GraphSum: discovering correlations among multiple terms for graph-based summarization

GraphSum: discovering correlations among multiple terms for graph-based summarization

Elena Baralis, Luca Cagliero*, Naeem Mahoto

Dipartimento di Automatica e Informatica, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129, Torino, Italy

Alessandro Fiori

IRC@C: Institute for Cancer Research at Candiolo, Str. Prov. 142 Km. 3.95 10060 - Candiolo (TO) - Italy

Abstract

Graph-based summarization entails extracting a worthwhile subset of sentences from a collection of textual documents by using a graph-based model to represent the correlations between pairs of document terms. However, since the high-order correlations among multiple terms are disregarded during graph evaluation, the summarization performance could be limited unless integrating ad-hoc language-dependent or semantics-based analysis.

This paper presents a novel and general-purpose graph-based summarizer, namely GraphSum (Graph-based Summarizer). It discovers and exploits association rules to represent the correlations among multiple terms that have been neglected by previous approaches. The graph nodes, which represent combinations of two or more terms, are first ranked by means of a PageRank

*Corresponding author. Tel.: +39 0110907084 Fax: +39 0110907099.

Email addresses: elena.baralis@polito.it (Elena Baralis), luca.cagliero@polito.it (Luca Cagliero), naeem.mahoto@polito.it (Naeem Mahoto), alessandro.fiori@ircc.it (Alessandro Fiori)

Preprint submitted to

strategy that discriminates between positive and negative term correlations. Then, the produced node ranking is used to drive the sentence selection process.

The experiments performed on benchmark and real-life documents demonstrate the effectiveness of the proposed approach compared to many state-of-the-art summarizers.
Keywords: Multi-document summarization, Text mining, Association rule mining, Graph ranking

---

1. Introduction

Nowadays Internet provides access to a huge number of electronic textual documents. However, to extract useful information from the accessed documents users commonly have to peruse tens and tens of pages. To ease the knowledge discovery process, a significant research effort has been devoted to studying and developing automated summarization tools, which produce a succinct overview of the most relevant document content, i.e., the summary.

The multi-document summarization task entails generating a summary of a collection of textual documents. Summarizers may be classified as: (i) sentence-based, if they partition the documents into sentences and select the most informative ones [8, 27, 42, 43], or (ii) keyword-based, if they detect salient document keywords [15, 21]. To tackle the sentence-based summarization problem, several research efforts have been devoted to adopting information retrieval and/or data mining techniques, such as clustering [29, 42, 43], probabilistic or co-occurrence-based strategies [11, 35], or graph-based algorithms [16, 41, 45, 46]. Specifically, graph-based summarization focuses on

building a graph in which nodes represent either single terms or document sentences, whereas an edge weighs the strength of the relationship between a pair of nodes. Based on the assumption that the nodes that are connected to many other nodes are more likely to carry salient information, the sentence selection process is driven by a graph index that is produced by an established ranking algorithm (e.g., PageRank [7], HITS [20]). However, graph-based summarizers sometimes do not achieve fairly high performance because of the intrinsic limitations of the graph-based models. In fact, on the one hand, since they do not consider the underlying correlations among multiple terms, some relevant facets of the analyzed data could be disregarded. On the other hand, some of the considered term correlations are potentially misleading, because they represent negatively correlated associations among terms (i.e., combinations of terms that occur less than expected in the analyzed data). Hence, there is a need for novel graph-based summarizers that also consider the correlations among multiple terms and the differences between positive and negative term correlations.

This paper presents GraphSum (Graph-based Summarizer), a new multi-document summarizer that relies on a graph-based summarization strategy. The proposed approach entails building and evaluating a correlation graph, in which the graph nodes represent sets of document terms of arbitrary size, whereas the edges have a weight that indicates the strength of the correlation between the corresponding pair of nodes. Specifically, two nodes are connected by a bidirectional edge if their corresponding terms co-occur frequently and are strongly correlated with each other (either positively or negatively) in the analyzed collection. The correlations among multiple terms

are extracted using an established data mining technique, i.e., association rule mining [1]. To guarantee the quality of the generated model and make the extraction task computationally tractable, the mining process is driven by the following constraints: (i) A minimum support threshold [1] (i.e., a minimum frequency of occurrence of the mined term combinations in the source data) and (ii) a maximum negative and a minimum positive correlation thresholds [38] (i.e., the minimum significance levels of the mined data correlations).

The selection of the most representative sentences is driven by the correlation graph. To this purpose, GraphSum adopts a variant of the popular PageRank ranking algorithm [7], which also discriminates between positive and negative term set correlations. The key idea is to mitigate the propagation of the PageRank scores through negatively correlated links in order to assign, on average, a higher rank to the nodes that have no negatively correlated links. The final summary will consist of the subset of sentences that best cover the previously generated model. In such a way, summaries are less likely to contain the sentences in which a combination of terms are negatively correlated with each other.

GraphSum does not rely on advanced semantics-based models (e.g., ontologies or taxonomies) to perform document analysis. Furthermore, it exploits only two basic language-dependent steps, i.e., lemmatization and stopword elimination. Hence, the proposed summarizer is potentially applicable to documents coming from rather different application contexts. A variety of experiments have been conducted on benchmark document collections and on a subset of real-life news articles which have been published by the most

renowned newspapers. The results demonstrate that GraphSum performs better than many state-of-the-art summarizers (e.g., [5, 11]) in terms of the most commonly used Rouge evaluation scores [21].

The main contributions of the paper could be summarized as follows:

- Using association rules in graph-based summarization to represent correlations among multiple terms,
- analyzing positive and negative term correlations separately for document summarization purposes, and
- performing no semantics-based analysis and a minimal number of language-dependent steps to preserve the flexibility and portability of the proposed approach.

The paper is organized as follows. Section 2 compares our work with previous approaches. Section 3 thoroughly describes the GraphSum summarizer. Section 4 experimentally evaluates our approach on a variety of multi-document collections. Finally, Section 5 draws conclusions and presents future work.

2. Related work

Many research efforts have been devoted to addressing the document summarization problem. For example, graph-based summarizers generate and exploit a graph to select the most relevant document sentences or keywords. Among them, sentence-based approaches (e.g., [4, 16, 41, 45, 46]) model the document sentences as graph nodes, whereas the graph edges are weighted

by a sentence similarity score. An indexing algorithm is commonly used to rank the sentences based on their relative authoritativeness in the generated graph. For example, in [16] the document sentences are ranked according to their eigenvector centrality, which is computed on the sentence linkage matrix by the established PageRank algorithm [7]. A similar graph-based model has also been adopted in [3, 4] to tackle the summarization problem by combining complex network analysis [31] with language-dependent text processing. In parallel, keyword-based summarizers that are based on graph indexing strategies have also been proposed [22, 45, 46]. For example, in [22] the graph nodes represent single keywords, which are indexed by the HITS algorithm [20]. A similar approach has been adopted in [45, 46] to address Web page summarization driven by the user-generated content coming from social networks. Unlike all of the above-mentioned approaches, our summarizer discovers association rules from the analyzed document to also represent the correlations among multiple terms in the graph-based model.

A parallel effort has been devoted to using clustering algorithms for summarizing documents [29, 42, 43]. For example, in [43] a cluster represents a group of sentences and the summary consists of the subset of the best cluster representatives (e.g., the centroids or the medoids). In contrast, MEAD [29] clusters documents rather than single sentences. For each cluster, it selects a worthwhile subset of sentences by considering (i) the tf-idf value [21] of the sentence terms, (ii) the sentence relevance within the cluster, and (iii) the sentence length. The approach presented in [42] addresses the issue of dynamic summary updating by exploiting an incremental clustering algorithm. Once a set of documents is added/removed from the analyzed collection, the

previously generated summary is updated without the need for recalculating the whole clustering model. However, clustering-based approaches appear to be less suitable for being applied to collections of documents that range over the same subject. Hence, their effectiveness is rather limited unless integrating advanced semantics-based models (e.g., ontologies or taxonomies) or language-dependent analysis.

The use of linear programming algorithms and probabilistic models for summarization purposes has also been investigated [12, 17, 37]. For example, in [17, 35, 37] the authors formalize the sentence selection task as a min-max optimization problem and tackle the problem by means of linear programming techniques. To effectively address the summarization problem in a multilingual context in [11] and [35] the authors exploit Markov Hidden Model and Singular Value Decomposition techniques, respectively, to extract the most representative document sentences. Unlike [11, 17, 37], the summarizer presented in this paper relies on a general-purpose, graph-based approach that discovers and exploits high-order correlations among multiple document terms.

Frequent itemset and association rule mining are widely exploratory techniques [1] to discover relevant correlations among data. They find application in many application domains (e.g., market basket analysis [1], biological data analysis [10]). A significant research effort has been devoted to summarizing transactional data by means of frequent itemsets [19, 24]. Since the result is a worthwhile subset of itemsets, rather than a subset of document sentences, the aim of the above-mentioned approaches is radically different from the one of this paper. A preliminary attempt to use frequent itemsets for

summarizing documents has been made in [5]. The authors select the most informative sentences by considering a compact subset of frequent itemsets, which have been extracted by means of an entropy-based strategy [24]. Unlike [5], GraphSum is a graph-based approach that discovers and exploits association rules to also consider the high-order correlations among multiple terms. To effectively discriminate between reliable and unreliable term correlations, a variant of an established graph ranking strategy [7] is used to identify the most authoritative term combinations.

3. The Graph-based Summarizer

GraphSum (Graph-based Summarizer) is a novel graph-based multi-document summarizer. Figure 1 summarizes its main steps, which are briefly described below:

- **Text processing.** The raw textual documents are prepared for the subsequent mining steps. To this aim, two established preprocessing steps, i.e., lemmatization and stopword elimination, are applied.

- **Correlation graph mining.** Frequent itemsets, which represent correlations among terms, are extracted from a transactional representation of the document collection and combined in a graph-based model. The graph edges are weighted by an established quality index, i.e., the lift [38], which measures the strength of the association between a pair of term sets. To improve the quality of the generated model, the model includes only the associations among terms that (i) occur frequently and (ii) have a strong (either positive or negative) correlation in the analyzed collection.

- **Graph indexing.** A variant of the popular PageRank [7] indexing algorithm is used to evaluate graph node relevance. The nodes that are positively correlated with many others placed first. In contrast, the nodes that are negatively correlated with their neighbors are, on average, penalized.

- **Sentence selection.** To generate the summary of the document collection, the subset of sentences that best covers the previously indexed graph is selected.

A more thorough description of each step is given in the following sections.

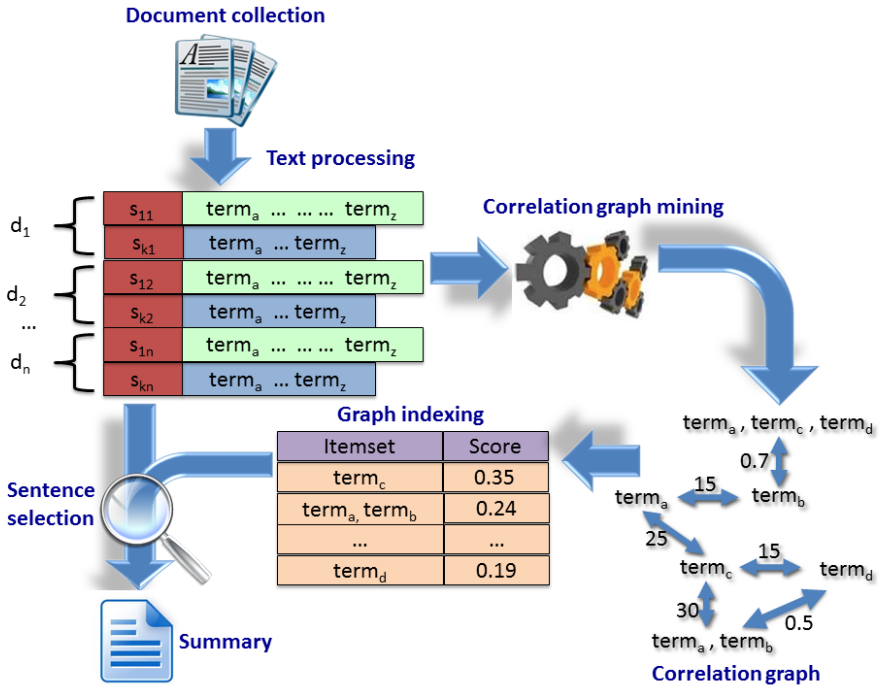


Figure 1: The GraphSum summarizer

3.1. Text processing

The raw textual content is commonly unsuitable for use in itemset and association rule mining. Hence, this processing step aims at preparing the document collection to the next mining steps. To preserve the portability and usability of the proposed system in different application contexts, it performs only two, very basic, language-dependent analysis. Specifically, stopword elimination searches for a matching between the text words and the words that are contained in a (language-dependent) stopword corpus. For example, high-frequency words like *The* and *To* are eliminated. The goal is to filter out of the documents before further processing the words that usually have little lexical content, because the mining process fails to distinguish them from other texts. To perform stopword elimination, we adopted the Natural Language Toolkit (NTLK) stopword corpus [23]. Furthermore, a lemmatization algorithm, which is based on Wordnet [44], is used to reduce the document words to their corresponding lemma. For example, the word *Analyzes* is reduced to its corresponding lemma *Analyze*.

The preprocessed collection complies with the bag-of-word (BOW) representation [21]. Specifically, each document $d_k \in D$ consists of a set of sentences $S_k=\{s_{1k}, \ldots, s_{zk}\}$, where each sentence contains an unordered subset of lemmas, which hereafter will be denoted as *terms*.

After the preprocessing steps, the document collection can be modeled as a transactional dataset. Transactions are an established data format that is commonly used for itemset and association rule mining [1]. A transactional dataset is a set of transactions, where each transaction is a set of items. For example, in the context of market basket analysis [1] a transactional dataset

contains the list of all the customer purchases, in which each transaction consists of a set of bought items (i.e., the market basket of a customer). For our purposes, we tailor the document collection to a transactional data format, in which each document sentence s_{jk} is a transaction tr_{jk} that contains all the distinct terms (i.e., the items) that occur in the BOW representation of s_{jk}. A more formal definition follows.

Definition 1. Transactional representation of a document collection. *Let D=$\{d_1, \ldots, d_N\}$ be a document collection. The transactional representation T of D contains, for every sentence s_{jk} such that $s_{jk} \in d_k \in D$, a transaction tr_{jk} that is composed of distinct terms w_q, where w_q is the q-th term in s_{jk}.*

For example, consider the document collection D in Table 1. It contains three documents, which consist of two sentences each. The corresponding transactional representation, which is reported in Table 2, contains 6 transactions. For each sentence, the corresponding set of non-repeated terms is reported. For instance, the first sentence of the document d_1 contains the distinct terms *Data*, *Analysis*, and *Mine*.

Table 1: Running example D before text processing.

Document	**Content**
d_1	This is about data analysis and data mining. In particular, it analyzes contextual information.
d_2	Information is hidden in data. However, through the analysis of the context we may enrich data.
d_3	Processing data is useful: an in-depth analysis produces actionable information.

Table 2: Running example D after text processing.

Document	Sentence ID	Sentence
d_1	1	Data, Analysis, Mine
	2	Analyze, Context, information
d_2	3	Information, Data, Hide
	4	Analysis, Context, Data, Enrich
d_3	5	Data, Processing, Useful
	6	Depth, Analysis, Information, Action

3.2. Correlation graph mining

This block aims at representing the most significant term correlations hidden in the analyzed document collection by means of a graph-based model, called correlation graph. It takes as input the transactional representation T of the analyzed document collection D=$\{d_1, \ldots, d_N\}$ and generates a graph G that will be used in the subsequent step to extract the most relevant sentences.

The correlation graph mining task entails the following steps: (i) Frequent itemset mining, (ii) term set correlation estimate, and (iii) graph generation. A more thorough description of each step is given below.

3.2.1. Frequent itemset mining

This step focuses on discovering recurrent combinations of terms, in the form of frequent itemsets [1], from the transactional representation of the document collection. In our context, a k-itemsets (i.e., an itemset of length k) is defined as a set of k distinct terms. Two itemsets are *disjoint* if they have no term in common. An itemset is said to *cover* a given transaction (sentence) tr_{jk} if all of its terms are contained in tr_{jk}. Given an itemset I and the transactional representation T of a collection D, the *support* of I in T is

defined as the ratio between the number of transactions in T that are covered by I and the total number of transactions in T. Every itemset for which its support value in D is equal to or higher than a given minimum support threshold *minsup* is said to be *frequent* in T. Recalling the running example (see Table 2), {Data, Analysis} is a 2-itemset that covers 2 transactions in T. Hence, its support value is $\frac{2}{6}$.

Given a transactional representation T of the analyzed document collection D and a minimum support threshold *minsup*, the frequent itemset mining process entails extracting all of the frequent itemsets in T. To accomplish the itemset mining task, GraphSum exploits an implementation [6] of the traditional Apriori rule mining algorithm [2]. However, different and more efficient itemset miners could be easily integrated as well.

3.2.2. Term set correlation estimate

This step aims at evaluating the significance of the previously extracted term correlations. An established approach to discover and evaluate associations between pairs of itemsets is *association rule mining* [1]. An association rule is an implication $A \rightarrow B$, where A and B are disjoint itemsets that have been extracted from the source collection. A more formal definition follows.

Definition 2. Association rule. *Let A and B be two disjoint itemsets. An association rule is represented in the form $A \rightarrow B$, where A and B are denoted as rule body and head, respectively. The length of $A \rightarrow B$ is defined as the length of $A \cup B$.*

A and B are also denoted as the antecedent and consequent of the association rule $A \rightarrow B$, respectively.

Many quality measures could be exploited to select the most interesting association rules [38]. GraphSum integrates three widely used rule quality measures, i.e., support, the confidence, and lift. Their definitions are given below.

Definition 3. Association rule support. *Let $A \rightarrow B$ be an association rule. Its support $s(A \cup B)$ is defined as the support of the itemset $A \cup B$.*

The support is the prior probability of A and B (i.e., its observed frequency) in the source dataset.

Definition 4. Association rule confidence. *Let $A \rightarrow B$ be an association rule. Its confidence $c(A \cup B)$ is given by $\frac{s(A \cup B)}{s(A)}$.*

The confidence of a rule $A \rightarrow B$ is the conditional probability of occurrence of B given A. It measures the strength of the implication. For example, recalling the running example in Table 2, the association rule $\{\text{Data}\} \rightarrow \{\text{Analysis}\}$ has a support equal to $\frac{2}{6}$ in T and a confidence equal to $\frac{2}{4}$, because the itemset {Data, Analysis} occurs twice in T, whereas the implication $\{\text{Data}\} \rightarrow \{\text{Analysis}\}$ holds in half of the cases.

However, measuring the strength of a term set correlation in terms of rule confidence could be misleading [38]. In fact, when the rule consequent has a relatively high support value, the corresponding rule could be characterized by a high confidence even if its actual strength is relatively low. To overcome this issue, the lift (or correlation) measure [38] could be used, rather than the confidence index, to measure the (symmetric) correlation between the body and the head of the extracted rules.

Definition 5. Association rule lift. *Let $A \rightarrow B$ be an association rule. Its lift l is given by*

$$\text{lift(A, B)} = \frac{\text{c}(A \rightarrow B)}{\text{s(B)}} = \frac{\text{s}(A \rightarrow B)}{\text{s(A)s(B)}} \tag{1}$$

where $\text{s}(A \rightarrow B)$ *and* $\text{c}(A \rightarrow B)$ *are the rule support and confidence, respectively, and* $\text{s}(A)$ *and* $\text{s}(B)$ *are the supports of the rule antecedent and consequent.*

If lift(A,B) is equal to or close to 1, then the itemsets A and B are not correlated with each other, i.e., they are statistically independent. Lift values significantly below 1 show negative correlation, whereas values significantly above 1 indicate a positive correlation between the itemsets A and B, meaning that the implication between A and B holds more than expected in the source data.

Since the interest of the statistically independent term associations is marginal in our context of analysis, GraphSum only considers the frequent and strongly correlated associations between pairs of term sets. Specifically, it selects only the association rules for which:

- the support value is equal to or higher than a minimum support threshold *minsup*, and
- the lift value is either in the range (0,$max^{-}lift$] or higher than or equal to $min^{+}lift$, where $max^{-}lift$ and $min^{+}lift$ are the maximum negative and the minimum positive correlation thresholds, respectively.

The contribution of the positive and negative term correlations will be differentiated during the summarization process, as described in the following

sections.

The association rule extraction task is traditionally accomplished by first generating all the possible subsets of the extracted frequent itemsets and then evaluating the candidate associations [2]. GraphSum adopts the same strategy and also exploits the symmetry of the lift measure [38] to avoid generating all the possible candidates. Specifically, since lift(A,B)=lift(B,A), to evaluate the interest of the correlation between the pair of term sets A and B GraphSum exclusively considers distinct pairs of disjoint frequent itemsets A and B such that the union $A \cup B$ (and, hence, the rule $A \rightarrow B$) is frequent with respect to the minimum support threshold *minsup*.

3.2.3. Correlation graph generation

To represent the most significant correlations among multiple terms a graph-based model, named *correlation graph*, is generated. The concept of correlation graph is formalized as follows.

Definition 6. Correlation graph. *Let T be a transactional representation of a document collection D and minsup, $max^{-}lift$, and $min^{+}lift$ three non-negative numbers. Let $\mathcal{I}$ be the set of frequent itemsets that were mined from T by enforcing a minimum support threshold minsup. A correlation graph G that is built on T is a bidirected graph for which the nodes are frequent itemsets (term sets) in $\mathcal{I}$, whereas its edges link arbitrary node pairs A and B such that either lift(A,B) $\in$ (0,$max^{-}lift$] or lift(A,B) $\geq$ $min^{+}lift$. The bidirected edges are weighted by the corresponding rule lift value.*

As an example, suppose setting *minsup* to 1%, $max^{-}lift$ to $\frac{4}{5}$, and $min^{+}lift$ to 10. Since the support of {Data, Analysis} in the running example dataset

is 33% and lift({Data},{Analysis})=$\frac{3}{5}$, then the corresponding correlation graph G contains two distinct nodes, which are related to the terms {Data} and {Analysis} and are linked by an edge with weight $\frac{3}{5}$. Note that, by the Apriori principle [2], both {Data} and {Analysis} are frequent with respect to the support threshold. Unlike previous approaches (e.g., [45, 46, 22]), the graph nodes could also represent a subset of terms with size higher than one (e.g., {Analysis, Context}). Some extracts of the correlation graphs that were mined from real-life documents are reported in Section 4.

3.3. Graph indexing

The term sets that are contained in the correlation graph typically do not have the same importance in the analyzed documents. To measure the relevance of the graph nodes that were extracted from the source collection, GraphSum adopts a variant of the traditional PageRank graph ranking algorithm [7].

PageRank [7], which has been brought to success by its adoption in the popular Google search engine, has already been applied in different application contexts to detect the most authoritative elements in a large collection. The key idea behind the PageRank algorithm can be summarized as follows. A weighted edge that connects the graph nodes A and B can be thought as a vote assigned by A to B. The higher the sum of all of the weights that are associated with any incoming link is the higher the relative importance of B in the graph becomes. PageRank models the graph as a Markov Chain model, in which Random Walks describe the probabilities of moving between the graph nodes. Specifically, since PageRank focuses on mimic the process of Web surfing, it analyzes the effect of simple Random Walks that incorpo-

rate random jumps on the graph. The probabilities that are evaluated in a stationary state of the Random Walk are those that should be reached after an infinite walk on the graph. Since, in our context, a node represents a specific term set, the stationary probabilities describe the expected probabilities of occurrence of each term set in the analyzed documents. In PageRank, the authority of a graph node is estimated by an iterative algorithm that aggregates the transitional probabilities between all graph nodes until a steady state is reached. Such iterative process results in a graph node ranking that is based on an authority score, called the PageRank score.

Formally speaking, the PageRank score $PR(N_i)$ of a graph node N_i can be approximated as follows [7]:

$$PR(N_i) = \frac{(1-d)}{|N|} + d \cdot \sum_{k=1}^{e} \frac{PR(N_k)}{C(N_k)} \tag{2}$$

where $|N|$ is the total number of graph nodes, e is the number of edges incoming into N_i, $PR(N_k)$ is the PageRank score of an arbitrary N_i's neighbor N_k, $C(N_k)$ is the outgoing degree of the node N_k, and $d \in [0, 1]$ is a damping factor that weights the PageRank score propagation from one node to another, which is usually set to 0.85 [7].

Our goal is to differentiate between negative and positive term set correlations in order to assign, on average, a lower PageRank score to the nodes that are negatively correlated to one another. To achieve this goal, we adopt the following variant of the traditional PageRank score given in Formula 2:

$$PR(N_i) = \frac{(1-d)}{|N|} + d \cdot (\sum_{k=1}^{e} \frac{1}{\sqrt{C^{-}(N_k)}} \frac{PR(N_k)}{C(N_k)}) \tag{3}$$

where $C^{-}(N_k)$ is the outgoing degree of a node N_k that is computed by considering only the negatively correlated edges. Note that the penalization factor $\frac{1}{\sqrt{C^{-}(N_k)}}$ mitigates the PageRank score propagation from the node N_k that have a significant number of negatively correlated neighbors to N_i. A penalization score corresponds to a rescaling of the transition matrix terms that are related to a negatively correlated node pair. Such rescaling smooths down the probability of randomly choosing, in a Random Walk, a negatively correlated link to follow. The penalization is null for the nodes that are linked to their neighborhood only through positively correlated links.

3.4. Sentence selection

GraphSum exploits the indexed correlation graph to evaluate and select the most relevant document sentences. To this aim, two main sentence features are considered: (i) the term relevance in terms of the corresponding PageRank indexes and (ii) the correlation graph coverage.

In the following we formalize both sentence relevance and coverage.

Sentence relevance score. The relevance score of a sentence s_{jk} in the document collection measures the significance of a sentence in terms of the authority of its contained term sets (nodes) in the correlation graph. It is defined as the normalized sum of the PageRank scores that have been assigned to each term set that occurs in the sentence:

$$SR(s_{jk}) = \frac{\sum_{i \mid n_i \subseteq t_{jk}} PR_{ik}}{N_{t_{jk}}} \tag{4}$$

where t_{jk} is the transaction that is associated with the sentence s_{jk}, $\sum_{i \mid n_i \in t_{jk}} PR_{ik}$ is the sum of the PageRank scores PR_{ik} that are associated with every cor-

relation graph node n_i covering t_{jk}, and $N_{t_{jk}}$ is the total number of nodes that cover t_{jk}.

Sentence coverage. The sentence coverage indicates how much a sentence s_{jk} is pertinent to the association rule graph G. To define the sentence coverage, we first associate with each sentence $s_{jk} \in D$ a binary vector SC_{jk}=$\{sc_1, \ldots, sc_{|N|}\}$, which hereafter will be denoted as *sentence coverage vector*, where $|N|$ is the number of nodes that are contained in the correlation graph G, and $sc_i = \mathbf{1}_{tr_{jk}}(n_i)$ indicates whether a term set n_i covers or not tr_{jk} (see Section 3.2.1). Formally speaking, given an arbitrary term set n_i contained in G, $\mathbf{1}_{tr_{jk}}$ is an indicator function that is defined as follows:

$$\mathbf{1}_{tr_{jk}}(n_i) = \begin{cases} 1 & \text{if } n_i \subseteq tr_{jk}, \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

The coverage of a sentence s_{jk} with respect to the correlation graph is defined as the number of ones in the coverage vector SC_{jk}.

To our purposes, we formalize the problem of selecting the most representative document sentences in terms of coverage and relevance scores as a set covering problem.

The set covering problem. GraphSum addresses a set covering optimization problem to select the sentences with maximal model coverage and relevance score. Specifically, we focus on selecting the minimal number of sentences that are characterized by maximal score. It applies the logic OR operator to the coverage vectors of the selected sentences, i.e., SC^*=$SC_1 \vee \ldots \vee SC_l$, and selects the subset of sentences for which the resulting binary vector con-

tains the maximal number of ones. The result of the OR operation SC^* will be denoted as the *summary coverage vector* throughout the section.

Since the set covering optimization problem is NP-hard, we tackle the problem by means of a greedy strategy similar to the one that has previously been applied in [5] in the context of document summarization. The greedy sentence selection strategy considers the sentence model coverage to be the most discriminating feature, i.e., sentences that cover the maximum number of graph nodes are selected first. At equal terms, the sentence with maximal coverage that is characterized by the highest relevance score SR is preferred. A more detailed description of the adopted greedy strategy follows.

The greedy strategy. The sentence selection algorithm relies on a step-wise procedure. At each step, it identifies the sentence s_{jk} with the best complementary vector SC_{jk} according to the current summary coverage vector SC^*, i.e., the sentence s_{jk} that covers the maximum number of graph nodes that have not already been covered by any sentence of the current summary.

A pseudo-code of the greedy selection strategy is given in Algorithm 1. It takes as input the set of sentences S, the set of sentence coverage vectors SC, and the set of sentence relevance scores SR. It generates a summary $\mathcal{SU}$ that is composed of the sentences that best cover the correlation graph. The main sentence selection steps are summarized below. The first step is variable initialization (lines 1-2). Next, the best candidate sentence is iteratively selected (lines 3-13). At each iteration, it selects, among the sentences with maximum coverage the one with maximal relevance score (see Formula 4) (line 6). Then, the selected sentence s_{best} is included in the summary $\mathcal{SU}$ (line 7) and the sentence coverage vectors are updated accordingly (lines 8-

Algorithm 1 Greedy sentence selection

Input: set of sentences S

Input: set of sentence coverage vectors SC

Input: set of sentence relevance scores SR

Output: summary $\mathcal{SU}$

1: $\mathcal{SU} = \emptyset$

2: SC^* = set_to_all_zeros() /*initialize the summary coverage vector with only zeros*/
/*Cycle until SC^* contains only 1s (i.e., until the generated summary covers all the itemsets of the model) */

3: **while** SC^* contains at least one zero **do**

4: $MaxOnesSentences$ = max_ones_sentences(S, SC) /*Select the sentences with the highest number of ones*/

5: **if** $MaxOnesSentences$ is not empty **then**

6: $s_{best} = arg\,max_{s_j \in MaxOnesSentences} SR(s_j)$ /*Select the sentence with maximum relevance score among the ones in $MaxOnesSentences$*/

7: $\mathcal{SU} = \mathcal{SU} \cup s_{best}$ /*Add the best sentence to the summary*/
/*Update the summary coverage vector SC^*. $SC_{s_{best}} \in SC$ is the sentence coverage vector associated with the best sentence s_{best} */

8: SC^*= SC^* OR $SC_{s_{best}}$ /* Set the bits associated with the term sets covered by s_{best} to one*/
/*Update the sentence coverage vectors in SC^*/

9: **for all** SC_i in SC **do**

10: $SC_i = SC_i$ AND $\overline{SC^*}$ /*Set the bits of SC_i associated with the term sets that are already covered by the summary to zero*/

11: **end for**

12: **end if**

13: **end while**

14: **return** $\mathcal{SU}$

11). The updating step excludes from the subset of coverable graph nodes the ones that have already been covered by the current summary. The procedure iterates until the graph-based model is fully covered by the summary, i.e., until the summary coverage vector contains only ones (line 3). Note that, when *minsup*>0, each frequent itemset that is contained in the model covers at least one document sentence. Hence, the sentence selection process always achieves a full graph coverage.

The experimental results, reported in Section 4.4, show that the greedy sentence selection strategy is more effective and efficient than a traditional branch-and-bound strategy [30] for summarization purposes.

4. Experimental results

We conducted an extensive experimental campaign to address the following issues: (i) A performance comparison between GraphSum and many other summarizers on benchmark documents (Sections 4.2), and (ii) an evaluation of the effectiveness of the proposed summarizer on real-life news document collections (Section 4.3), (iii) an analysis of the impact of the main system parameters and features on the GraphSum performance (Section 4.4).

All the experiments were performed on a 3.0 GHz 64 bit Intel Xeon PC with 4 GB main memory running Ubuntu 10.04 LTS (kernel 2.6.32-31). The code for GraphSum is available, for research purposes, at *http://dbdmg.polito.it/wordpress/wp-content/uploads/2013/03/GraphSum.zip*. A brief description of the analyzed document collections follows.

4.1. Document collections

We conducted experiments on (i) the benchmark dataset for the task 2 of the DUC'04 [14], i.e., the English-written benchmark collections that have been used for the Document Understanding Conference (DUC) contest on multi-document summarization and (ii) five real-life news article collections.

To the best of our knowledge, the task 2 of DUC'04 [14] is the latest Document Understanding Conference (DUC) contest on generic English-written multi-document summarization [42]. DUC'04 documents were clustered in 50 document groups. Each cluster consists of approximately 10 documents. For

each group the DUC'04 conference organizers provided one or more golden summaries, which were generated by a pool of domain experts.

To evaluate the effectiveness of the proposed summarizer in a real-life context, we also tested GraphSum on Five English-written news article collections that were retrieved from the Web from August 2011 to November 2011. Each real-life collection ranges over a different topic and consists of 10 news articles. Specifically, the list of the analyzed news topics is given below:

- **Italian austerity**: the package of austerity measures has been approved by the Italian Government to lead Italy out of its debt crisis.
- **World terrorism**: the war of the U.S.A. government against the international terrorism
- **Strauss Kahn scandal**: Dominique Strauss Kahn charged of sexual assault
- **Libya war**: The civil war in the North African state of Libya breaks out
- **Irene hurricane**: the Irene Hurricane beats down on the U.S. East Coast

From the result of a targeted query to the Google News search engine, the top-10 news articles were selected. The considered topics are representatives of different case studies, because they cover (i) Very focused news of topical interest for a short time period (e.g., U.K. riots, Strauss Kahn), (ii) averagely focused past events having side effects on future events (e.g., Irene Hurricane), and (iii) broad-spectrum and multi-faceted news (e.g, Debt crisis, Terrorism).

The news collections are available for research purposes at *http://dbdmg.polito.it/wordpress/wp-content/uploads/2013/03/news_articles.zip*.

4.2. DUC'04 benchmark dataset summarization

We analyzed and compared the GraphSum performance on the DUC'04 benchmark collections with that of (i) the 35 summarizers that were submitted to the DUC'04 conference, (ii) the 8 summaries that were generated by the humans and made available by the DUC'04 organizers (beyond the golden summaries), (iii) two widely used opensource text summarizers, i.e., the Open Text Summarizer (OTS) [33] and TexLexAn [39], and (iv) a recently proposed itemset-based summarizer [5], named ItemSum (Itemset-based Summarizer). For the DUC'04 competitors, we considered the best results that were given by the DUC'04 organizers [14], whereas for the other competitors we used the best algorithm configuration that has been suggested by the respective authors. Specifically, for ItemSum [5] the best configuration is minimum support threshold $minsup$=3% and model size ms=12. For GraphSum we set, as *standard* configuration, the minimum support threshold $minsup$ to 3%, the maximum negative correlation threshold $max^{-}lift$ to 0.6, and the minimum positive correlation threshold $min^{+}lift$ to 15. Section 4.4 thoroughly analyzes the impact of the GraphSum input parameters on the summarization performance.

To perform a quantitative comparison of the summarizer performance, we used the ROUGE toolkit [21], because it has been adopted as official DUC'04 tool[1]. It measures the quality of a summary by counting the unit overlaps between the candidate summary and the golden summaries. The summarizer that achieves the highest ROUGE scores is considered to be the most effec-

[1]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

tive one. To perform a fair comparison the generated summaries have been preliminary normalized before using the ROUGE tool by truncating each of them at 665 bytes (rounding the number down in case of straddled words). Hence, the summary size, in terms of word count, is approximately the same for all the summaries. Several automatic evaluation scores are implemented in ROUGE. For the sake of brevity, we only report the most representative ones [21], i.e., ROUGE-2 and ROUGE-SU4. Similar results were achieved for the other scores.

Table 3 reports the results that were achieved on the DUC'04 benchmark datasets by GraphSum, ItemSum, OTS, TexLexAn, the 8 humanly generated summaries, and the 10 most effective summarizers that have been presented in DUC'04. For the top ranked DUC'04 summarizer, i.e., CLASSY, we reported all of its submitted versions (i.e., peer65, peer66, and peer67). Note that the most recent CLASSY summarizer version has been presented in [11]. It is just an extension of its preliminary DUC'04 version [12] that is also able to cope with not English-written documents. Since CLASSY [11] ranked first both in the DUC'04 contest [14] and in the multi-lingual TAC'11 contest [40] on the English-written document collections, it can be considered to be the most effective state-of-the-art summarizer on English-written documents. To validate the statistical significance of the GraphSum performance improvement against its competitors, we used the paired t-test [13] at 95% significance level for all of the evaluated measures.

GraphSum performs significantly better than ItemSum, OTS, TexLexAn for all of the tested measures. Furthermore, it performs significantly better than all the DUC'04 competitors in terms of ROUGE-2 and ROUGE-SU4

Table 3: DUC'04 Collections. Comparisons between GraphSum and the other approaches. Statistically relevant differences in the comparisons between GraphSum (standard configuration) and the other approaches are starred.

Summarizer		ROUGE-2			ROUGE-SU4		
		R	Pr	F	R	Pr	F
TOP RANKED DUC'04 PEERS	peer67	0.089*	0.095*	0.092*	**0.015**	0.017*	0.016*
	peer120	0.076*	0.103*	0.086*	**0.015**	0.018*	0.016*
	peer65	0.087*	0.091*	0.089*	**0.015**	0.016*	0.015*
	peer66	0.086*	0.093*	0.089*	0.013	0.014*	0.014*
	peer121	0.071*	0.085*	0.077*	0.012*	0.014*	0.013*
	peer11	0.070*	0.087*	0.077*	0.012*	0.015*	0.012*
	peer44	0.075*	0.080*	0.078*	0.012*	0.013*	0.012*
	peer81	0.077*	0.080*	0.078*	0.012*	0.012*	0.012*
	peer124	0.078*	0.082*	0.080*	0.011*	0.012*	0.011*
	peer35	0.081*	0.085	0.083*	0.010*	0.011*	0.011*
DUC'04 HUMANS	A	0.088*	0.092*	0.090*	0.009*	0.010*	0.010*
	B	0.091	0.096	0.093*	0.013	0.013	0.013*
	C	0.094	0.102	0.098	0.011*	0.012*	0.012*
	D	0.100	**0.106**	0.102	0.010*	0.010*	0.010*
	E	0.094	0.099	0.097	0.011*	0.012	0.012*
	F	0.086*	0.090	0.088*	0.008*	0.009*	0.009*
	G	0.082*	0.087*	0.084*	0.008*	0.008*	0.007*
	H	**0.101**	0.105	**0.103**	0.012*	0.013*	0.012*
OTS		0.069*	0.079*	0.074*	0.008*	0.009*	0.009*
texLexAn		0.059*	0.068*	0.063*	0.006*	0.007*	0.007*
ItemSum		0.083*	0.085*	0.084*	0.012*	0.014*	0.014*
GraphSum		**0.093**	**0.099**	**0.097**	**0.015**	**0.021**	**0.019**

F1-measure. Although it does not exploit neither advanced linguistic processing steps nor semantics-based analysis, GraphSum performs as good as the top-ranked DUC'04 summarizers (i.e., CLASSY and Peer120) in terms of ROUGE-SU4 Recall.

GraphSum and CLASSY are the only summarizers that, in some cases, perform better than the humans. More specifically, both of them outperform

the humans in terms of ROUGE-SU4 F1-measure. GraphSum performs significantly better than 4 out of 8 humans in terms of ROUGE-2 F1-measure, whereas the best CLASSY summarizer's version (peer67) outperforms only 2 out of 8 humans. Hence, GraphSum appears to be, on average, more effective than CLASSY on the DUC'04 collections.

4.3. Real-world news summarization

This section summarizes the results that were achieved by GraphSum on the real-life news document collections (see Section 4.1). Specifically, Section 4.3.1 reports a qualitative comparison between the generated summaries, whereas Section 4.3.2 evaluates the performance of both GraphSum and its competitors in terms of ROUGE scores [21] on the same documents.

4.3.1. Summary comparison

We performed a qualitative comparison between the summaries that were generated by GraphSum and the subset of competitors for which a publicly available code version is available (i.e., ItemSum, OTS, TexLexAn).

Tables 4 reports the summaries that were generated from the Irene Hurricane collection, which has been chosen as representative of all the other news collections. For further appreciation, the complete set of results that were generated from the news collections is available at *http://dbdmg.polito.it/wordpress/wp-content/uploads/2013/03/news_results.zip*.

For GraphSum and the other summarizers we used the algorithm configurations that are reported in Section 4.2.

GraphSum appears to produce the most focused summary, because the summary provides a concise yet informative description of the news content.

Figure 2 reports an extract of the correlation graph that were generated from the Irene Hurricane collection. $\{Rain\} \leftrightarrow \{Torrential\}$, $\{Rain\} \leftrightarrow \{Heavy\}$, and $\{Rain\} \leftrightarrow \{Water,Creek\}$ are three examples of strong term correlations (e.g., lift($\{Rain\}$, $\{Heavy\}$) = 46). The same pair of terms occurs in the first two sentences of the document summary, because the terms are deemed to be significant for summarization purposes. In contrast, the association $\{Town\} \leftrightarrow \{Flood\}$, which occurs in the TexLexAn's summary, is disregarded by GraphSum, because it is characterized by a negative correlation value (lift=0.45). The summaries that were generated by the other competitors seem to be rather generic and partially redundant. For example, the summary that were generated by ItemSum also contains the following uninteresting content: *"If this is what it means to live in the nanny state, I'm very content" Krasnow said.*

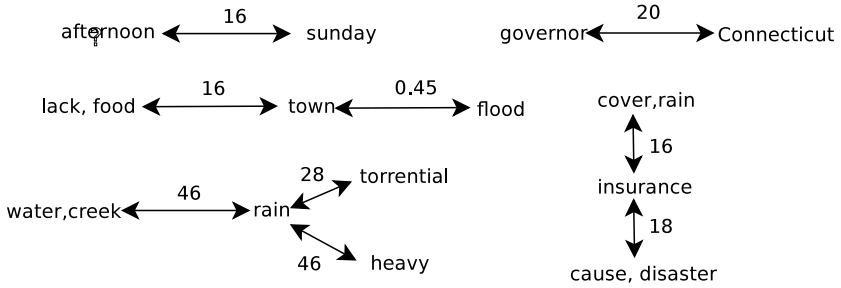


Figure 2: Irene Hurricane collection. Portion of the extracted correlation graph.

4.3.2. Performance comparison

We compared the GraphSum performance on the real-life news articles with that achieved by the following publicly available summarizers: (i) the Open Text Summarizer (OTS) [33], (ii) TexLexAn [39], and (iii) ItemSum [5].

Table 4: Summary examples. Irene hurricane news collection

Method	Summary
GraphSum	New York was pounded by heavy winds and torrential rain on Sunday morning as Hurricane Irene bore down on the city, threatening to cause flash flooding and widespread damage in the US's most populous city. It's one of several towns in states such as New Jersey, Connecticut, New York, Vermont and Massachusetts dealing with the damage of torrential rain and flooding spawned by Hurricane Irene.
ItemSum	New York was pounded by heavy winds and torrential rain on Sunday morning as Hurricane Irene bore down on the city, threatening to cause flash flooding and widespread damage in the US's most populous city. "If this is what it means to live in the nanny state, I'm very content," Krasnow said.
OTS	As emergency airlift operations brought ready-to-eat meals and water to Vermont residents left isolated and desperate, states along the Eastern Seaboard continued to be battered Tuesday by the after effects of Irene, the destructive hurricane turned tropical storm. Dangerously-damaged infrastructure, 2.5 million people without power and thousands of water-logged homes and businesses continued to overshadow the lives of residents and officials from North Carolina through New England, where the storm has been blamed for at least 44 deaths in 13 states.
TexLexAn	As emergency airlift operations brought ready-to-eat meals and water to Vermont residents left isolated and desperate, states along the Eastern Seaboard continued to be battered Tuesday by the after effects of Irene, the destructive hurricane turned tropical storm. Search-and-rescue teams in Paterson have pulled nearly 600 people from flooded homes in the town after the Passaic River rose more than 13 feet above flood stage, the highest level since 1903.

Since the golden summaries are not publicly available for the real-life collections, to quantitatively evaluate the summarizer performance by means of the ROUGE toolkit [21] we performed, as previously done in [9], a leave-one-out cross validation. More specifically, for each category we summarized nine out of ten news documents and we compared the summaries with the remaining document, which was considered to be the golden summary at

that stage. Next, we tested all the other combinations by varying the golden summary and we computed the average performance results, in terms of precision (P), Recall (R), and F1-measure (F1), achieved by each summarizer for the ROUGE-2 and ROUGE-SU4 evaluation scores. Note that, in our context, assuming that a document is a representative summary of the rest of the collection is a good approximation, because we specifically cope with documents that range over the same topic. For the ROUGE experimental design, we used the same settings that have previously been described in Section 4.2.

Table 5 compares the average results that were obtained by GraphSum and the other summarizers. To validate the statistical significance of GraphSum performance improvement, we used again the paired t-test [13] at 95% significance level for all of the evaluated datasets and measures. The statistically relevant differences in the comparisons between GraphSum and the other approaches are starred in Tables 5. Furthermore, for each considered dataset and measure the best results are written in boldface.

The proposed approach yields promising results in terms of ROUGE scores on the real-life news articles. GraphSum always performs significantly better than all of its competitors in terms of ROUGE-SU4 Precision, Recall, and F1-measure. Furthermore, it also performs best on 4 out of 5 news collections in terms of ROUGE-2 F1-measure.

4.4. Performance analysis

In this section we analyzed the impact of the main GraphSum parameters and features on the summarization performance. Specifically, we analyzed the effect of (i) the input parameters (i.e., the minimum support and

Table 5: News Collections. Comparisons between GraphSum and the other approaches. Statistically relevant differences in the comparisons between GraphSum (standard configuration) and the other approaches are starred.

Article	Summarizer	ROUGE-2			ROUGE-SU4		
		R	Pr	F	R	Pr	F
ITALIAN AUSTERITY	OTS	**0.044**	**0.313**	**0.077**	0.014*	0.101*	0.024*
	texLexAn	0.039	0.283*	0.068	0.009*	0.068*	0.016*
	ItemSum	0.038	0.265*	0.067*	0.009*	0.065*	0.016*
	GraphSum	0.042	0.299	0.073	**0.015**	**0.108**	**0.027**
WORLD TERRORISM	OTS	0.007*	0.069*	0.013*	0.001*	**0.002**	0.002*
	texLexAn	0.008	0.073*	0.015	0.001*	0.001*	0.001*
	ItemSum	0.008	**0.118**	0.015	0.002*	0.001*	0.002*
	GraphSum	**0.010**	0.085	**0.017**	**0.004**	**0.003**	**0.005**
STRAUSS KAHN SCANDAL	OTS	0.017*	0.146*	0.030*	0.002*	0.015*	0.003*
	texLexAn	0.018*	0.162*	0.032*	0.002*	0.014*	0.003*
	ItemSum	0.019*	0.192*	0.035*	0.002*	0.019*	0.003*
	GraphSum	**0.023**	**0.198**	**0.040**	**0.004**	**0.040**	**0.008**
LIBYA WAR	OTS	**0.012**	0.134	**0.022**	0.001*	0.002*	0.001*
	texLexAn	**0.012**	**0.138**	**0.022**	0.001*	0.001*	0.001*
	ItemSum	0.005*	0.114	0.009*	0.002*	0.002*	0.001*
	GraphSum	**0.012**	0.135	**0.022**	**0.004**	**0.004**	**0.004**
IRENE HURRICANE	OTS	0.011*	0.108*	0.021*	0.002*	0.001*	0.002*
	texLexAn	0.012*	0.122*	0.023*	0.001*	0.002*	0.002*
	ItemSum	0.006*	0.153	0.012*	0.002*	0.002*	0.002*
	GraphSum	**0.016**	**0.157**	**0.029**	**0.005**	**0.005**	**0.006**

lift thresholds), (ii) the use of a greedy graph coverage strategy, and (iii) the type of extracted itemsets on the GraphSum performance.

For the experimental evaluation, we tested our summarizer on the DUC'04 collections using the same experimental setting that is described in Section 4.2.

Impact of the input parameters. In Figure 3 we plot the representative ROUGE-2 F1-measure that were achieved by GraphSum by varying the values of

support and lift thresholds.

The support threshold affects the quality of the generated summary significantly. When enforcing relatively high support thresholds (e.g., 7%), many (potentially relevant) patterns are discarded, because their support value is less than the given threshold. Hence, some relevant facets of the analyzed document collection are disregarded. On the other hand, when very low support thresholds are enforced (e.g., 0.5%) the analyzed collection could be overfitted by the generated model. Hence, the model coverage procedure is prone to errors. At medium support thresholds (e.g., 3%), the best trade-off between model specialization and generality was achieved.

The GraphSum performance appears to be slightly affected by the minimum lift thresholds when setting the maximum negative and minimum positive correlation thresholds in the ranges [0.4,0.75] and [5,25], respectively. In contrast, setting the lift thresholds out of these ranges yields, on average, a relevant performance worsening. In fact, increasing the selectivity of the lift thresholds (i.e., when $\max^{-}\text{lift}<0.4$ or $\min^{+}\text{lift}>25$) may cause the pruning of potentially useful patterns. On the other hand, the term correlations with lift value close to 1 are misleading and should be discarded.

Impact of the coverage strategy. To solve the set covering optimization problem GraphSum exploits a greedy strategy. It yields an approximated solution to the problem of selecting the subset of sentences that best covers the graph-based model. Since the set covering problem is a min-max problem, it has been converted into a linear programming problem and tackled by means of combinatorial optimization strategies. To evaluate the impact of the greedy strategy on the summarization performance, we compared the

GraphSum performance with that achieved by a variant of our summarizer, namely GraphSum$_{BB}$, that accomplishes the coverage task by exploiting a branch-and-bound algorithm [30]. In Figure 3 we compared the ROUGE-2 F1-measure achieved by GraphSum and GraphSum$_{BB}$ by varying the support and lift thresholds, respectively.

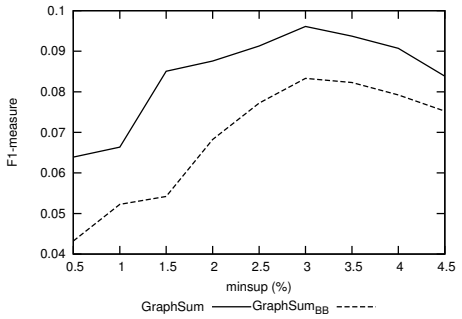
For all of the tested configurations, GraphSum performs significantly better than GraphSum$_{BB}$ and appears to be less sensitive to errors and data overfitting. Furthermore, GraphSum takes a lower execution time compared to GraphSum$_{BB}$ (e.g., at least 20% less than GraphSum$_{BB}$ for all of the considered settings).

Choice of the type of extracted itemsets. When coping with complex data distributions [18], the redundancy of the itemset mining result could worse the quality of the generated models. Hence, we also tested two variants of the GraphSum summarizer, in which maximal and closed frequent itemsets are extracted rather than the whole set of frequent itemsets. Specifically, we integrated two traditional closed [28] and maximal [32] itemset mining algorithms into the GraphSum summarizer.

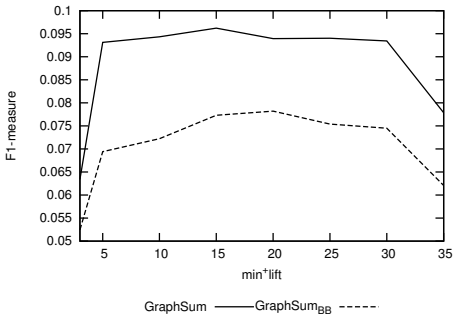
The results, not reported here for the sake of brevity, demonstrate that discarding some of the frequent itemsets on average worsens the GraphSum performance, because some of the potentially relevant data facets remain uncovered by the generated summary.

5. Conclusions and future work

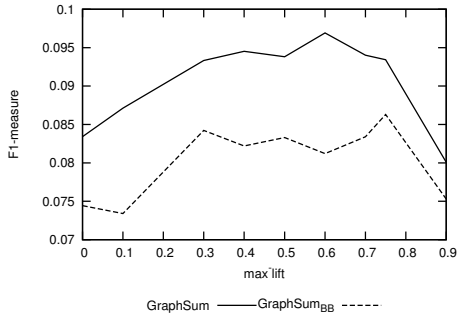
This paper presents GraphSum, a general-purpose graph-based summarizer that combines the association rules that were extracted from the



(a) Impact of *minsup*. $max^{-}lift$=0.6. $min^{+}lift$=15.



(b) Impact of $min^{+}lift$. *minsup*=3%. $max^{-}lift$=0.6.



(c) Impact of $max^{-}lift$. *minsup*=3%. $min^{+}lift$=15.

Figure 3: Parameter analysis and comparison between GRAPHSUM and GRAPHSUM$_{BB}$ in terms of Rouge-2 F1-measure

analyzed documents in a graph-based model in order to consider the correlations among multiple terms during the summarization process. To select the most informative sentences, GRAPHSUM adopts a graph ranking strategy that discriminates between positive and negative term correlations.

The experimental results demonstrate the effectiveness and usability of

the proposed summarizer on benchmark and real-life documents. GRAPHSUM performs better than many state-of-the-art approaches, including those that heavily rely on advanced semantics-based models (e.g., ontologies) or complex linguistic processing steps.

As future work, we plan to (i) adapt and evaluate the proposed summarizer into a multilingual contest (e.g., the TAC'11 contest [40]), (ii) exploit advanced document structure analysis (e.g., [34]) to perform news document summarization, (iii) integrate ontology- or dictionary-based mining strategies (e.g., [25, 26, 36]) to further improve the summarization performance.

References

[1] R. Agrawal, T. Imielinski, Swami, Mining association rules between sets of items in large databases, in: ACM SIGMOD 1993, pp. 207–216.

[2] R. Agrawal, R. Srikant, Fast algorithms for mining association rules in large databases, in: Proceedings of the 20th VLDB conference, pp. 487–499.

[3] D.R. Amancio, M.G. Nunes, O.N.O. Jr., L. da F. Costa, Extractive summarization using complex networks and syntactic dependency, Physica A: Statistical Mechanics and its Applications 391 (2012) 1855 – 1864.

[4] L. Antiqueira, O.N. Oliveira, Jr., L.d.F. Costa, M.d.G.V. Nunes, A complex network approach to text summarization, Inf. Sci. 179 (2009) 584–599.

[5] E. Baralis, L. Cagliero, S. Jabeen, A. Fiori, Multi-document summariza-

tion exploiting frequent itemsets, in: Symposium on Applied Computing (SAC'12), pp. 782–786.

[6] C. Borgelt, Efficient implementations of apriori and eclat, in: Proc. 1st IEEE ICDM Workshop on Frequent Item Set Mining Implementations (FIMI 2003, Melbourne, FL). CEUR Workshop Proceedings 90, p. 90.

[7] S. Brin, L. Page, The anatomy of a large-scale hypertextual web search engine, in: Proceedings of the seventh international conference on World Wide Web 7, pp. 107–117.

[8] G. Carenini, R.T. Ng, X. Zhou, Summarizing email conversations with clue words, in: World Wide Web Conference Series, pp. 91–100.

[9] W.T. Chuang, J. Yang, Extracting sentence segments for text summarization: a machine learning approach, in: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR '00, ACM, New York, NY, USA, 2000, pp. 152–159.

[10] G. Cong, A.K.H. Tung, X. Xu, F. Pan, J. Yang, Farmer: finding interesting rule groups in microarray datasets, in: Proceedings of the 2004 ACM SIGMOD international conference on Management of data, SIGMOD '04, ACM, New York, NY, USA, 2004, pp. 143–154.

[11] J. Conroy, J. Schlesinger, J. Kubina, P. Rankel, D. OLeary, Classy 2011 at tac: Guided and multi-lingual summaries and evaluation metrics, in: TAC'11: Proceedings of the The 2011 Text Analysis Conference, pp. 1–8.

[12] J.M. Conroy, J. Goldstein, J.D. Schlesinger, D.P. O'leary, Left-brain/right-brain multi-document summarization, in: In Proceedings of the Document Understanding Conference (DUC'04), pp. 1–9.

[13] T.G. Dietterich, Approximate statistical tests for comparing supervised classification learning algorithms, Neural Comput. 10 (1998) 1895–1923.

[14] Document Understanding Conference, HTL/NAACL workshop on text summarization, 2004.

[15] M. Dredze, H.M. Wallach, D. Puller, F. Pereira, Generating summary keywords for emails using topics, in: Proceedings of the 13th international conference on Intelligent user interfaces, IUI '08, ACM, New York, NY, USA, 2008, pp. 199–206.

[16] G. Erkan, D.R. Radev, Lexrank: graph-based lexical centrality as salience in text summarization, J. Artif. Int. Res. 22 (2004) 457–479.

[17] E. Filatova, A formal model for information selection in multi-sentence text extraction, in: In Proceedings of the International Conference on Computational Linguistics (COLING), pp. 397–403.

[18] T. Hamrouni, S.B. Yahia, E.M. Nguifo, Looking for a structural characterization of the sparseness measure of (frequent closed) itemset contexts, Information Sciences 222 (2013) 343 – 361.

[19] S. Jaroszewicz, D.A. Simovici, Interestingness of frequent itemsets using bayesian networks as background knowledge, in: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 178–186.

[20] J.M. Kleinberg, Authoritative sources in a hyperlinked environment, J. ACM 46 (1999) 604–632.

[21] C.Y. Lin, E. Hovy, Automatic evaluation of summaries using n-gram co-occurrence statistics, in: Proceedings of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1, pp. 71–78.

[22] M. Litvak, M. Last, Graph-based keyword extraction for single-document summarization, in: Proceedings of the Workshop on Multi-source Multilingual Information Extraction and Summarization, MMIES '08, Association for Computational Linguistics, Stroudsburg, PA, USA, 2008, pp. 17–24.

[23] E. Loper, S. Bird, Nltk: The natural language toolkit, in: In Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics. Philadelphia: Association for Computational Linguistics, pp. 1–4.

[24] M. Mampaey, N. Tatti, J. Vreeken, Tell me what i need to know: succinctly summarizing data with itemsets, in: Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '11, ACM, New York, NY, USA, 2011, pp. 573–581.

[25] G. Mansingh, K.M. Osei-Bryson, H. Reichgelt, Using ontologies to facilitate post-processing of association rules by domain experts, Inf. Sci. 181 (2011) 419–434.

[26] J.M. Mendel, M.M. Korjani, Charles ragin's fuzzy set qualitative comparative analysis (fsqca) used for linguistic summarizations, Information Sciences 202 (2012) 1 – 23.

[27] J.G.V. Mittal, J. Goldstein, V. Mittal, J. Carbonell, M. Kantrowitz, Multi-document summarization by sentence extraction, in: In Proceedings of the ANLP/NAACL Workshop on Automatic Summarization, pp. 40–48.

[28] N. Pasquier, Y. Bastide, R. Taouil, L. Lakhal, Discovering frequent closed itemsets for association rules, in: Proceedings of the 7th International Conference on Database Theory, ICDT '99, Springer-Verlag, London, UK, UK, 1999, pp. 398–416.

[29] D.R. Radev, H. Jing, M. Stys, D. Tam, Centroid-based summarization of multiple documents, Information Processing and Management 40 (2004) 919–938.

[30] T. Ralphs, M. Guzelsoy, The SYMPHONY callable library for mixed integer programming, The Next Wave in Computing, Optimization, and Decision Technologies 29 (2006) 61–76. Software available at *http://http://www.coin-or.org/SYMPHONY*.

[31] R. Ribaldo, A.T. Akabane, L.H.M. Rino, T.A.S. Pardo, Graph-based methods for multi-document summarization: exploring relationship maps, complex networks and discourse information, in: Proceedings of the 10th international conference on Computational Processing of

the Portuguese Language, PROPOR'12, Springer-Verlag, Berlin, Heidelberg, 2012, pp. 260–271.

[32] J. Roberto, J. Bayardo, Efficiently mining long patterns from databases, in: L.M. Haas, A. Tiwary (Eds.), SIGMOD 1998, pp. 85–93.

[33] N. Rotem, Open text summarizer (ots). retrieved from http://libots.sourceforge.net/ in july 2011, 2011.

[34] D. Shahaf, C. Guestrin, Connecting two (or less) dots: Discovering structure in news articles, ACM Trans. Knowl. Discov. Data 5 (2012) 24:1–24:31.

[35] J. Steinberger, M. Kabadjov, R. Steinberger, H. Tanev, M. Turchi, V. Zavarella, Jrc's participation at tac 2011: Guided and multilingual summarization tasks, in: TAC'11: Proceedings of the The 2011 Text Analysis Conference, pp. 1–9.

[36] A. Tagarelli, Exploring dictionary-based semantic relatedness in labeled tree data, Information Sciences 220 (2013) 244 – 268.

[37] H. Takamura, M. Okumura, Text summarization model based on the budgeted median problem, in: Proceeding of the 18th ACM conference on Information and knowledge management, pp. 1589–1592.

[38] P.N. Tan, V. Kumar, J. Srivastava, Selecting the right interestingness measure for association patterns, in: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'02), pp. 32–41.

[39] TexLexAn, Texlexan: An open-source text summarizer. retrieved from http://texlexan.sourceforge.net/ in july 2011, 2011.

[40] Text Analysis Conference, NIST text analysis conference summarization track. http://www.nist.gov/tac/2011/summarization. last accessed: 20/03/2013, 2011.

[41] K. Thakkar, R. Dharaskar, M. Chandak, Graph-based algorithms for text summarization, in: Emerging Trends in Engineering and Technology (ICETET), 2010 3rd International Conference on, pp. 516 –519.

[42] D. Wang, T. Li, Document update summarization using incremental hierarchical clustering, in: Proceedings of the 19th ACM international conference on Information and knowledge management, pp. 279–288.

[43] D. Wang, S. Zhu, T. Li, Y. Chi, Y. Gong, Integrating document clustering and multidocument summarization, ACM Trans. Knowl. Discov. Data 5 (2011) 14:1–14:26.

[44] Wordnet, Wordnet Lexical Database. Available at http://wordnet.princeton.edu. Last access: 30/01/2013, 2013.

[45] Z. Yang, K. Cai, J. Tang, L. Zhang, Z. Su, J. Li, Social context summarization, in: Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval, SIGIR '11, ACM, New York, NY, USA, 2011, pp. 255–264.

[46] J. Zhu, C. Wang, X. He, J. Bu, C. Chen, S. Shang, M. Qu, G. Lu, Tag-oriented document summarization, in: Proceedings of the 18th in-

ternational conference on World wide web, WWW '09, ACM, New York, NY, USA, 2009, pp. 1195–1196.