Keyphrase Extraction with Sequential Pattern Mining

Qingren Wang[1], Victor S. Sheng[2], Xindong Wu[1,3]

[1]Department of Computer Science, Hefei University of Technology, Hefei, P.R. China
[2]Depart of Computer Science, University of Central Arkansas, Conway, AR 72035, USA
[3]Department of Computer Science, University of Louisiana, Lafayette, Louisiana 70503, USA
+86 13866144175, qingren.wang@mail.hfut.edu.cn, ssheng@uca.edu, xwu@louisiana.edu

Abstract

Existing studies show that extracting a complete keyphrase candidate set is the first and crucial step to extract high quality keyphrases from documents. Based on a common sense that words do not repeatedly appear in an effective keyphrase, we propose a novel algorithm named KCSP for document-specific **k**eyphrase **c**andidate **s**earch using sequential **p**attern mining with gap constraints, which only needs to scan a document once and automatically specifies appropriate gap constraints for words without users' participation. The experimental results confirm that it helps improve the quality of keyphrase extraction.

Introduction

A keyphrase is a list of words that capture a main topic discussed in a document, which can help understand the main points of this document (Liu et al. 2012). Keyphrases have been successfully used in many text mining tasks, such as topic extraction and document summarization. Many approaches have been proposed to discover high quality keyphrases from documents. To the best of our knowledge, all these approaches have a same first step, i.e., keyphrase candidate search. Searching a complete keyphrase candidate set that includes semantic relations in context is crucial to extract high quality keyphrases.

Previous works simply treated independent or contiguous high frequency words as keyphrase candidates. However, no matter how many contiguous words that a candidate has, it still can be viewed as a high frequency word alone, and words alone ignore semantic relations in context. Since a keyphrase candidate is a list of words which can be treated as an independent pattern, Xie et al. (2014) proposed an efficient algorithm SPMW, which combines wildcards with sequential pattern mining to search keyphrase candidates. Comparing with frequent word based approaches, sequential pattern mining with wildcards can discover a richer keyphrase candidate set since wildcards provide gap constraints with a great flexibility for mining patterns to capture semantic relations (Agrawal and Srikant, 1995), which helps improve the quality of keyphrase extraction. However, repeated document scanning and manually setting gap constraints are two main weaknesses of sequential-pattern-mining-based approaches.

To solve the two weaknesses, on the basis of a common sense that words do not repeatedly appear in an effective keyphrase, we propose a novel algorithm called KCSP for document-specific **k**eyphrase **c**andidate **s**earch using sequential **p**attern mining with gap constraints. Besides, since many mechanisms based on TF-IDF have been shown to work well in practice (Liu et al. 2009 and Hasan et al. 2010), we propose another mechanism (pattern frequency with entropy) based on entropy and TF-IDF to extract keyphrases from a pattern set. That is, after we obtain a pattern set of a document using KCSP, we can use the new TF-IDF mechanism to extract keyphrase from the pattern set. Our approach is referred to as KeyRank hereafter.

Algorithm KCSP

The main steps of KCSP are shown as follows.

- Step 1: KCSP scans a document with length L from left to right once to obtain every word whose frequency is no less than the support threshold *min_sup*, and together with its corresponding positions, puts them into an array called *WordArray*. Any two adjacent positions pos_1 and pos_2 ($pos_1<pos_2$) of a word w_1 can generate an interval $Range[pos_1, pos_2]$, which is treated as a gap constraint of w_1 at position pos_1. Besides, since the last position of each word does not have a bigger adjacent position, KCSP uses $L+1$ instead.

This research has been supported by the Program for Changjiang Scholars and Innovative Research Team in University (PCSIRT) of the Ministry of Education, China, under grantIRT13059, the National 973 Program of China under grant 2013CB329604, the National Natural Science Foundation of China under grant 61229301, and the US National Science Foundation under grant IIS-1115417.

- Step 2: KCSP tries each word in *WordArray* one by one as the first word of a pattern. If word w_1 can be the first word of a pattern P, KCSP outputs w_1 as an independent pattern and calls Step 3 (with input P). When all words in *WordArray* have been tried, KCSP stops.
- Step 3: KCSP chooses a word w_2 from *WordArray*, and calls Step 4 to verify whether P and w_2 could be concatenated as a new pattern P_1. If positive, KCSP calls Step 3 (with input P_1) recursively and outputs P_1 as an independent pattern; otherwise, KCSP calls Step 2.
- Step 4: KCSP lists all gap constraints of the last word of pattern P to calculate how many positions of w_2 satisfy the gap constraints. If the number of positions of w_2 that satisfy gap constraints is no less than *min_sup*, it means that P and w_2 could be concatenated as a new pattern P_1.

Table 1. Example for KCSP interpretation

word	*c*	*t*
positions & gap constraints	1 *Range* [1,3]	2 *Range* [2,4]
	3 *Range* [3,5]	4 *Range* [4,5]

Here we use a simple example to briefly explain how KCSP works. Given an ordered sequence database *T*=*ctct* with length *L*=4, and *min_sup*=2. We can easily get positions (starting from 1) and corresponding gap constraints of words *c* and *t*, listed in each column of Table 1. Since word *c* has the minimum position, the search process starts from *c*. As *c*'s frequency is 2, which equals *min_sup*, it can be the first word of a pattern, and meanwhile itself is an independent pattern. Then, KCSP calculates all positions of *t* (shown in the third column in Table 1) to see if they could satisfy any gap constraint of *c* (shown in the second column in Table 1). Position 2 of *t* satisfies *Range*[1,3] of *c*, and position 4 of *t* satisfies *Range*[3,5] of *c*. Hence, the number of positions of *t* that satisfy gap constraints of *c* is 2, which equals *min_sup*. KCSP concatenates *c* and *t* as a new pattern *ct*. Besides, since *t*'s frequency equals *min_sup*, KCSP outputs *t* as an independent pattern too. Finally, KCSP finds three patterns, i.e., *c*, *t*, and *ct*.

Experiments

To investigate the performance of KeyRank, we conduct experiments on a dataset SemEval-2010, which contains 244 articles (144 for training and 100 for testing). We compare KeyRank with two popular approaches, Kea (Witten et al. 1999) and TextRank (Mihalcea et al. 2004). Besides, precision (*P* in short), recall (*R* in short), and the F_1 score are used as the performance metrics of keyphrase extraction:

$$P = \#correct/\#extracted$$
$$R = \#correct/\#labeled$$
$$F_1 = 2 \times P \times R/(P + R)$$

where *#correct* denotes the number of correctly extracted keyphrases, *#extracted* denotes the number of extracted keyphrases, and *#labeled* denotes the number of labeled keyphrases.

Experimental results (with respect to different numbers of keyphrases, from 3 to 25) are shown in Figures 1 and 2. Both figures show that our approach KeyRank performs the best, followed by Kea. Both KeyRank and Kea significantly perform better than TextRank.

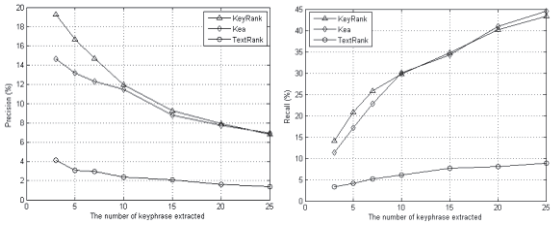

Figure 1. The precisions & recalls of KeyRank, Kea, and TextRank

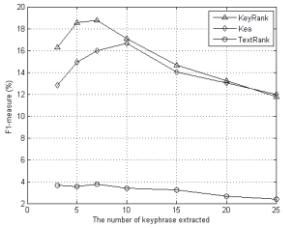

Figure 2. The F_1 scores of KeyRank, Kea, and TextRank

Conclusion

Based on a common sense that words do not repeatedly appear in an effective keyphrase, we propose a novel algorithm called KCSP for document-specific keyphrase candidate search using sequential pattern mining with gap constraints. The experiments confirm that it helps improve the quality of keyphrase extraction.

References

Liu X., Song Y., Liu S., and Wang H. 2012. Automatic taxonomy construction from keywords, *ACM SIGKDD 2012*, 1433-1441.

Xie F., Wu X., and Zhu X. 2014. Document-Specific Keyphrase Extraction Using Sequential Patterns with Wildcards, *IEEE ICDM 2014*, 1055-1060.

Agrawal R. and Srikant R., 1995. Mining sequential patterns, *IEEE ICDE 1995*, 3-14.

Witten I. H., Paynter G.W., Frank E., Gutwin C., and Nevill-Manning C.G. 1999. KEA: Practical Automatic Keyphrase Extraction, *ACM JCDL 1999*, 254-255.

Liu F., Pennell D., Liu F., and Liu Y. 2009. Unsupervised Approaches for Automatic Keyword Extraction Using Meeting Transcripts, *NAACL 2009*, 620-628.

Hasan K. S. and Vincent Ng 2010. Conundrums in Unsupervised Keyphrase Extraction: Making Sense of the State-of-the-Art, *COLING 2010*, 365-373.