D-039

KeyFlow: Viewpoint Extraction based on Word Frequency and Dependency Relations

Yanting Li† Kai Cheng‡

1. Introduction

With the fast increase of electronic documents, techniques for making efficient use of such documents become crucial. In this paper, we consider the issue of automatic extraction of viewpoints from machine readable electronic documents. Viewpoints are main ideas or standpoints of a document that are often hidden among the long text. For example, it would be interesting to refer to the viewpoints of people from blog articles on iPad, the new tablet computer produced by Apple, so as to decide whether to buy or not. It is also important to understand viewpoints of people from different cultural background on the controversial visits of political leaders to the Yasuguni Shrine, because better understanding of each other will be a help for better cross-culture communication.

Viewpoint extraction is related to the issues of theme generation, text summarization, or text decomposition [3]. Unlike these techniques, here viewpoint extraction is aimed to extract a set of "anchor" sentences in the document. Most techniques for text summarization require training data to train the summarizer, while others require special dictionaries for semantic analysis.

In this paper, we propose a novel algorithm called KeyFlow to extract viewpoints from free text documents. KeyFlow algorithm builds a weighted directed graph for each document, where nodes represent words or phrases of high frequency in the document, and edges represent dependency relations between them. Our algorithm extends the KeyGraph algorithm [2] for automatic keyword extraction in two steps: Firstly, we introduce syntactic dependency relations among words so that viewpoint instead of individual keywords can be extracted. Secondly, we extract heaviest triangles as anchor points of sentences that better represent the viewpoints of document.

2. Preliminary Definitions

2.1 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 an indicator function of dependency relationship defined as follows:

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

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

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

2.2 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 $\delta > 0$.

2.3 Dependency Graph

A *dependency graph* of a document D is a directed graph $G = (V, E)$, where V is a set of *vertices* and E is a set of *edges*, and

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 vertices represent high frequency words in D and edges represent the dependency relations in between a pair of words.

The *dependency weight* of graph G=(V, E) is

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

2.4 Clustering Coefficient

Clustering coefficient (or CCF for short) is a measure of degree to which vertices in a graph tend to cluster together in graph theory [1]. 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 CCF.

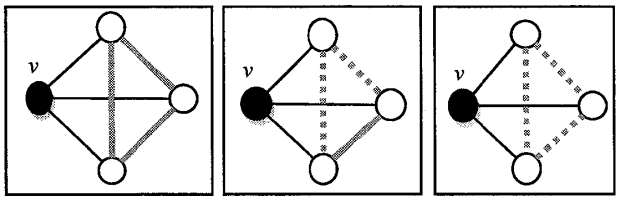

Fig. 1 Examples of local clustering coefficients

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

$$N_i = \{v_j \mid e_{ij} \in E \vee e_{ji} \in E\}$$

The CCF will be 1 if every neighbor connected to v_i is also connected to every other vertex within the neighborhood, and 0 if no vertex that is connected to v_i connects to any other vertex that is connected to v_i.

† Graduate School of Information Science, Kyushu Sangyo University

‡ Faulty of Information Science, Kyushu Sangyo University

Let k_i be the degree of vertex v_i. The local CCF measure for directed graphs can be defined as follows:

$$ccf(v_i) = \frac{|\{e_{jk}\}|}{k_i(k_i - 1)}, v_j, v_k \in N_i, e_{jk} \in E$$

Fig. 1 gives some examples for how to calculate the CCF for a single vertex. The vertex v (dark) has three neighbors (white), which could have a maximum of three connections among them. The CCF of vertex v is computed as the proportion of connections among its neighbors which are actually realized compared with the number of all possible connections.

3. KeyFlow Algorithm for Viewpoint Extraction

Now we will describe our KeyFlow algorithm for automatic view point extraction. Basically, algorithm works on single input document. After a series of process, the algorithm will output set of triangles, i.e., triple words that are strongly connected to each other. These triangles can then be used as anchors for topic sentences that represent the viewpoints.

Algorithm *KeyFlow*

Input:

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

Output:

- $\{A_1, A_2, \ldots, A_t\}$: a set of heaviest triangles

Procedure

1) . Process D and construct graph $G=(V, E)$
   V = getHighFreq(D,δ) - S;
   E = getDependency(D,λ);
2) . If deleting $e \in E$ will result in a cut, then delete it
3) . Compute *ccf(v)* for each vertex $v \in V$
4) . If $ccf(v) < \mu$ then delete v
5) . Do depth-first search on each connected partial graphs of G. Extract all triangles and compute *df*(A) for each extracted triangle A
6) . Extract t triangles $\{A_1, A_2, \ldots, A_t\}$ with *largest* dependency weights

An efficient way to extract triangles from a graph is to do depth-first search on the graph. Number each visited vertex with its depth. If a vertex at depth d has a back-edge to a vertex at depth $d - 2$, then a triangle has been found. Mark those already found triangles and continue the depth-first search until all vertices are visited. It's easy to show that if no such back-edges are found, there are no triangles.

With heaviest triangles, viewpoints can be easily extracted in different approaches.

(1) *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.

(2) *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 KeyFlow as shown in Fig. 2. 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 viewpoint.

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。オバマ大統領は２日、韓国関係の会合に向け、「北朝鮮によるいわれのない攻撃に対し、韓国国民に深い哀悼の意を捧げる」との声明を出した。

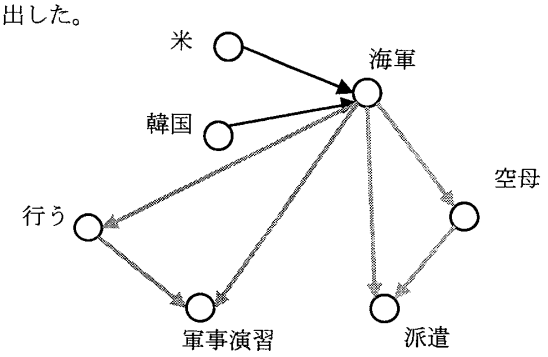

Fig. 2 Example of KeyFlow of the given document

4. Concluding Remarks

In this paper we have proposed a novel algorithm called KeyFlow for automatic extraction of viewpoints from electronic documents. The proposed algorithm works on single document without training data so that training cost could be removed. KeyFlow is supposed to efficiently extract heavy triangles as anchor points of the input document. These triangles can then be used to identify sentences that central to the topic of the article.

We have described an efficient method to extract triangles from connected graph. When run on real world documents, we have to the some thresholds λ, δ and μ. Also the proposed algorithm relies on high quality results of morphology parsing and syntactic parsing. These and the experimental evaluation will be our future work.

References

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

[2]. 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.

[3]. 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.