Multi-document summarization using closed patterns

Ji-Peng Qiang a,b, Ping Chen b, Wei Ding b, Fei Xie a,c, Xindong Wu a,d,e

a Department of Computer Science, Hefei University of Technology, Hefei 230009, China
b Department of Computer Science, University of Massachusetts Boston, Boston 02125, USA
ca Department of Computer Science and Technology, Hefei Normal University, Hefei 230601, China
d Department of Computer Science, University of Vermont, Burlington, VT 05405, USA

crossMark

A R T I C L E   I N F O

Article history:
Received 6 March 2015
Revised 20 January 2016
Accepted 21 January 2016
Available online 2 March 2016

Keywords:
Multi-document summarization
Closed patterns
Text mining
Diversity
Content coverage

A B S T R A C T

There are two main categories of multi-document summarization: term-based and ontology-based methods. A term-based method cannot deal with the problems of polysemy and synonymy. An ontology-based approach addresses such problems by taking into account of the semantic information of document content, but the construction of ontology requires lots of manpower. To overcome these open problems, this paper presents a pattern-based model for generic multi-document summarization, which exploits closed patterns to extract the most salient sentences from a document collection and reduce redundancy in the summary. Our method calculates the weight of each sentence of a document collection by accumulating the weights of its covering closed patterns with respect to this sentence, and iteratively selects one sentence that owns the highest weight and less similarity to the previously selected sentences, until reaching the length limitation. The sentence weight calculation by patterns reduces the dimension and captures more relevant information. Our method combines the advantages of the term-based and ontology-based models while avoiding their weaknesses. Empirical studies on the benchmark DUC2004 datasets demonstrate that our pattern-based method significantly outperforms the state-of-the-art methods. Multi-document summarization can be used to extract a particular individual’s opinions in the form of closed patterns, from this individual’s documents shared in social networks, hence provides a useful tool for further analyzing the individual’s behavior and influence in group activities.

© 2016 Elsevier B.V. All rights reserved.

1. Introduction

Multi-document summarization has attracted much attention in recent years. With the rapid development of the World Wide Web, the explosion of electronic documents presents a serious challenge for readers to extract useful information from many relevant and similar documents. The Internet provides access to a huge volume of documents on a variety of topics with a considerable amount of redundancy. It calls for a robust multi-document summarization system, which can generate a succinct representation of a document collection by reducing information redundancy.

A large number of multi-document summarization systems have been presented in the literature. For example, the centroid-based methods [23,37] use clustering algorithms to generate sentences’ clusters by calculating sentence similarity, and then select the most representative sentences from different clusters. The graph-based approaches [45,50] build a graph-based model, and then select sentences by means of voting from their neighbors using ideas like the well-known PageRank algorithm [7]. By considering latent semantics of document content, many methods based on latent semantic analysis [14] and non-negative matrix factorization [22,36] have been proposed. In addition, some ontology-based approaches [5,16] have also been used to produce summaries using lexical semantics.

Existing approaches basically fall into two major categories: term-based and ontology-based methods. A term-based method has the advantages of efficiency and maturity for term weight calculation. However, the main drawback is that it only focuses on single word significance without considering the problems of polysemy and synonymy, where polysemy means multiple meanings for a given word, and synonymy means multiple words express the same meanings. To solve these problems, ontology-based approaches take into account of meanings of lexicons. But they are restricted in some specific application domains where ontologies are available, and they cannot attain the semantic meanings of terms that do not exist in the ontology. Meanwhile, the construction of ontology is usually prohibitively expensive. To overcome these inherent weaknesses and keep the advantages of both term-based and ontology-based methods, we propose to generate a
summary based on closed patterns, because closed patterns can capture the associations among the words, and no additional resources are required.

Over the past decade, a large number of association mining technologies have been proposed for a variety of tasks, including association rule mining, frequent itemset mining, sequential pattern mining, closed pattern mining, and maximum pattern mining [25]. In this paper, we will discuss how to effectively use these patterns in multi-document summarization. There are many types of sequential patterns, including frequent patterns, closed patterns, and so on [11,33]. As closed patterns are more compact and contain more information than frequent patterns without losing any information, we choose closed patterns for term weight calculation. To be specific, this paper will discuss a novel method for multi-document summarization using closed patterns, namely pattern-based summarization, which simultaneously considers content coverage and non-redundancy. It holds good statistical properties and captures more relevant and diverse information relative to the term-based methods. Compared with the ontology-based approaches, our multi-document summarization using closed patterns can capture informative terms in the document collection. The method does not rely on any external resources such as lexical knowledge bases. It only relies on the information in a document collection from which the summary is to be created.

Our pattern-based method includes the following steps. First, we mine all closed sequential patterns from a corpus. Then, we present a novel method that represents all sentences using these closed patterns. The model of sentence representation covers the main content of the document collection by calculating pattern weights with respect to the distribution of the closed patterns. Finally, we iteratively choose informative and non-redundant sentences by adopting a variant of the maximal marginal relevance evaluation strategy [8]. Experiments on the standard benchmark DUC2004 data sets demonstrate that the proposed algorithm outperforms the state-of-the-art term-based and ontology-based methods.

The rest of this paper is organized as follows. Section 2 discusses related work. Section 3 presents pattern-based summarization. Section 4 shows experimental results. Finally, Section 5 concludes the paper.

2. Related work

 Depending on the number of documents, automatic document summarization includes single-document summarization and multi-document summarization [3,46]. Single-document summarization only condenses one document into a summary, whereas multi-document summarization condenses a document collection into a single shorter representation. Multi-document summarization is considered as an extension of single-document summarization, and needs more sophisticated technologies and attracts much attention [29,31].

 Multi-document summarization methods can be classified into two classes: extractive summarization and abstractive summarization [24,26]. Extractive summarization extracts the most informative document components, and abstractive summarization involves reformulation of contents. Extractive summarization is a simple but robust method without the requirement for advanced post-processing steps. Abstractive summarization requires deep natural language processing techniques for understanding the documents [30]. Extractive summarization is more feasible and has become the standard in multi-document summarization. Summarization techniques also can also be categorized into query-based or generic (given a query or not), supervised or unsupervised methods (with a training set or not).

In this paper, we focus on unsupervised, extractive, generic, multi-document summarization. Unsupervised, extractive, and generic methods usually adopt the bag-of-words model for term weight calculation, also called term-based methods, which often use term frequency/inverse document frequency (TF–IDF) weighting model and some extended schemes [20].

Term-based methods can be divided into the following categories. The centroid-based methods, as one of the most popular extractive methods, group document sentences into homogeneous clusters, and then select the representative sentences through computing the similarity values between sentences and the centroids of the clusters. For example, MEAD [23] computes the average cosine similarity between sentences and the rest of the sentences in the document collection as the centroid value of a sentence. Gong and Liu [14] presented a method to identify semantically important sentences using the latent semantic analysis (LSA). They first created a term–sentence matrix with each entry representing the weight of a term in its documents. Then they derived the latent semantic information by applying singular value decomposition (SVD). Some methods were proposed based on non-negative matrix factorization (NMF) [22,36]. The NMF-based methods also first create a term–sentence matrix to select meaningful sentences. Other methods were also developed including conditional random fields [32] and hidden Markov model [9].

The graph-based approaches [12,45,50] also belong to extractive summarization. They first produce a similarity graph, in which each node represents a sentence. When the cosine similarity value between a pair of sentences exceeds a threshold, these two sentences are connected by an edge. Erkan and Radev [12] proposed a method, called LexPageRank, which ranks the sentences based on the similarity graph following the well-known PageRank algorithm. Other improved graph-based algorithms have been proposed [6,45,50]. Bollegala et al. [6] presented a bottom-up approach to arrange sentences extracted for multi-document summarization. They defined four criteria, chronology, topical-closeness, precedence and succession, for capturing the association and order of two sentences.

Compared to the term-based methods, some ontology-based approaches [5,16] have been used to produce summaries. Specifically, ontologies have been used to (i) identify the concepts that are either most pertinent to a query [17,43] or most suitable for performing query expansion [27], (ii) model the context in which summaries are generated in a variety of domains, such as business domain [40], disaster management domain [20] and so on. Baralis et al. [5] proposed an ontology-based approach, called Yago-based summarization, which relied on Wikipedia [39] to map the words to non-ambiguous ontological concepts called entities. Yago-based summarization selects document sentences according to the previously assigned entities. Ontology-based approaches are limited in specific application domains, and also it takes much effort to construct the ontologies.

Pattern-mining techniques have been extensively studied for many years in data mining, such as frequent itemset algorithms [Apriori [1], FP-tree [15]], sequential pattern algorithms (PrefixSpan [28], SPADE [47]), closed sequential pattern algorithms (CloSpan [44], AGRAp[13], BIDE [38]). As frequent itemsets or frequent patterns include more contextual semantic information than an individual term, they can improve the effectiveness of text mining applications, for example, text classification [2,41,49] and text clustering [19,48]. Algarini and Li [2] calculated term weights based on both their frequencies in documents and their distributions in sequential patterns. Zhang et al. [48] proposed maximum capturing (MC) for text clustering using frequent itemsets. MC can be divided into two components: constructing document clusters and assigning document topics.
Baralis et al. [4] presented a method for summarization based on frequent itemsets and the traditional bag-of-word model. This method does not perform very well without considering the low frequency problem. Given a specific topic, a general pattern is usually frequent, and a specific pattern is not [49]. If we decrease the minimum support, many noisy patterns would be generated. So it is a difficult problem how to use frequent patterns to accurately extract a summary. In addition, frequent itemsets do not take the order of terms into account, and include a lot of redundant patterns. In this paper, we conduct a study on multi-document summarization based on closed patterns.

In summary, we aim to generate a summary through closed patterns in this paper. Compared with the term-based methods, our pattern-based method has three advantages. The first is that the representation reduces the dimensionality which only needs these terms from all closed patterns, and term-based methods need all terms in a document collection. The second advantage is that the representation can capture more contextual semantic information than individual terms. The third advantage is that the representation can solve the low frequency problem, because we calculate term weights through the distribution of terms in closed patterns rather than their distributions in the document collection. Compared with the ontology-based methods, our method does not require as much manpower to construct the ontology. Compared with frequent itemsets, closed patterns not only contain the term order information, but also are more concise.

3. Pattern-based summarization

In this section, we discuss how to generate a summary using closed patterns, denoted as pattern-based summarization.

3.1. Overview of pattern-based summarization

Consider a document collection \( D = \{d_1, d_2, \ldots, d_i, \ldots, d_N\} \), where \( d_i \) represents the \( i \)th document of \( D \). \( N \) is the number of documents in \( D \). Each document \( d_i \in D \) consists of a set of sentences \( \{s_{i1}, \ldots, s_{ij}, \ldots, s_{ike}\} \), where \( s_{ij} \) represents the \( j \)th sentence of document \( d_i \). Here, \( |d_i| \) is the number of sentences in document \( d_i \). Let \( U \) represents all sentences in \( D \), and \( n \) be the total number of sentences in \( D \), where \( n = \sum_{i=1}^{N} |d_i| \). Let \( \{p_1, p_2, \ldots, p_e\} \) be the set of closed patterns from all sentences in \( D \) respectively. Fig. 1 illustrates the proposed graph model which is different from the traditional term-based methods. Both sentence ranking and sentence similarity can be calculated through closed patterns belonging to the sentences. The aim is to generate a summary \( S = \{s_{il}\} (1 \leq l \leq N, 1 \leq j \leq l) \) that contains a subset of sentences that are representative of \( U \).

The outline of pattern-based summarization is shown in Fig. 2. Specifically:

1. Sentence representation using closed patterns. This step obtains all closed sequential patterns from document collection by calling a closed sequential pattern algorithm. Then each sentence is represented by its terms and their corresponding weights that are calculated by accumulating the weight of its covering closed patterns with respect to this sentence.
2. Sentence ranking. To select the most pertinent and meaningful sentences into the summary, sentences are ranked according to the previous sentence representation.
3. Sentence selection. The sentences that include more closed patterns with high weight and less similar to other sentences are chosen into the summary. We iteratively select one sentence that owns the highest weight and less similar to the previously selected ones, until reaching the length limit.

3.2. Sentence representation using closed sequential patterns

3.2.1. Mining closed sequential patterns

First, we will mine all closed sequential patterns from a set of all sentences in \( D \). A pattern \( p = \{t_1, t_2, \ldots, t_m\} \) is an ordered list of terms. For brevity, a pattern is also written as \( p = \{t_1, t_2, \ldots, t_m\} \), \( p = \{t_1, t_2, \ldots, t_m\} \) is a sub-pattern of another pattern \( p' = \{t'_1, t'_2, \ldots, t'_l\} (l \leq M) \), denoted by \( p \subseteq p' \) (or \( p' \) is a super-pattern of \( p \)), if there exists a sequence of positions \( 1 \leq j_1 < j_2 < \cdots < j_m \leq n \) s.t. \( t_j = t'_k \) for \( i = 1, 2, \ldots, m \).

Definition 1 [44]. Given a pattern \( p \) that is composed of a set of terms, its covering sentences are denoted as \( \text{coverSent}(p) = \{s|s \in U, p \subseteq s\} \). The support of a pattern \( p \) is the number of sentences in \( D \) containing \( p \), namely \( \text{sup}(p) = |\text{coverSent}(p)| \), where \( |\text{coverSent}(p)| \) is the number of sentences.

Definition 2. Given a pattern \( p \), its covering documents are denoted as \( \text{coverDoc}(p) = \{d|d \in D, p \subseteq d\} \). Here, \( |\text{coverDoc}(p)| \) is the number of documents.

Definition 3 [49]. A pattern \( p \) is frequent if its support, namely \( \text{coverSent}(p) \), is more than or equal to a user-specified minimum support (sup) value.

Example 1. Given a set of sentences from two news reports \( D \) in Table 1, and minimum support 3. In preprocessing, all stop words of the input sentences are removed, and all letters are converted into lowercase. We only consider the first occurrence when a sentence has multiple occurrences of one term. These terms that occur more than 3 times are saved in Table 2. Suppose pattern \( p = "Obama McConnel" \), we can see there are three sentences \( "s_{11}, s_{12}, s_{22}\" \) that include this pattern. So this pattern is a frequent pattern. Through pattern mining algorithm, all the frequent patterns are shown in Table 3 with minimum support 3.

Definition 4 [44]. A pattern \( p \) is a closed pattern if and only if there does not exist any super-pattern \( p' (p \subseteq p') \) s.t. \( \text{sup}(p) = \text{sup}(p') \), namely, \( \text{sup}(p) > \text{sup}(p') \) for all patterns \( p' \geq p \). \( p \) is a non-closed pattern if and only if there exists a super-pattern \( p' \) s.t. \( \text{sup}(p) = \text{sup}(p') \).

For example, in Table 3, the support of pattern "Obama leader" is 4, which is a frequent pattern. And the support of its all super-patterns is smaller than 4, so the pattern "Obama leader" is a closed pattern.
Table 1
A set of sentences from two news reports.

<table>
<thead>
<tr>
<th>Document</th>
<th>Sentence</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>d_1</td>
<td>s_1^1</td>
<td>Obama and McConnell, when republicans take charge in the senate in January, who will become majority leader.</td>
</tr>
<tr>
<td></td>
<td>s_1^2</td>
<td>President Obama's determination to act alone to change the immigration system promptly drove a wedge Wednesday into the post-election commitment from the president and republican leaders to find common ground under the new political alignment.</td>
</tr>
<tr>
<td>d_2</td>
<td>s_2^1</td>
<td>President Barack Obama and Senate Republican leader McConnell pledged to try to turn divided government into a force for good rather than gridlock on Wednesday, yet warned of veto showdowns as well.</td>
</tr>
<tr>
<td></td>
<td>s_2^2</td>
<td>No one was more invested in Tuesday's election than Senate minority leader McConnell, whose power in Washington is about to vastly expand, and with it his ability to thwart President Obama's agenda.</td>
</tr>
</tbody>
</table>

Table 2
Terms to represent the sentences.

<table>
<thead>
<tr>
<th>Sentence</th>
<th>Terms</th>
</tr>
</thead>
<tbody>
<tr>
<td>s_1^1</td>
<td>Obama, McConnell, republican, senate, leader</td>
</tr>
<tr>
<td>s_1^2</td>
<td>President, Obama, republican, leader</td>
</tr>
<tr>
<td>s_2^1</td>
<td>President, Obama, senate, republican, leader, McConnell</td>
</tr>
<tr>
<td>s_2^2</td>
<td>Senate, leader, McConnell, president, Obama</td>
</tr>
</tbody>
</table>

In all, there are 23 frequent patterns. Not all frequent patterns in Table 3 are useful. For example, pattern [leader] always occurs with term "Obama", i.e., the shorter pattern [Obama] is always part of the longer pattern [leader, Obama] in all its covering sentences. Therefore, the shorter one, [Obama], is a redundant pattern, and we only keep the longest patterns, namely closed patterns. After pruning non-closed patterns, we only have four closed patterns, which are shown in bold font.

We shall point out that a pattern is different from N-gram or term co-occurrence. N-gram is a contiguous sequence of n terms from a given set of sentences, and pattern allows flexible gaps between items. Term co-occurrence only considers the frequency between two words, and a pattern is a set of ordered items that appear frequently together in a set of sentences.

3.2.2. Sentence representation

In this section, we will represent all sentences using closed patterns. At first, we introduce the composition operation ⊕. Let t be a term-weight pair composing of a set of terms and their weights, such as \( [(t_1, q_1), ..., (t_m, q_m)] \), where \( t_i \) denotes a single term, and \( q_i \) is its weight. A closed pattern can be expressed as a term-weight pair.

**Definition 5.** Let \( p_i = [(s_1, ..., s_m), (y_1, ..., y_y)] \) denote a closed pattern, where \( m \) is the number of the terms, and \( y \) is the number of its covering sentences. The weight of the closed pattern can be calculated as follows:

\[
\begin{align*}
   w(p_i) = |\text{coverSent}(p_i)| = |\text{coverDoc}(p_i)|/N
\end{align*}
\]

Table 3
Frequent patterns and their covering sentences with sup = 3.

<table>
<thead>
<tr>
<th>Frequent patterns</th>
<th>Covering sentences</th>
<th>Frequent patterns</th>
<th>Covering sentences</th>
</tr>
</thead>
<tbody>
<tr>
<td>Obama</td>
<td>s_1^1, s_2^1, s_3^1, s_4^1</td>
<td>Obama republican</td>
<td>s_1^1, s_2^1, s_3^1, s_4^1</td>
</tr>
<tr>
<td>McConnell</td>
<td>s_1^1, s_2^1, s_3^2</td>
<td>Senate leader</td>
<td>s_1^1, s_2^1, s_3^1, s_4^1</td>
</tr>
<tr>
<td>Senate</td>
<td>s_1^1, s_2^1, s_3^2, s_4^2</td>
<td>Leader president</td>
<td>s_1^1, s_2^1, s_3^2, s_4^2</td>
</tr>
<tr>
<td>Leader</td>
<td>s_1^1, s_2^1, s_3^2, s_4^2</td>
<td>Obama president</td>
<td>s_1^1, s_2^1, s_3^2, s_4^2</td>
</tr>
<tr>
<td>Republican</td>
<td>s_1^1, s_2^1, s_3^1, s_4^2</td>
<td>Obama McConnell</td>
<td>s_1^1, s_2^1, s_3^1, s_4^2</td>
</tr>
<tr>
<td>Obama McConnell</td>
<td>s_1^1, s_2^1, s_3^2</td>
<td>Leader McConnell</td>
<td>s_1^1, s_2^1, s_3^2, s_4^2</td>
</tr>
<tr>
<td>Obama senate</td>
<td>s_1^1, s_2^1, s_3^2</td>
<td>Obama McConnell</td>
<td>s_1^1, s_2^1, s_3^2, s_4^2</td>
</tr>
<tr>
<td>McConnell senate</td>
<td>s_1^1, s_2^1, s_3^2</td>
<td>Obama republican</td>
<td>s_1^1, s_2^1, s_3^2, s_4^2</td>
</tr>
<tr>
<td>Obama leader</td>
<td>s_1^1, s_2^1, s_3^1, s_4^2</td>
<td>Obama president</td>
<td>s_1^1, s_2^1, s_3^2, s_4^2</td>
</tr>
<tr>
<td>Republican leader</td>
<td>s_1^1, s_2^1, s_3^2</td>
<td>Obama leader McConnell</td>
<td>s_1^1, s_2^1, s_3^2, s_4^2</td>
</tr>
</tbody>
</table>
where \(|\text{coverSent}(p_i)|\) is the number of \(p_i\)'s covering sentences, \(|\text{coverDoc}(p_i)|\) is the number of \(p_i\)'s covering documents, and \(N\) is the number of all documents. Through this formula, one closed pattern existing in more documents will have a higher weight than the other closed pattern existing in fewer documents under the same support.

For example, closed pattern \{Obama, republican, leader\} and its covering sentences \((s_1^1, s_2^1, s_3^1)\) in Table 3 can be represented as one term-weight pair, \(tw = \{(\text{Obama, 3), (republican, 3), (leader, 3)}\}\), where 3 is the weight of this closed pattern according to (1).

**Definition 6.** Let \(tw_1 = \{(t_1,a_1),\ldots,(t_y,a_y)\}\) and \(tw_2 = \{\{w_j,b_j\},\ldots,\{w_j,b_j\}\}\) be two term-weight pairs associated to two patterns, the composition operation \(\oplus\) between \(tw_1\) and \(tw_2\), \(tw_1 \oplus tw_2\) denoted as:

\[
tw_1 \oplus tw_2 = \left\{ (t_i,a_i+b_j) \mid (t_i,a_i) \in tw_1, (w_j,b_j) \in tw_2, t_i = w_j \right\} \\
\cup \left\{ (t_i,a_i) \mid (t_i,a_i) \in tw_1, t_i \notin \bigcup_j W_j \right\} \\
\cup \left\{ (w_j,b_j) \mid (w_j,b_j) \in tw_2, w_j \notin \bigcup_i T_i \right\}
\]  

(2)

The composition operation is interchangeable, namely, \(tw_1 \oplus tw_2 = tw_2 \oplus tw_1\). Here, \(tw_1 \oplus tw_2 = tw_1\), where \(\oplus\) represents null. For example, if \(tw_1 = \{(t_1,3),(t_2,2),(t_4,4)\}\), \(tw_2 = \{(t_1,5),(t_2,3),(t_5,1)\}\), we have \(tw_1 \oplus tw_2 = \{(t_1,3),(t_2,7),(t_2,2),(t_4,4),(t_5,1)\}\).

Let \(tw_1, tw_2,\ldots, tw_k\) be the term-weight pairs of closed patterns from all sentences \(U\) in \(D\). Here, we only use a closed pattern whose size is more than 1, because it can capture the semantic relationship of terms. Let \(\{tw_1, tw_2,\ldots, tw_k\}\) be the set of closed patterns with respect to sentence \(s_i\), where \(i \leq i_1, i_2,\ldots, i_e \leq e\). Finally, the representation of sentence \(s_i\) can be obtained using the following formula:

\[
tw(s_i) = tw_1 \oplus tw_2 \oplus \cdots \oplus tw_e
\]  

(3)

For example, using Table 3, the term-weight pairs of all closed patterns are expressed as:

\[
tw_1 = \{(\text{Obama, 4}), (\text{leader, 4})\}, \quad tw_2 = \{(\text{Obama, 3}), (\text{republican, 3}), (\text{leader, 3})\}, \quad tw_3 = \{(\text{Obama, 3}), (\text{president, 3}), (\text{leader, 3})\}, \quad tw_4 = \{(\text{Obama, 3}), (\text{leader, 3}), (\text{McConnel, 3}), (\text{senate, 3})\}\n\]

**Definition 7.** A sentence is a set of term-weight pairs that appear in this sentence.

For example, using \(tw_1, tw_2, tw_3\) in Table 3, we have \(s_1^1 = \{tw_1, tw_2, tw_4\}\), \(s_2^1 = \{tw_1, tw_2, tw_3\}\), \(s_3^1 = \{tw_1, tw_2, tw_3, tw_4\}\). Then we can obtain the representation of each sentence below:

\[
tw(s_1^1) = \{(\text{senate, 3}), (\text{leader, 10}), (\text{Obama, 10}), (\text{republican, 3}), (\text{McConnel, 3})\}\n\]

\[
tw(s_2^1) = \{(\text{Obama, 10}), (\text{republican, 3}), (\text{president, 3}), (\text{leader, 10})\}\n\]

\[
tw(s_3^1) = \{(\text{senate, 3}), (\text{McConnel, 3}), (\text{leader, 13}), (\text{Obama, 13}), (\text{republican, 3}), (\text{president, 3})\}\n\]

Algorithm 1 describes the process of obtaining the set of sentence representation. For all sentences in a document collection, all closed patterns \(P\) are discovered using closed pattern mining algorithm (e.g. CloSpan [44], AGrA [13], SPMW [42]) in Step 1, which returns all closed patterns and their corresponding covering sentences. The weight of each pattern in \(P\) can be calculated using (1) from Steps 2 to 4. Thereafter, from Steps 6 to 10, the representation of each sentence in \(U\) can be obtained using (3).

Let \(n\) be the number of all sentences, \(e\) be the number of closed patterns, \(I\) be the average number of the covering sentences of each closed pattern, and \(k\) be the average number of terms of each sentence. Step 1 calls the algorithm of closed pattern mining that takes \(O(CP)\), where \(CP\) is the time complexity of the algorithm of closed pattern mining. It takes \(O(n)\) from Steps 2 to 4 and \(O(n)\) at Step 5. The \(\oplus\) operation takes \(O(k^2)\) at most. So it takes \(O(ek^2)\) from Steps 6 to 10. Therefore, the time complexity of sentence representation is \(O(CP + n + (el + ek^2) = O(CP + n + ek^2)\).

### 3.3. Sentence ranking

Each sentence has been represented using term-weight pairs. In this step, pattern-based summarization ranks all sentences according to their sentence representation. The basic idea of sentence ranking is that these sentences will have high scores that contain more closed patterns with high weight. Let \(s_j^i\) be the \(j\)th sentence in document \(d_i\). Hence, the score of the sentence \(s_j^i\) is denoted as \(score(s_j^i)\) and defined as follows:

\[
score(s_j^i) = \frac{weight(s_j^i)}{|s_j^i|} \times \left( 1 - \frac{j - 1}{|d_i|} \right)
\]

(4)

where \(weight(s_j^i)\) is a weight assigned to the sentence \(s_j^i\) using the weighting scheme described in Section 3.4. \(|s_j^i|\) is the number of words in sentence \(s_j^i\). The aim is to get the highest average weight that is calculated as \(weight(s_j^i)\) divided by \(|s_j^i|\). By multiplying \(\frac{1}{i - 1 + \frac{1}{|d_i|}}\), we integrate the position of each sentence in its document into the sentence ranking. Intuitively, the starting sentences in a document contain more new information than the following sentences. Through the second part of Eq. (4), the first sentence of a document gets maximum value, and the last sentence gets the minimum value. For example, in Table 1, the value of \(s_1^1\) is 1,
and the value of \( s_2 \) is 0.5 when only computing the second part of Eq. (4).

From the closed patterns, we know that they represent these terms with high frequency in the document collection. And the weight of each sentence is decided by the number of closed patterns belonging to the sentence and the support of each closed pattern. Thus, we can draw a conclusion that these sentences containing more closed patterns have high scores. Note that the sentences that do not contain any closed patterns have minimal scores (i.e., 0).

3.4. Sentence selection

Given the result of sentence ranking, we focus on generating the summary of the document collection considering both content coverage and non-redundancy, until a given length of the summary is reached. For reducing redundancy in the summary, some methods measure the similarity of next candidate sentence to that of previously selected ones, and select it if its similarity is below a threshold [31]. The most popular method is maximal marginal relevance (MMR), which was first introduced in query-based summarization [8]. At each iteration, MMR tries to select sentences that are dissimilar from the ones already selected. As our method is not query-based, we adapt the former selection strategy.

The most popular method of computing sentence similarity in MMR is the term frequency–inverse sentence frequency (TF–ISF) of the candidate terms [8,23]. As sentences are very short comparing to text, most terms only have one occurrence in a sentence. Therefore, TF of TF–ISF cannot play a role in computing sentence similarity. So TF–ISF cannot hardly make a distinction without considering the context information of each term. We thus propose a new method of computing sentence similarity using their common closed patterns.

Here, let \( U = \{ s_1, s_2, \ldots, s_n \} \) be all sentences in \( D \). We use an adjacency matrix \( R \) to describe sentence–sentence similarity with each entry corresponding to the weight of a link in Fig. 1. \( R = [R_{ij}]_{m \times n} \) is defined as follows:

\[
R_{ij} = \sum_{l=1}^{e} |p_l| \times w(p_l) |p_l \cap s_i \cap s_j|, \tag{5}
\]

where \( R_{ij} \) is the similarity value of sentence \( s_i \) and sentence \( s_j \), and \( |p_l| \) is the number of terms in pattern \( p_l \). Through this formula, the similarity value between sentences is generated based on their common closed patterns and the distribution of these closed patterns.

Let \( s_i \) be an arbitrary sentence in \( U \) whose representation is \( ts(s_i) = \{(t_{i1}^1, a_{i1}^1), \ldots, (t_{il}^1, a_{il}^1)\} \), and \( S \) is the set of sentences already selected. The weight of each sentence is calculated as:

\[
weight(s_i) = \lambda \sum_{j=1}^{k} a_{ij} - (1 - \lambda) \max_{j \in S} R_{ij}, \tag{6}
\]

where \( \lambda \) is a user-specified parameter that controls the tradeoff between content coverage and non-redundancy: if \( \lambda = 1 \), the content coverage is highlighted; if \( \lambda = 0 \), only the non-redundancy is considered. The impact of \( \lambda \) on the summarization performance is discussed in Section 4.

Pattern-based summarization adopts (4) to rank sentences. Using the ranking function, we iteratively select the top \( |S| \) sentences having the highest scores, where the length of these \( |S| \) sentences must be less than or equal to the length limitation \( L \).

The main process of pattern-based summarization (PatSum) is illustrated in Algorithm 2. Step 1 calls SentRep (see Algorithm 1). Steps 3 to 7 compute sentence similarity, which takes \( O(|k|) \), where \( |k| \) is the number of the closed patterns, and \( k \) is the average number of terms of each sentence. Steps 13 to 17 rank sentence, whose time complexity is \( O(n) \). Steps 10 to 20 take \( O(|S|n) \), where \( |S| \) be the number of sentences in summary. The time complexity of Algorithm 2 is \( O(|k|^2 + |S|n) \).

Based on the time complexity of Algorithms 1 and 2, the total time complexity of PatSum is \( O(|CP|n + n + |k| |E|) = O(|CP| + |E| + |S|) \).

4. Experimental results

We implemented pattern-based summarization in JAVA. All the experiments are performed on a Windows machine with an Intel 2.9 GHz CPU and 8GB memory.

4.1. Experiment setting

In order to assess the effectiveness of our method for multi-document summarization, we use the standard benchmark
DUC2004 dataset, which is from Document Understanding Conference (DUC) (http://duc.nist.gov) for generic summarization evaluation. There are 50 document clusters in DUC2004, where each cluster consists of 10 English documents. To evaluate automatic methods, DUC2004 provides at least four human-generated summaries in each cluster. Participants to the DUC2004 contest submitted their own summaries and were evaluated against human-generated summaries.

The preprocessing of DUC2004 dataset is as follows. All documents are segmented into sentences using a script distributed by DUC. In our experiments, stop-words are removed using the stop-word list \(^1\) and the remaining words are stemmed using Porter’s scheme \(^2\).

We use the ROUGE toolkit \(^{21}\) (version 1.5.5) to evaluate the summarization performance, which is adopted by DUC as the official metric for document summarization. ROUGE evaluates the quality of a summary by counting the unit overlaps between the candidate summary and a set of human-generated summaries. The summary that achieves the highest ROUGE score is considered to be the most similar to the human-generated summary. To carry out a fair comparison, we normalize the generated summaries by truncating each of them at 665 bytes before using the ROUGE toolkit. As previously done in these literatures \(^{4,5}\), we also choose the ROUGE-2 and ROUGE-4 representative scores \(^{21}\) in this paper.

### 4.2. Evaluation results on DUC2004

We compare our method with systems participated in DUC2004, four newly proposed methods, one commonly used method, and 8 human-generated summaries provided by DUC2004.

2. 8 human-generated summaries: provided by the DUC2004.
3. TexLexAn: one widely used open source text summarization \(^{35}\).

We present the results of our methods in Table 4, which are calculated using the ROUGE toolkit and compared to other methods.

---

### Table 4

ROUGE values for methods using DUC2004 dataset.

<table>
<thead>
<tr>
<th>Summarizer</th>
<th>ROUGE-2</th>
<th>ROUGE-4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Recall</td>
<td>F-measure</td>
</tr>
<tr>
<td>Top ranked DUC2004 peers</td>
<td>peer124</td>
<td>0.083*</td>
</tr>
<tr>
<td></td>
<td>peer65</td>
<td>0.092*</td>
</tr>
<tr>
<td></td>
<td>peer19</td>
<td>0.080*</td>
</tr>
<tr>
<td></td>
<td>peer44</td>
<td>0.076*</td>
</tr>
<tr>
<td></td>
<td>peer81</td>
<td>0.081*</td>
</tr>
<tr>
<td></td>
<td>peer104</td>
<td>0.085*</td>
</tr>
<tr>
<td>TexLexAn</td>
<td>0.067*</td>
<td>0.067*</td>
</tr>
<tr>
<td>ItemSum</td>
<td>0.083*</td>
<td>0.085*</td>
</tr>
<tr>
<td>Baseline</td>
<td>0.092*</td>
<td>0.091*</td>
</tr>
<tr>
<td>Yago</td>
<td>0.095*</td>
<td>0.094*</td>
</tr>
<tr>
<td>MSSF</td>
<td>0.098</td>
<td>0.098</td>
</tr>
<tr>
<td>PatSum</td>
<td>0.102</td>
<td>0.102</td>
</tr>
<tr>
<td>DUC2004 Humans</td>
<td>A</td>
<td>0.088*</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>0.091*</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>0.094*</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>0.100</td>
</tr>
<tr>
<td></td>
<td>E</td>
<td>0.094*</td>
</tr>
<tr>
<td></td>
<td>F</td>
<td>0.085*</td>
</tr>
<tr>
<td></td>
<td>G</td>
<td>0.083*</td>
</tr>
<tr>
<td></td>
<td>H</td>
<td>0.101</td>
</tr>
</tbody>
</table>

---


\(^2\) http://www.tartarus.org martin/Porter-Stemmer.

\(^3\) The provided command is: ROUGE-1.5.5.pl -e data -x -m -2 4 -u -c 95 -r 1000 -n 4 -f A -p 0.5 -t 0 -d -a.
summarization compared to Yago. It means that ontology is useful for Yago. Hence, pattern-based ranking and selection strategies are more effective than other strategies.

We also compare PatSum with the six top-ranked methods in the DUC2004 contest. It is clear that PatSum outperforms all the methods from DUC2004. We further compare our results with the eight human-generated summaries. We see that in terms of ROUGE-2 F-measure, only one out of eight outperforms our method, and none outperforms our method in terms of ROUGE-4.

For better demonstration of the results, Fig. 3 visually illustrates the comparison. Note that we subtracted the ROUGE scores of the worst method TexLexAn in all automatic methods and added the number 0.01 in these figures, thus the difference can be observed more clearly. We show ROUGE-2 and ROUGE-4 F-measures in Fig. 3.

4.3. Result analysis

A more detailed comparison is discussed in this section. We adopt the relative improvement that is defined as \( (b - a)/a \), when \( b \) is compared to \( a \). In our experiments, \( b \) is PatSum, and \( a \) is the method we compare with. Table 5 shows the results of PatSum compared with other automatic methods in terms of ROUGE-2 and ROUGE-4.

In Table 5 “+” means the result outperforms and “−” means the opposite. Compared to ontology-based method, we see that PatSum improves the performance by 0.07, 0.09, 0.07, 0.18, 0.18 and 0.18, respectively, compared with the state-of-the-art Yago-based summarization. Compared to the best method MSSF of term-based methods, PatSum improves the performance by 0.04, 0.04, 0.04, 0.18, 0.18 and 0.18, respectively. In addition, in Table 5, we have the following observations:

1. Although ItemSum exploits frequent itemsets, PatSum is better than ItemSum. Usually a general pattern occurs more frequently than a specific pattern. PatSum takes advantage of this property and achieves better performance than ItemSum. The promising results can be explained that PatSum can deal with the problem of low frequency. We calculate sentence weights with respect to the distribution of closed patterns rather than their distributions in the document collection. In Section 4.5, we will show the results when using frequent itemsets based on our strategy. We can see that PatSum using frequent itemsets outperforms ItemSum, which clearly shows that the low frequency information is very important for multi-document summarization.

<table>
<thead>
<tr>
<th>Summarizer</th>
<th>ROUGE-2</th>
<th>ROUGE-4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Recall</td>
<td>Precision</td>
</tr>
<tr>
<td>Top ranked DUC2004 peers</td>
<td>peer124</td>
<td>+0.19</td>
</tr>
<tr>
<td></td>
<td>peer65</td>
<td>+0.28</td>
</tr>
<tr>
<td></td>
<td>peer44</td>
<td>+0.43</td>
</tr>
<tr>
<td></td>
<td>peer81</td>
<td>+0.26</td>
</tr>
<tr>
<td></td>
<td>peer104</td>
<td>+0.19</td>
</tr>
<tr>
<td>TexLexAn</td>
<td>+0.52</td>
<td>+0.52</td>
</tr>
<tr>
<td>ItemSum</td>
<td>+0.23</td>
<td>+0.20</td>
</tr>
<tr>
<td>Baseline</td>
<td>+0.11</td>
<td>+0.12</td>
</tr>
<tr>
<td>Yago</td>
<td>+0.07</td>
<td>+0.09</td>
</tr>
<tr>
<td>MSSF</td>
<td>+0.04</td>
<td>+0.04</td>
</tr>
</tbody>
</table>

Fig. 3. Comparison of the methods in terms of ROUGE-2 and ROUGE-4 F-measures.
summarization. In addition, there are a lot of redundant patterns among frequent itemsets compared to closed patterns.

(2) PatSum performs better than Yago, and Yago outperforms most of term-based methods (DUC2004 peers, TexLexAn, Baseline). Because term-based methods have the problems of polysemy and synonymy, Yago captures the semantic information of terms. As a document collection usually comes from the same topic, Yago relies on Wikipedia too many resources and ignores the context information of the current document collection. Yago is restricted to specific application domains because it cannot attain the semantic meanings for terms that do not belong to its ontology. Meanwhile, the construction of ontology can be prohibitively expensive. On the other hand, PatSum can effectively capture the correlations among the words, and also do not require additional resources. Therefore, PatSum outperforms Yago.

(3) PatSum outperforms all term-based methods including TexLexAn, Baseline, and MSSF. Among the term-based methods, MSSF has the best performance which greedily chooses the sentences using submodular functions. But, compared to term-based methods, PatSum outperforms all these methods, because term-based methods suffer from the problems of polysemy and synonymy.

4.4. Parameter setting

The user-specified parameters (λ and sup) could affect the performance of PatSum. Therefore, for analyzing their impact on PatSum, we carry out the experiments with different parameters. The experiments are divided into two categories: one by varying the value of λ in the range [0.2, 0.7] under sup = 4; the other by varying the value of sup in the range [2, 7] under λ = 0.5.

In Fig. 4, the performance of PatSum is shown, as the value of the parameter λ changes. When increasing the value of λ, the performance increases first, and declines finally. The function of λ controls the tradeoff between content coverage and non-redundancy. If the value of λ is a small number, the performance of PatSum has receded dramatically, because PatSum does not reduce much information redundancy through Formula 6. If the λ value surpasses a certain threshold value (e.g., 0.5), it is difficult to remove the redundancy among sentences. As both coverage and non-redundancy are critical for summarization, the best performance through the experiments is achieved by setting λ equals 0.5 under sup = 4 for a good trade-off between coverage and non-redundancy.

Fig. 5 shows the results of pattern-based method by varying the value of minimal support (sup). We can see that the best performance is achieved when sup = 4. Furthermore, the results remain relatively stable when λ ranges between 2 and 7. Since our strategy can solve the problem of low frequency, it means that PatSum is hardly influenced by the minimum support. When sup is sufficiently large, the results are lower than the original results, because only few patterns are generated when support is greater than 6.

4.5. Different representation

In our model, we represent all sentences using closed patterns, and compute their scores. We have conducted some experiments with frequent itemsets, closed itemsets and frequent patterns to study whether they influence the performance of the algorithm. Table 6 illustrates the experimental results, and the best results of each representation are shown in bold font.

We can see that all these strategies are quite effective compared with the state-of-the-art methods. The method using frequent patterns outperforms the method using frequent itemsets, and PatSum outperforms closed itemsets. The reason is that, term order is useful for the summarization, which is not considered by itemsets.
Meanwhile, these two methods using closed itemsets or patterns are superior to the other two methods. This suggests that removing the redundant patterns is critical for summarization. Taking these two aspects into consideration, PatSum chooses closed patterns for summarization.

5. Conclusion

Term-based methods suffer from the problems of polysemy and synonymy, and ontology-based approaches are often restricted in specific application domains. In this paper, we propose a novel method for multi-document summarization based on closed patterns, namely pattern-based summarization (PatSum), which simultaneously considers content coverage and non-redundancy. Compared to term-based methods, PatSum can capture more contextually semantic information than individual terms. Compared to ontology-based methods, PatSum not only can capture the correlations among the words, but also do not require additional information. Experimental results have shown that pattern-based summarization outperforms not only term-based methods and all participating systems on DUC2004 datasets, but also ontology-based method. Multi-document summarization can extract a particular individual’s opinions in the form of closed patterns, from this individual’s documents shared in social networks, hence provides a useful tool for further analyzing the individual’s influence in group activities. In the future, we plan to extend our work to query-based multi-document summarization and multi-document update summarization, with applications to behavior and influence analysis in social networks.

Acknowledgments

This research is partially supported by the National 973 Program of China (no. 2013CB329804), the National Natural Science Foundation of China (no. 61229301), and the Program for Changjiang Scholars and Innovative Research Team in University (PCSIRT) of the Ministry of Education, China (no. IRT13059).

References


J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, M.-C. Hsu, Prefixspan: Mining sequential patterns efficiently by prefix projected pattern growth, in: Proceedings of the Twenty-ninth IEEE International Conference on Data Engineering (ICDE), 2001, p. 0215.


J. Wang, J. Han, BIDE, Efficient mining of frequent closed sequences, in: Proceedings of the Twentieth International Conference on Data Engineering, 2004, pp. 79–90.


X. Yan, J. Han, R. Afhar, CloSpan: Mining closed sequential patterns in large datasets, in: Proceedings of the SIAM International Conference on Data Mining, 2003, pp. 166–177.


