Key Sentence Extraction from Single Document based on Triangle Analysis in Dependency Graph

Yanting LI Graduate School of Information Science, Kyushu Sangyo University
Kai CHENG Graduate School of Information Science, Kyushu Sangyo University
chengk@is.kyusan-u.ac.jp, http://www.is.kyusan-u.ac.jp/~chengk/

Abstract— ***Document summarization is a technique aimed to automatically extract main ideas from electronic documents. In this paper, we propose a novel algorithm, called TriangleSum for key sentence extraction from single document based on graph theory. The algorithm builds a dependency graph for the underlying document based on co-occurrence relation as well as syntactic dependency relations. The nodes represent words or phrases of high frequency, and edges represent dependency, or co-occurrence relations between them. The clustering coefficient is computed from each node to measure the strength of connection between the node and its neighborhood nodes in a graph. By identifying triangles of nodes in the graph, a part of the dependency graph can be extracted as marks of key sentences. At last, a set of key sentences that represent the main document information can be extracted.***

Keywords— ***document summarization; key sentence; dependency structure analysis; clustering coefficient; triangle finding***

I. INTRODUCTION

With the fast increase of electronic documents available on the network, techniques for making efficient use of such documents become increasingly important. Document summarization is a technique aimed to extract main ideas from electronic documents so that it is easy to get gist of the underlying document. Document summarization is related to the issues of keywords or key phrases extraction, or text decomposition [4][6]. Basically there are two approaches for document summarization: extraction and abstraction. Extraction techniques merely copy the information deemed most important to the summary, while abstraction involves paraphrasing sections of the source document. In general, abstraction can condense a text more strongly than extraction, but it is harder to achieve than extraction approach.

Some of the well-known approaches to extractive document summarization utilize supervised learning algorithms that are trained on collections of “ground truth” summaries built for a relatively large number of documents. However, they cannot be adapted to new languages or domains without training on each new type of data.

Sentence extraction is a technique used for automatic summarization where statistical heuristics are used to identify the most salient sentences in a document. Sentence extraction is a cost-efficient approach compared to more knowledge-intensive approaches where additional knowledge bases such as ontology or linguistic knowledge are required. In short, sentence extraction works as a filter which allows only important sentences to pass.

In the early seminal research [1], H. P. Luhn proposed to assign more weight to sentences at the beginning of the document or a paragraph. Edmundson stressed the importance of title-words for summarization and proposed to employ stop-words list in order to filter out uninformative words of low semantic content [2]. He also distinguished the differences between bonus words and stigma words, i.e. words that probably occur together with important or unimportant information, i.e. words which occur significantly frequent in the document. With large linguistic corpora available today, the tf–idf value which originated in information retrieval, can be successfully applied to identify the keywords of a text: If for example the word "cat" occurs significantly more often in the text to be summarized (TF = "term frequency") than in the corpus (IDF means "inverse document frequency"; here the corpus is meant by "document"), then "cat" is likely to be an important word of the text; the text may in fact be a text about cats.

KeyGraph is a technique for keyword extraction from machine readable documents developed by Ohsawa [4][5]. KeyGraph is based on occurrence frequency and the co-occurrence relation between any two words in the document. The algorithm includes the following steps. Firstly, all the words with high occurrence frequency will be extracted, denoted by HighFreq. For any words in the HighFreq, their co-occurrence frequency will then be calculated. The pairs of words which have high value of co-occurrence frequency are connected by edge so that an undirected word graph is formed. In this undirected graph, the nodes represent the words, and the edges represent the relationship among these words. The last step is the calculation of co-occurrence frequency between any two words and the graph. Keywords in the document can be extracted as a fix number of words after processing.

In this paper, we propose a graph theory based novel algorithm for the task of single document summarization. Our algorithm extends the KeyGraph algorithm [4] for automatic keyword extraction in the following ways: Firstly, a dependency graph is built based on the extracted words with high frequency, and the dependency relationship between words. We introduce syntactic dependency relations among words so that key sentences instead of individual keywords can be ex-

tracted. Secondly, we extract heaviest triangles as anchor point of key sentences. Secondly, clustering coefficient is computed to measure the importance of each node. Thirdly, strongly connected components of the dependency graph will be extracted in terms of triangles based on the network's transitivity.

The goal of key sentence extraction is to identify key sentences that best summarize the main ideas of the underlying document. To do this, we employ techniques of natural language process to extract words of high occurrence frequency in the document and syntactic dependency relations as links between words. As a result, a dependency graph is obtained. Then, we do analysis on the graph mathematically to find closely connected words in the graph.

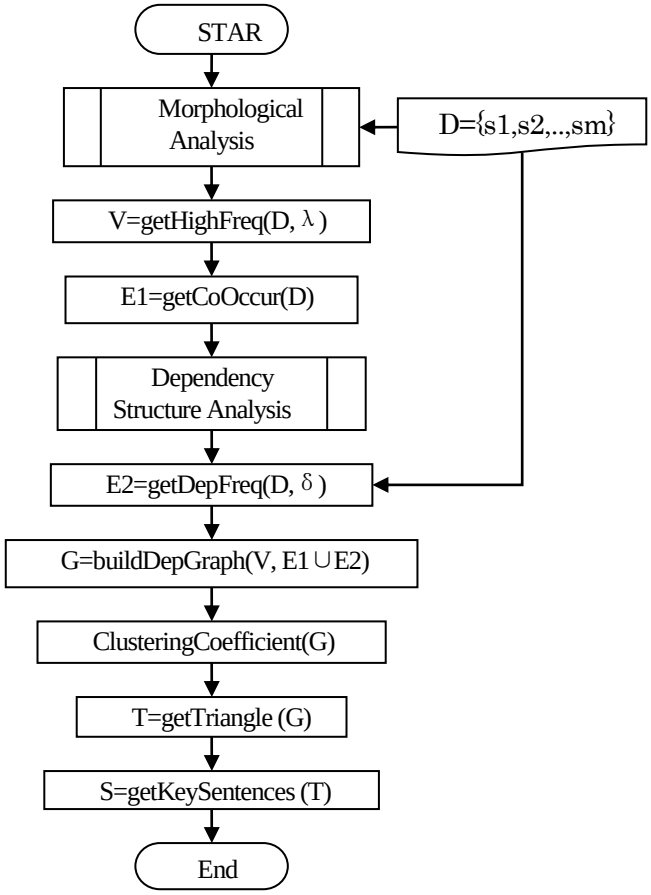

Figure 1. Framework of Key Sentence Extraction

The framework of our proposal consists of three main steps (Figure 1). Firstly, a dependency graph is built based on the extracted words with high frequency, and the dependency relationship between them. We introduce syntactic dependency relationship among words so that key sentences instead of individual keywords can be extracted. Secondly, the heaviest triangles are extracted as anchor points of key sentences. Here, a modified version of clustering coefficient to measure the importance of each vertex so that partial dependency graph will be extracted. Thirdly, strongly connected components of the partial dependency graph will be extracted in terms of triangles based on the network's transitivity.

II. PRELIMINARY DEFINITIONS

In this section, we begin with the introduction of some important definitions.

A. Co-occurrence Frequency

Co-occurrence is important indicator in linguistics where terms, stems, and concepts that co-occur more frequently tend to be related to each other. Since co-occurrence in linguistic sense can be interpreted as an indicator of semantic proximity or an idiomatic expression, it has been used in a number of techniques, such as keyword-brand associations, brand visibility across search engines, co-citation of products and services, search volume co-occurrence, positioning of documents in search results pages, keywords research and terms discovery, analysis of seasonal trends, design of thematic sites.

Co-occurrence can be global, extracted from databases, or local, extracted from individual documents or sentences; or fractal, extracted from self-similar, scaled distributions. In this dissertation, we will use the local meaning of co-occurrence which two words are said to be co-occurred if they occur in the same sentence and the distance between them is less than a given threshold, for example, less than a given threshold 1.

$$co(w_i, w_j, s_k) = \begin{cases} 1, & w_i \text{ is adjacent to } w_j \text{ in } s_k \\ 0, & \text{otherwise} \end{cases}$$

B. Dependency Frequency

A document is defined as a set of sentences, denoted by D = $\{s_1, s_2, \ldots, s_m\}$, where s_k (k=1, 2,…, m) is called a sentence of D. A word w_i is said to be *dependent on* word w_j or simply w_i *depends on* w_j in sentence s_k if w_i is *syntactically modified by* word w_j, denoted by $w_i \rightarrow w_j$. For example, in sentence "Tom sent three letters to Jim this week" , Tom→sent, sent→ letters, letters (to) →Jim.

Let $dep(w_i, w_j, s_k)$ be the indicator function of dependency relationship defined as follows:

$$dep(w_i, w_j, s_k) = \begin{cases} 1, & w_i \text{ depends on } w_j \text{ in } s_k \\ 0, & \text{otherwise} \end{cases}$$

Since in our following analysis, the direction of dependency is not important, we ignore the direction of dependency and define the un-directed dependency relation of w_i and w_j in sentence s_k in the following equation:

$$dp(w_i, w_j, s_k) = dep(w_j, w_i, s_k) \vee dep(w_i, w_j, s_k)$$

The dependency frequency between w_i and w_j in document D can be defined as below:

$$df(w_i, w_j) = \sum_{k=1}^{m} co(w_i, w_j, s_k) \vee dp(w_i, w_j, s_k)$$

C. Word Frequency

The *word* frequency or *term frequency* of a word *w* in document D is the occurrence frequency of *w* in D, denoted by *tf*(*w*). Let *Stop* be the set of stop words. Let *HighFreq* be such a set of words in D that any $w \in HighFreq$ and $w \notin Stop$ satisfies that $tf(w) > \delta$ for some $\delta > 0$.

D. Dependency Graph

A *dependency* graph of a document D is a directed graph *G* = *(V, E)*, where *V* is a set of nodes and *E* is a set of *edges*,*V* = HighFreq, $E = \{(w_i, w_j) \mid df(w_i, w_j) > \lambda$, for some $\lambda > 0\}$. In other words, G is a weighted directed graph whose nodes represent high frequency words in D and edges represent the dependency relations in between a pair of words. The dependency weight (df) of graph G=(V, E) can be denoted by the following equation:

$$df(G) = \sum_{(v_i, v_j) \in E} df(v_i, v_j)$$

E. Clustering Coefficient

Clustering coefficient (or ccf for short) is a measure of degree to which nodes in a graph tend to cluster together in graph theory. This was proposed by Watts and Strogatz in 1998[1] for analyzing the social network in real world. There are two versions of this measure: the global and the local. The global version was designed to give an overall indication of the clustering in the network, whereas the local gives an indication of the connectivity of single nodes. In this paper, we only consider local clustering coefficient.

Given an undirected graph *G* = (*V*, *E*), where V is a set of nodes, and E is a set of directed edges between nodes. The $e_{ij} = (v_i, v_j)$ is an edge between node v_i and v_j. The neighborhood $N(v_i)$ for a node v_i is defined as its immediately connected neighbors as follows:

$$N(v_i) = \{v_j \mid e_{ij} = (v_i, v_j) \in E\}$$

Let $d(v_i)$ be the degree of node *v*, *i.e.* $d(v_i) = |N(v_i)|$. The degree $d(v_i)$ of node v_i is the number of nodes adjacent to v_i. The local clustering coefficient measure for undirected graphs is defined as *the probability that a random pair of its neighbors is connected by an edge*, i.e.:

$$ccf(v_i) = \frac{|\{e_{jk} \mid v_j, v_k \in N(v_i), e_{jk} \in E\}|}{\binom{|N(v_i)|}{2}}$$

A complete subgraph of three nodes of G can be considered as a *triangle*. Let $\lambda(v_i)$ be the number of triangles including node v_i. A *triple* at a node v_i is a path of length two for which v_i is the center node. Let $\tau(v_i)$ be the number of triples on $v_i \in V$. In other words, $\tau(v_i)$ is the number of subgraphs (not necessarily induced) with 2 edges and 3 nodes, one of which is v_i and such that v_i is adjacent to both edges. We can also define the clustering coefficient of node v_i as

$$ccf(v_i) = \frac{\lambda(v_i)}{\tau(v_i)}$$

The value of clustering coefficient is a real number between 0 and 1. The maximal value is 1 when every neighbor connected to v_i is also connected to every other node within the neighborhood, and the minimal ccf is 0 if none of the nodes connected to v_i connects to each other.

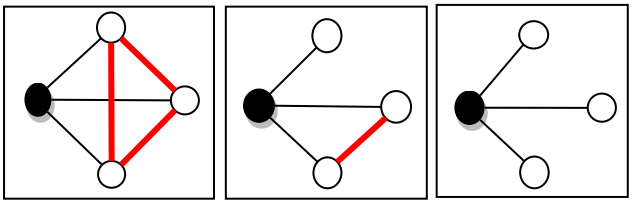

Figure 2. Examples of Clustering Coefficient

Figure 2. gives some examples for how to calculate the clustering coefficient for a single node. The degree of node v_i (dark) is 3, i.e. it has three neighbors (white). The number of edges between the 3 nodes is 3, 2, and 0 from left to tight. So the ccfs are 1, 1/3 and 0 respectively.

III. SINGLE DOCUMENT SUMMARIZATION

Now we propose algorithms for single document summarization. Given a document D and a set of stop words. To summarize D, we begin by syntax analysis of D and build a dependency graph G for D. Then compute local clustering coefficient for each node of G. Delete nodes with clustering coefficient less than a threshold and obtain graph G'. Indentify all triangle in G' whose dependency weight below a threshold. Suppose these obtained triangles form a set $T=\{T_1, T_2, T_3, T_4 \cdots\cdots T_i\}$. Indentify sentences in document D where the extracted triangles are anchored.

1. $N(v_i)$: neighborhood of node v_i
2. $d(v_i)$: degree of node v_i
3. $df(w_i, w_j)$: dependency frequency between w_i node w_j
4. $tf(w_i)$: word frequency or term frequency of word w_i

Algorithm *TriangleSum*

Input:

- *D*: documents to be summarized,
- S: a set of stop words.
- δ: threshold for high frequent words
- λ: threshold for high frequent dependency relations
- μ: threshold of clustering coefficient
- *t*: number of triangles to extract

Output:

- S: a set of key sentences

Procedure:

1）. Process *D* and construct dependency graph *G*=(*V*, *E*),where *V* = *getHighFreq*(*D*, δ) - S;

E = getDependency(D, λ);
2) . For each e ∈ E, if {e} is a cut, E=E-{e}
3) . For each v ∈ V, if d(v)=0, V=V-{v}
4) . For for v ∈ *V* compute ccf(v)
5) . If ccf(v) <μ then V=V-{v}
6) . Do breadth-first search on each connected partial graphs of G and extract all triangles T
7) . For each triangle $T_i \in T$ and compute df(T_i), dependency weight of T_i
8) . Extract *t* triangles {T_1, T_2, …, T_t} with *largest* dependency weight
9) . Identify sentences that contain at least one triangle

Figure 3. Algorithm of Document Summarization

With the extracted triangles, summarization of a document can be easily approached in different ways.

- *Entrance sentences*. Extract sentences on the entrance of the paragraphs containing more triangles. The rationale behind this approach is that bushy paths (or paths connecting highly connected paragraphs) are more likely to contain information central to the topic of the article.
- *Anchored* sentences. Extract sentences anchored with more triangles. In this approach, triangles are used to indicate important sentences.

For example, given the following document, we can construct a TriangleSum as shown in Figure 4. . Based on this graph, we can extract two triangles. Two triangle are A_1 = {海軍，空母，派遣}, A_2 = {海軍，軍事演習，行う}. Based on these triangles, we can extract the first sentence as one summarization.

米**海軍**は昨年１０月にも韓国海軍との合同**軍事演習**を**行う**ため、黄海に**空母**「ジョージ・ワシントン」を**派遣**しているが、偶発的な軍事衝突が起きる危険性の高まる中での派遣は初めてだ。

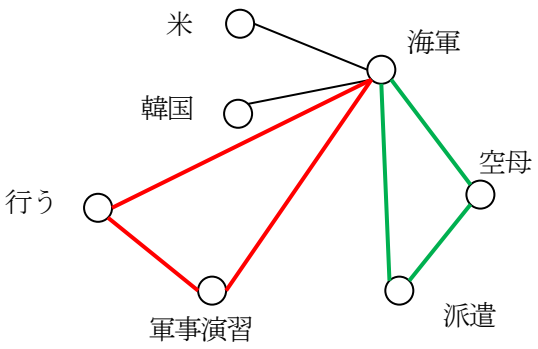

Figure 4. Example of TriangleSum

To implement the proposed algorithm, we need efficient computation of local clustering coefficient for each node in V and extraction of all triangles from each connected partial graphs. Both require efficient algorithm for triangle identification. Next we will describe a breadth-first search based triangle identification algorithm.

A triangle is represented as a triple <*u*, *v*, *w*> where u, *v*, *w* ∈ *V* and each node is adjacent to the other two. To identify a triangle, we do a breadth-first traversal for the given graph, from a starting node. Check nodes in the neighborhood to see if any of them are connected to each other.

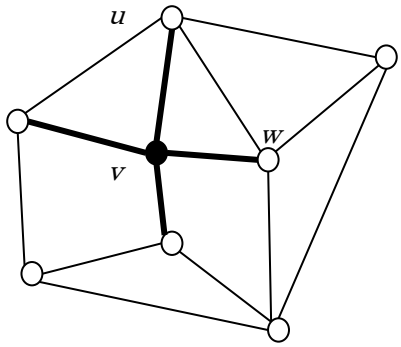

Figure 5. Example for Triangle Identification

In Figure 5. after *v* is visited, the neighborhood *N(v)* of v will enter the queue. At this time, we can check if there are any edges between them. An edge between two neighbors indicates a triangle. In this example, there is an edge between node u and w. Since both of *u* and *w* are neighbors of *v*, <*u*, *v*, *w*> is identified as a triangle.

IV. Concluding Remarks

In this paper we proposed a novel algorithm for automatic extraction of key sentences from electronic documents. The proposed algorithm works on single document without training data so that cost can be reduced. This algorithm efficiently extracts heavy triangles as anchor points of key sentences from the input document. These triangles can then be used to identify sentences that central to the topic of the document.

We have described an efficient method which does not need to build any dictionary or training data or examples before the key sentences extracted from a document to extract triangles from connected graph. When running on real world documents, we have to tune some thresholds such as λ, δ and μ.

References

[1] H. P. Luhn The Automatic Creation of Literature Abstracts. IBM Journal: pp.159-165. April 1958.

[2] H. P. Edmundson. New Methods in Automatic Extracting. Journal of the ACM 16 (2), pp. 264-285, 1969.

[3] D. J. Watts and Steven Strogatz. Collective dynamics of 'small-world' networks. Nature 393(1998): 440–442.

[4] Y. Ohsawa, N. E. Benson and M. Yachida. KeyGraph: Automatic indexing by co-occurrence graph based on building

construction metaphor, IEEE ADL'98, pp.12-18.

[5] Y. Ohsawa, Nels E. Benson and M. Yachida. KeyGraph: Automatic Indexing by Segmenting and Unifing Co-occurrence Graphs. In IEICE, Vol.J82-D-1, No.2, pp.391-400, 1999.

[6] G. Salton, J. Allan, C. Buckley, and A. Singhal. Automatic analysis, theme generation, and summarization of machine readable texts. Science 264(5164):1421-6, June 1994.

[7] J. Carbonell and J. Goldstein. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval* (SIGIR '98*)*. pp. 335-336. 1998.

[8] C.-Y. Lin, Training a selection function for extraction. In Proceedings of CIKM ’99, pp. 55–62, New York, NY, USA.

[9] D. Kawahara, S. Kurohashi and K. Hasida. Construction of a Japanese Relevance-tagged Corpus, In Proceedings of the 3rd International Conference on Language Resources and Evaluation, pp.2008-2013, 2002.

[10] Xiaojun Wang, Jianguo Xiao (2010), Exploiting Neighborhood Knowledge for Single Document Summarization and Keyphrase Extraction, ACM Transactions on Information Systems, Vol. 28, No. 2, Article 8.

[11] A. Itai and M. Rodeh. Finding a minimum circuit in a graph. SIAM Journal on Computing, 7(4):413–423, 1978.

[12] M. Latapy, Main-memory triangle computations for very large (sparse (power-law)) graphs, Theoretical Computer Science, v.407 n.1-3, p.458-473, November, 2008.

[13] D. J. Watts and Steven Strogatz. Collective dynamics of 'small-world' networks. Nature 393(1998): pp440–442.

[14] Y. Ohsawa, N. E. Benson and M. Yachida. KeyGraph: Automatic indexing by co-occurrence graph based on building construction metaphor, IEEE ADL'98, pp.12-18.

[15] G. Salton, J. Allan, C. Buckley, and A. Singhal. Automatic analysis, theme generation, and summarization of machine readable texts. Science 264(5164):1421-6, June 1994.

[16] J. Carbonell and J. Goldstein. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval* (SIGIR '98). pp. 335-336. 1998.

[17] Lin, C.-Y. (1999). Training a selection function for extraction. In Proceedings of CIKM ’99, pages 55–62, New York, NY, USA.

[18] J. M. Conroy and D. P. O’leary, Text summarization via hidden markov models. In Proceedings of SIGIR ’01, pages 406–407, New York, USA, 2001.

[19] T. Schank and D. Wagner (2004). Approximating clustering coefficient and transitivity. In Journal of Graph Algorithms and Applications, Vol. 9, pp. 265-275.

[20] B. Li, L. Zhou, S. Feng and K. Wong. A unified graph model for sentence-based opinion retrieval. In proceeding of ACL’10 of the 48th Annual Meeting of the Association for Computational Linguistics, pp. 1367-1375, 2010.