w_i to two or more clusters are selected to create new links in G. We depict such links representing columns by dotted line (see Figure 1-b).

Finally, nodes in G are sorted by the sum of above $column$ of touching columns. Terms represented by nodes of higher values of these sums than a certain threshold are extracted as the keywords for document D. Currently, we choose 12 top terms (empirically determined, again). If more terms are equally ranked, all the equally ranked terms are extracted.

4 Example and Refinement of KeyGraph

The ideal case is a set of clusters in G where each term in a cluster (foundation) is linked to every other node in the same cluster, and no link (one solid line in Figure 1) connects different clusters. However, there are cases where a cluster is held together weakly, but not weakly enough to be separated.

For example, Figure 2 depicts obtained foundations (solid lines and their touching nodes), columns (dotted lines), and roofs (double circles) for a document. It is noteworthy that "speed", "integer programming" and "approximate method" were selected in spite of their low frequency (twice or less in the entire document of 6899 words), considering that this paper presented a new method of speeding up abduction in predicate logic by NBP [16], an inference based on approximate integer programming. However, considering that this paper is contributed for extending the NBP method from propositional to predicate logic, it is not a good result that neither "predicate" nor "time" are in double circles. This is due to weak links like the one depicted with scissors in Figure 2. That is, $based(w_0, g_0)$ can not be given if w_0 represents "predicate" and the largest cluster g_0 including "solution" also includes w_0.

In order to cope with this problem, an effective method is to prune weak links (with scissors, in Figure 2) and segment each cluster into denser clusters, for obtaining "predicate" as a term holding two clusters (the largest cluster including "solution" and the one including "predicate") together, in the example above. Looking for maximum cliques in the clusters may do for this purpose. However, this approach is time-consuming, i.e., a maximum clique problem is NP-complete. We adopted a simpler approach - insert the following step at the end of phase 1) in 3.3.

for all $e \in$ edges in G, prune e from G if there is no path in $G - e$ connecting the two ends of edge e,

In other words, a cluster is segmented into sub-clusters which, in G, were connected via one link. This step take $O(L^2)$ time where L is the number of links in G, because L links are pruned temporarily in turn, each link with path-tracing in G. It may improve the results if more links are pruned temporarily, but we do not choose that way because the time increases ($O(L^{d+1})$ time if d links are pruned temporarily). For the example in Figure 2, by this refinement, we obtained the

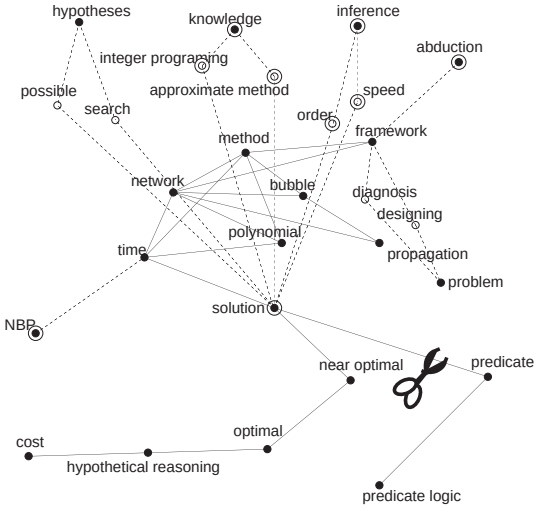

Figure 2: A weak link (marked by scissors).

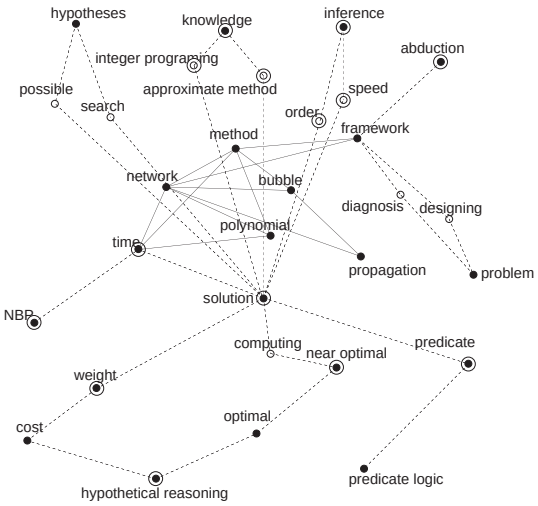

Figure 3: An improved result by further segmentation.

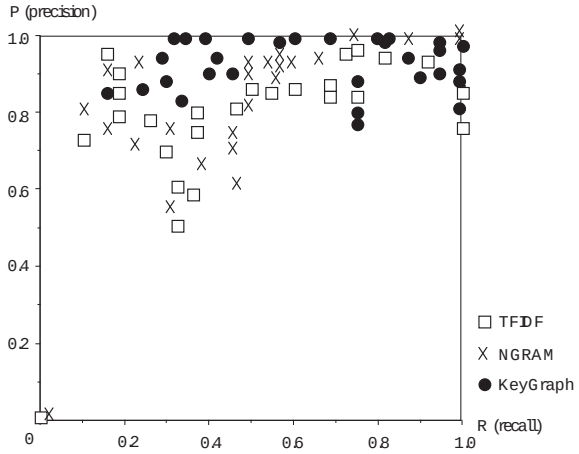

Figure 4: Precision and Recall (defined in Eq.(11)) by KeyGraph vs. other indexing methods.

result in Figure 3. Important terms "predicate" and "time" are successfully extracted.

5 Statistics of KeyGraph Performance

This section presents experimental results to show if our two goals below are achieved by KeyGraph.

(1) Obtaining keywords for author's original point for retrieving documents matching search engine user's special interest, or for summarizing documents.

(2) Efficient, fast indexing.

We conducted the following experiments by KeyGraph written in C++. The system used was Lynux on a Pentium 166MHz computer with 64M RAM.

5.1 KeyGraph's performance as an information retrieval device

In order to test the performance of KeyGraph on item **(1)**, we evaluated the documents retrieved by comparing user queries and keywords extracted from each document in the database by KeyGraph and previous methods. The values we measured are commonly-used *recall* (how many of the documents relevant to user's interest are obtained) and *precision* (how many of the retrieved documents are relevant user's interest) for each query.

Precision is defined as

$$P = \frac{Retrieved(M) \cap Relevant}{Retrieved(M)}, \tag{9}$$

and recall is commonly defined as

$$R(M) = \frac{Retrieved(M) \cap Relevant}{Relevant}, \tag{10}$$

where $Retrieved(M)$ and $Relevant$ denote the document-sets, one retrieved by indexing method M

and one relevant to user's interest, respectively. However, $R(M)$ defined in Eq.(10) can not be efficiently computed for our experiment, because the number of relevant documents cannot be determined by experts, since we deal with queries with interests unique to each user. In order to allow users try many queries from special interests, it is very inefficient to name enough number of relevant documents as *Relevant* from a large number of documents for each query.

In order to avoid this waste of time, we made a search engine whose output document list is a *shuffled union* of the retrieved documents by three indexing methods: TFIDF, NGRAM [4] and KeyGraph. We do not compare KeyGraph with thesauri-based information retrieval [10, 11], in which terms in user's query are compared with documents in a database via thesaural associations, because our test is whether or not *extracted terms from a document* represent the point or not.

TFIDF and NGRAM use a corpus, which is the document collection itself in this experiment. The difference between TFIDF and NGRAM is that TFIDF discards stems and stop-words first, while NGRAM computes the weight of all the strings (not always words) of a certain length by the same weight function as TFIDF. Users selected documents matching their interest from the list, and the system put out relevant documents selected by each indexing method, shuffled the documents, and presented a list of these documents to users. Because the number of relevant documents for each query is common to all the three indexing methods, we can compare the recall performance only by the number of relevant documents of those retrieved by each method. For this reason, we define recall as:

$$R(M) = \frac{Retrieved(M) \cap Relevant}{\bigcup_M Retrieved(M) \cap Relevant}, \qquad (11)$$

i.e., the rate of relevant documents retrieved by method M (TFIDF, NGRAM, or KeyGraph) per all the relevant documents retrieved.

Each dot in Fig. 4 shows the result. Among 5900 documents concerning artificial intelligence, 9 users in the domain of artificial intelligence got documents. We took this set of documents and users, in order to investigate if a user with *original* ideas got satisfied with the documents. In total, 130 queries were given. Fig. 4 depict the average values of precision of 5 queries. From this graph, the following tendencies are observed.

A The recall values of KeyGraph are the highest. This comes from KeyGraph's feature of obtaining important terms even if their appearance frequencies are low. That is, a document will be retrieved easily so far as the terms in user's query carry significant points in the document.

B For queries of low recall, the precision values of KeyGraph are the highest. This is because keywords extracted by KeyGraph, if few, mostly carry important ideas. On the other hand, TFIDF and NGRAM are likely to extract only terms with low significance to authors if few terms are extracted.

These tendencies were even more remarkable, in cases when users in a special domain used the search engine for satisfying users' special interests. For example, a user in the artificial intelligence domain obtained 8 documents retrieved, all of which were relevant to his query *agent communication*, while both TFIDF and NGRAM retrieved 3 relevant documents. Also, for the query *agent on the web*, KeyGraph retrieved 3 documents all relevant, while TFIDF and NGRAM retrieved none. On the other hands, for the query *data mining* which is now a popular name of a domain, by KeyGraph no documents were retrieved, while NGRAM obtained 5 relevant and 4 irrelevant documents and TFIDF did 2 relevant documents and no irrelevant ones.

Though we do not have enough space to show the detailed results, tendencies **A** and **B** were clearly observed for queries by patients for medical documents. These results mean that KeyGraph works fine for users looking for documents matching their special, unique interests or ideas, because KeyGraph extracts the asserted point in each document in the collection.

5.2 The time performance of KeyGraph

The upper bounds of the time for each phase in KeyGraph are as follows.

T_{dp} **Document preparation phase** $T_{dp} = O(W \cdot log(W))$ for sorting phrases, where W is the number of words in D.

T_{ef} **Extracting foundations phase** $T_{ef} = T_{HF} + T_{links}$, where T_{HF} and T_{links} are time for obtaining HF and links in G, respectively. T_{HF} and T_{links} are evaluated as $O(|D_{terms}| \cdot log|D_{terms}|)$ and $O(|HF|^2 \cdot log|HF|)$ respectively, considering the sorting operation of terms in D_{terms} and term-pairs in HF. Pruning weak links, as stated in section 4, can be done within the time of $O(L^2)$. Since L is bound to constant by the size of HF, T_{ef} can be evaluated to be $O(|D_{terms}| \cdot log|D_{terms}|)$.

T_{ec} **Extracting columns and roofs** T_{ec} is the time for sorting $O(|D_{terms}| \cdot |clusters\ in\ G|)$ columns. Since $|clusters\ in\ G|$ is bounded to constant value, T_{ec} is not of higher order than T_{ef}.

Conclusionally, $T_{dp} + T_{ef} + T_{ec}$ is within the oder of $O(W \cdot log(W) + |D_{terms}| \cdot log|D_{terms}|)$, not much higher than TFIDF ($|D_{terms}| \cdot log|D_{terms}|$, mainly for sorting terms) which is one of the simplest and fastest (much faster than NGRAM) previous methods. Actually, the time increased near-linearly with W (within 1 sec for 7000 to 8000 words in average, where it took 0.1 sec for a 625 words document).

Furthermore, a corpus, if employed, have to be learned with the progress in the domain, leading to radical changes in frequent words in the domain. TFIDF, NGRAM and other corpus-based indexing in

general have to revise the corpus when new documents occur. Accordingly, keywords of all the documents already indexed have to be revised - an enormous loss of time especially for searching from articles in very new and progressing domains or from collections of e-mails. On the other hand, KeyGraph does not have to revise corpus, because KeyGraph does not use one.

6 Conclusion

We presented an algorithm KeyGraph, to extract keywords representing a summary of the original point of a document. This indexing is based on the information within the document such as term frequency and term location, not on corpus which should be domain-specific, so that KeyGraph is domain independent.

Indexing based on building-construction metaphor is new and effective. The greatest benefit is the extraction (dismissal) of low (high) frequency words which carry (do not carry) the effect of the document, i.e., concepts the author is presenting. This merit leads to the satisfaction of search engine users with unique interests or ideas. Also, this method is simple and fast, and do not waste much time for the heavy revision of corpus and keywords already indexed documents. Finally, such indexing without corpus is domain-independent, in a more strong sense than [4], which make KeyGraph help searching users whose idea are quite new, not belonging to any traditional areas.

Acknowledgments

The core idea of this work started from Y.Ohsawa's private talk with Associate Professor Seiji Yamada of the Tokyo Institute of Technology, and the progress of this study has been supported by the Inamori Foundation. We also thank Professor Tadahiro Kitahashi, Associate Professor Takashi Washio in the Institute of Scientific and Industrial Research of Osaka University, for all the discussions concerning KeyGraph.

References

[1] H.P. Luhn, A Statistical Approach to the Mechanized Encoding and Searching of Literary Information, *IBM J. Research and Development* Vol.1, No.4, 309-317, 1957.

[2] G.Salton, C.S. Yang, On the Specification of Term Values in Automatic Indexing, *J.Documentation*, Vol.29, No.4, 351-372, 1973.

[3] Salton, G. and McGrill, M.J., Introduction to Modern Information Retrieval, McGraw-Hill, Inc, 1983.

[4] Cohen, J. Highlights: Language- and Domain-Independent Automatic Indexing Terms for Abstracting. *J. American Society for Information Science*, Vol.46, No.3, 162-174, 1995.

[5] Salton, G. and Buckley, C., Term-Weighting Approaches in Automatic Text Retrieval, *Information Processing and Management* Vol.14, No.5, 513-523, 1988.

[6] Quinlan, J.R., C4.5: Programs for Machine Learning, Morgan Kaufmann, 1993.

[7] Armstrong, R., et al, WebWatcher: A Learning Apprentice for the World Wide Web, *AAAI Spring Symposium Series on Information Gathering from Distributed, Heterogeneous Environments*, 1995.

[8] Tzeras,K. and Hartmann,S.: Automatic Indexing Based on Bayesian Inference Networks, *Proc. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval, Inference Networks*, 22-34, 1993.

[9] Swaminathan,K., Tau: A domain-independent approach to information extraction from natural language documents. *DARPA workshop on document management*, Palo Alto, 1993.

[10] Chen, H, et al, A Parallel Computing Approach to Creating Engineering Concept Spaces for Semantic Retrieval: The Illinois Digital Library Initiative Project, *IEEE Trans. Pattern Analysis and Machine Intelligence* (PAMI) Vol.18, No.8, 771-782, 1996.

[11] Dumais, S.T., Latent semantic indexing (LSI) and TREC-2. *Proc. Text Retrieval Conference (TREC-2)*, 105-115, 1994.

[12] Porter, M.F., An Algorithm for Suffix stripping, *Automated Library and Information Systems* Vol.14, No.3, 130-137, 1980.

[13] Morris,J. and Hirst, G., Lexical cohesion computed by thesaural relations as an indicator of the structure of text, *Computational Linguistics*, Vol.17, No.1, 21-48, 1991.

[14] Marti, A. Hearst, Multi-paragraph segmentation of expository text, *Proc. Annual Meeting of the Assoc. Computational Linguistics*, 9-16, 1994.

[15] The Use of Term Position Devices in Ranked Output Experiments, *J. Documentation*, Vol.47, No.1, 1-22, 1991.

[16] Ohsawa,Y. and Ishizuka,M., Networked Bubble Propagation: A Polynomial-time Predicate-logic Hypothetical Reasoning by Networked Bubble Propagation Method, *Advances in Artificial Intelligence* LNAI 1081, 375-387, 1996.