

An approach to graph-based analysis of textual documents

Antoon Bronselaer¹ Gabriella Pasi²

¹Department of Telecommunications and Information Processing, Ghent University

²Department of Informatics, Systems and Communication, Università degli Studi di Milano Bicocca

Abstract

In this paper a new graph-based model is proposed for the representation of textual documents. Graph-structures are obtained from textual documents by making use of the well-known Part-Of-Speech (POS) tagging technique. More specifically, a simple rule-based (re)classifier is used to map each tag onto graph vertices and edges. As a result, a decomposition of textual documents is obtained where tokens are automatically parsed and attached to either a vertex or an edge. It is shown how textual documents can be aggregated through their graph-structures and finally, it is shown how vertex-ranking methods can be used to find relevant tokens.¹

Keywords: Text Analysis, Graph model, Multi Document Summarization

1. Introduction

In the past decades, the automatic analysis of texts has become an important and challenging topic in several research areas such as Information Retrieval (IR), Natural Language Processing (NLP) and Computational Linguistics (CL). Related to these fields of research, a variety of tasks has been addressed, including text clustering, text classification, topic extraction, text filtering and text indexing. At the basis of the solutions to these tasks lies a narrow range of commonly accepted representation models.

In Information Retrieval, text representation is usually based on keyword (or n -gram) association and weighting, thus producing the classical bag-of-words approach. Well-known weighting schemes are Tf*Idf [1] and BM25 [2]. The bag-of-words model has led to effective solutions to several problems in the past; however, it is commonly recognized that it has some limitations that deserve attention. An important issue with the bag-of-words model is that it omits the syntactic structure of text to a large extent. More specifically, the order of terms in sentences is ignored, boundaries between terms are quite arbitrary, and connections between terms are not represented. In addition, semantic aspects are not represented in the bag-of-words model.

The aim of this paper is to propose an alternative representation of textual documents to the aim of modeling (at some extent) the syntactic structure of texts. This alternative model originates from the idea that a text carries several concepts, and that these concepts are related to one another; to take into account these relations, in this paper a model of a text as a *graph* is proposed. Like the approach in [3], the purpose is to create a graph that is labeled with n -grams of variable length. However, whereas the approach in [3] uses a set of predefined grammatical patterns, this premise is not valid here; in fact, the proposed method discovers these patterns. The proposed approach is, to the authors' knowledge, the first approach that avoids the usage of an external knowledge base to produce a labeled graph.

It will be shown that each textual document can be transformed into a graph by means of a four-steps procedure. The semantics of several operations such as graph union will be analyzed. Attention will be paid to finding relevant subgraphs in order to identify and represent the main topics of one or more texts. Therefore, vertex rank methods will be mutually compared in terms of the similarity between the subgraphs they imply.

The remainder of this paper is structured as follows. In Section 3, some preliminary concepts involving graph theory and Part-Of-Speech (POS) tagging are explained. In Section 4, an algorithm is presented to transform textual documents into graphs; the (dis)advantages of our approach are also discussed. In Section 5, some graph operations are introduced, and it is shown how they can be used in text analysis. In Section 6, some experiments related to these graph operations are reported. In Section 7, future work is discussed and finally, in Section 8, the most important contributions of this paper are summarized.

2. Related work

To the aim of representing texts in a more semantic oriented way, several approaches have been recently proposed based on a network structure to represent a text. The most important of these contributions are reported in this section.

In [3] a method is proposed to infer *conceptual graphs* from textual documents. In this approach, each sentence is first analyzed in order to discover the different semantic roles it contains. The seman-

¹**Acknowledgment:** This work is supported by the Flemish Fund for Scientific Research (FWO-Vlaanderen).

tic role of a token within a sentence describes its function with respect to the central verb. The identification of semantic roles is based on semantic patterns available in VerbNet and WordNet. Based on the identified semantic roles and on a set of rules a graph is produced. In [4], the proximity of features (i.e. tokens) is modeled in a graph by labeling its vertices with features, and by drawing edges between the features that appear close to each other in a document. Closeness is here determined by a window of fixed size that is slid over the stream of tokens. The edges are then annotated with features that describe the strength of the relationship between related concepts. Several such features (e.g., co-occurrence frequency and transition probabilities) are proposed. A similar idea is exploited in [5], where such a model is applied for text similarity assessment. In [6], a graph model for documents is exploited in the field of document classification. Similar to the approach in [3], the grammatical structure of textual documents is taken into account when producing the graph. Finally, in [7], an approach is presented to construct a document-concept bipartite graph by using Wikipedia as an external knowledge resource.

3. Preliminaries

3.1. Graphs

A graph G is represented as a pair (V, E) where V denotes a set of vertices and E denotes a set of edges having that $E \subseteq V \times V$. In several applications of graph theory, edges are associated with features (e.g., weights, capacity, vertex distance...). To this aim a function ϕ is defined that maps edges onto a feature space F . Formally:

$$\phi : E \rightarrow F.$$

The function ϕ will be used to annotate edges with a character string that represents a specific relationship between two vertices in the graph. In Section 4.4, the construction of ϕ will be explained in detail. If we have for $G = (V, E)$ that E is a multiset rather than a set, G is called a *multigraph*. Informally, a multigraph is a graph where multiple edges between two vertices can occur. In the following, the more general case of multigraphs will be considered.

3.2. Part-Of-Speech tagging

At the basis of our approach for text analysis lies the mechanism of Part-Of-Speech tagging. The purpose of POS tagging is to assign the correct lexical category (e.g., noun, verb, article...), to each word in a text. The main difficulty with POS tagging is that the assignment of a word class is often an ambiguous task as the lexical category of a word usually depends on the context in which it is used. For example, the word “store” can be used

as a both noun or a verb. To deal with this ambiguity, POS taggers usually consider sequences of n words in order to derive the context in which words are used. The two main approaches for POS tagging are rule-based methods [8, 9] and probabilistic methods [10, 11, 12, 13].

4. Graph-based text representation

In this section, the proposed graph-based representation model for textual documents is described. In the following it is assumed that a set of documents is available; this set is also called the *corpus* and it is denoted as:

$$\mathcal{D} = \{d_1, \dots, d_n\}.$$

Each document is assumed to contain a finite sequence of characters; the approach for transforming a document $d \in \mathcal{D}$ into a graph consists of four simple steps:

- Tokenization
- Part-Of-Speech tagging
- Reclassification
- Graph generation

Basically, to generate a graph representation of a text, the text is first tokenized. Next, each of the tokens is assigned a Part-Of-Speech tag. The novelty of the approach presented in this paper lies in the *reclassification* of tags: in order to generate a graph structure, a simple rule-based algorithm is applied that observes tokens (paired with tags) and reclassifies each token in one of the following classes: Vertex, Edge, Ignore. The final phase of our method, i.e. the graph generation, generates a graph by only using the classes generated during the reclassification. Figure 1 schematically illustrates the proposed approach.

4.1. Tokenization

The first step of the approach proposed in this paper is simple tokenization. Within the scope of this paper, tokenization uses white spaces and punctuation signs as delimiters to split of tokens. It transforms a text into a list of tokens, which can be processed by the POS-tagger. In the following, the set of tokens will be denoted as \mathcal{T} .

4.2. Part-Of-Speech tagging

In the second step, the obtained stream of tokens is annotated by the Part-Of-Speech tagger. This implies that each token is associated with a tag that indicates the grammatical function of the token within a sentence (e.g. verb, noun, article...). To this aim we use the probabilistic TreeTagger [13] because of the many languages that it supports with high effectiveness. The TreeTagger provides for each token both the POS-tag and the stem of the token.

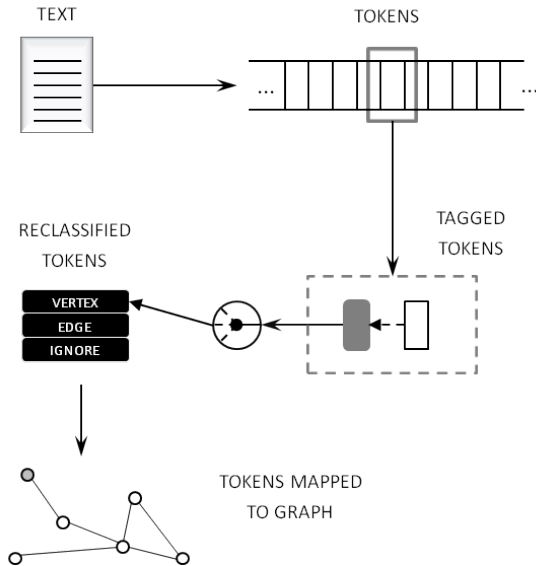


Figure 1: The graph conversion process

4.3. Reclassification

In the third step, the identified POS-tags are reclassified. The purpose of reclassification is to simplify the grammatical knowledge about the text, so that it can be easily mapped onto a graph structure. The general idea behind the approach proposed in this paper is that each sentence consists of concepts that are mutually linked by some relationship. In addition, some tokens are irrelevant to the content of the text, and as such they can be excluded from the representation. In this paper, we propose a reclassifier for the English language that downscales tags of the Penn treebank tagset [14]. Therefore, a class set $\mathcal{C} = \{Vertex, Edge, Ignore\}$ is considered. Each of these classes is explained in more detail below.

The first class that is used by the reclassifier is “Ignore”, which is used to mark tokens that should be excluded from the graph because of their irrelevance. The tokens that are tagged as determiner (e.g., “the”), pronoun (e.g., “me”) and adverb (e.g., “extremely”) are mapped to this class. The reason for removing irrelevant tokens this way instead of using a stopwords filter is two folded. First, removing stopwords before POS-tagging would significantly decrease the performance of the POS-tagger because removing stopwords implies removing context from the sentences. Secondly, the language model underlying a POS-tagger provides a more complete view on irrelevant tokens than does an exhaustive list of stopwords.

The second class is “Edge” and it is used to label tokens that identify a relationship within a sentence. In this paper we consider three types of relationships:

- Conjunction
- Preposition
- Verb

A conjunction is (as the term indicates) a connection between (sequences of) words in a sentence (e.g. John **and** Paul) that are therefore clearly related. Prepositions are alternative parts of speech to indicate relationships (often spatial in nature, but not exclusively) between words (e.g. John is **in** Milan). Finally, verbs indicate actions undertaken by a subject, and they are also clearly relational in nature (e.g. John **is** young). Tags belonging to any of the above three classes are reclassified as “Edge”. The graph generation algorithm associates then the corresponding tokens with edges in the graph.

The third class generated by the reclassifier is “Vertex”. This class is used to identify conceptual information and the corresponding tokens are attached to vertices in the graph. By default (i.e. if a token/tag pair does not match any of the classification rules) a token is assumed to be of the class “Vertex”. This assumption avoids the consideration of too numerous semantic patterns for the discovery of conceptual information as proposed in [3].

It is worth to notice that the rule set used within the scope of this paper has an extremely simple and intuitive structure. Although only designed for the English language, it is emphasized that an equivalent rule set can be easily constructed for other languages. Given the fact that (a) the TreeTagger has a rich variety of trained parameter models and (b) the proposed approach does not rely on external knowledge such as WordNet, VerbNet or YAGO, it can be seen that the representation model can be easily applied on many different languages.

4.4. Graph generation

Based on the reclassified set of tokens, an algorithm has been defined that generates a graph containing concepts as vertices and relationships between those concepts; the algorithm’s outcome is a graph $G = (\mathcal{T}, E)$. As such, each vertex is equal to a token from \mathcal{T} . Each edge $e \in E$ indicates a relationship between two vertices. In addition, a function ϕ with feature space $F = \mathcal{T}$ is used. Formally:

$$\phi : E \rightarrow \mathcal{T}.$$

Thus, each edge $e \in E$ is mapped onto a token that represents a specific relationship observed in the text. Algorithm 1 presents the graph generation in pseudo-code. Basically, Algorithm 1 works in two steps. In the first step, the list of token-class pairs is reduced by merging adjacent pairs of the same class together into one pair with concatenated tokens. At the same time, tokens of the class “Ignore” are deleted. In the second step, the reduced list is examined for Vertex-Edge-Vertex triplets. For each such triplet, two vertices and an edge are added to the graph. Both the vertices and the edge are labeled with the tokens that have been observed.

Algorithm 1 Graph generation pseudo-code

Require: $L \subseteq (\mathcal{T} \times \mathcal{C})^k$ **Ensure:** $G = (\mathcal{T}, E)$

```
1: for all  $(t_i, c_i) \in L$  do
2:   if  $c_i = \text{Ignore}$  then
3:     delete( $L[i]$ )
4:   else
5:     while  $c_i = c_{i+1}$  do
6:        $t_i := t_i \oplus t_{i+1}$ 
7:       delete( $L[i+1]$ )
8:     end while
9:   end if
10: end for
11: for all  $(t_i, c_i) \in L$  do
12:   if  $c_i = c_{i+2} = \text{Vertex}$  and  $c_{i+1} = \text{Edge}$  then
13:     addVertex( $G, t_i$ )
14:     addVertex( $G, t_{i+2}$ )
15:     addEdge( $G, t_i, t_{i+2}, t_{i+1}$ )
16:   end if
17: end for
```

Example 1

In order to illustrate the transformation of a text into a graph, we apply our method to the following example text introduced in [15]:

“John and Bill wanted money. They bought ski-masks and guns and stole an old car from a neighbor. Wearing their ski-masks and waving their guns, the two entered the bank, and within minutes left the bank with several bags of \$100 bills. They drove away happy, throwing away the ski-masks and guns in a sidewalk trash can. They were never caught.”

The application of our method produces the graph shown in Figure 2. Hereby, the vertices are tokens as prescribed by Algorithm 1. Similarly, the edges between vertices are drawn in the graph and they are marked with their label that defines the relationship between the vertices.

A clear advantage of the graph generation method is that it labels both vertices and edges with tokens that consist of a variable amount of words. These tokens are identified because they consist of parts of speech that are mapped to the same class in \mathcal{C} . As such, the advantage of the proposed method is that relevant combinations of words are discovered without the application of any probabilistic co-occurrence model (e.g. [4]). Another advantage of this graph-generation model is that it can be applied to any language for which a POS tagging model is available. An important observation with respect to our method is that the computational complexity of the method strongly depends on the complexity of the POS-tagger. As this indeed can be a problem, in the future we will investigate how the usage of the POS-tagger can be reduced, by making for example a selection of sentences to be tagged, rather than tagging the entire text. However, experiments with respect to the computational complexity of the method are outside the scope of this paper.



Figure 2: Graph-model for robbery story

5. Graph operations

In this section, two well-known operations on graphs are revised: *graph union* and *vertex ranking*. It will be explained how these operations allow to exploit the graph model as a solution to some known text analysis problems.

5.1. Graph Union

The union of two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ is denoted as $G_1 \cup G_2$ and equals $(V_1 \cup V_2, E_1 \cup E_2)$. Graph union can be seen as a way of merging two graphs without any loss of information (i.e., without excluding any vertices or edges). It is therefore a useful operator to merge information from multiple textual documents. Indeed, if we consider the corpus $\mathcal{D} = \{d_1, \dots, d_n\}$ and if G_i denotes that graph-model corresponding to d_i , then the combined information in the corpus is represented by:

$$G_{\mathcal{D}} = \bigcup_{i=1}^n G_i.$$

Needless to say, this representation quickly leads to an overwhelming amount of information (i.e., the number of vertices grows quickly). Therefore, the usage of graph union to merge textual documents typically requires post-processing with an operator that extracts *relevant* information from the combined graph. One method to do this is by ranking vertices according to their relevance. Such ranking techniques will be discussed in the next section.

5.2. Vertex Ranking

Let us assume now a corpus \mathcal{D} of documents centered on a same topical domain, and let us suppose we wish to model the topic(s) covered by these documents. As previously discussed, it is possible to represent all documents in \mathcal{D} in one graph by applying the graph union operator to each single document graph. Let us denote that graph $G_{\mathcal{D}}$. In order to model the topic(s) covered by the documents in \mathcal{D} , a subgraph $G_{(k)}^{\bullet} \subseteq G$ must be constructed that contains only the top- k most relevant vertices from $G_{\mathcal{D}}$. In order to find such a subgraph $G_{(k)}^{\bullet}$, a vertex rank [16] method will be applied to find the k most important vertices. Next, the graph $G_{\mathcal{D}}$ will be projected by only retaining these k vertices, thus producing a subgraph $G_{(k)}^{\bullet}$. Note that this projection of $G_{\mathcal{D}}$ can lead to a subgraph $G_{(k)}^{\bullet}$ that consists of several (mutually non-connected) components.

Definition 1 (Vertex rank)

Given a graph $G = (V, E)$, an \mathbb{L} -valued vertex rank method is defined by a function:

$$r : V \rightarrow \mathbb{L}$$

where (\mathbb{L}, \leq) is a totally ordered lattice. The function r implies an order relation on V , denoted \leq_r that satisfies (using infix notation):

$$\forall (v_1, v_2) \in V^2 : (v_1 \leq_r v_2) \Leftrightarrow (r(v_1) \leq (r v_2)).$$

In practical settings, \mathbb{L} is instantiated with \mathbb{R} , \mathbb{N} or $[0, 1]$. The simplest example of vertex ranking is based on the *degree* of the vertex, which is based on counting for each vertex $v \in V$, the number of neighbors of v . More formally:

$$\text{deg} : V \rightarrow \mathbb{N}$$

where:

$$\text{deg}(v) = \left| \{v' | v' \in V \wedge \exists (v, v') \in E\} \right|.$$

The function deg assigns more importance to vertices that have many neighbors. This means that a vertex is important if its token is related to many other tokens. A second and more advanced vertex rank method is called *closeness* and is defined by:

$$\text{cls} : V \rightarrow \mathbb{R}$$

where:

$$\text{cls}(v) = \sum_{v' \neq v \in V} \frac{1}{d_{sp}(v, v')}$$

and where $d_{sp}(v, v')$ is the shortest path distance between v and v' . The function cls assigns more importance to vertices that are close to all other vertices in the graph. Given a vertex rank method r , it is possible to select the k highest ranked vertices and retain only the part of $G_{\mathcal{D}}$ that contains these vertices. This leads to the following definition.

Definition 2 (Top- k projection)

Given a graph $G = (V, E)$, a vertex rank method r and a natural number $k \leq |V|$, the top- k projection of G is a subgraph $G_{(k)}^{\bullet} = (V', E')$, such that V' contains the k highest ranked vertices from V and E' contains all edges from E between vertices in V' .

Informally, the top- k projection of a graph maintains only those k vertices in the graph that are ranked highest according to a vertex rank method r . In addition, only edges between two vertices from the top- k are preserved. An example of top- k projection with $k = 5$ and $r = \text{deg}$ is shown in Figure 3 where the top-5 ranked vertices are colored in dark grey.

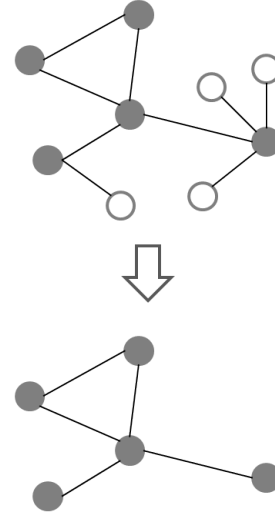


Figure 3: Graph G and its top-5 projection with $r = \text{deg}$

6. Experiments

In this section, two experiments involving the proposed graph-model are reported. In the first experiment the impact of the graph composition (weighted vs. non-weighted edges, directed vs. non-directed edges) is investigated. In the second experiment our graph-model is evaluated in a setting of Multi-Document Summarization (MDS). Both experiments are conducted on a dataset provided by the Document Understanding Conference (DUC)². More specifically, we use the DUC-2002 dataset. The dataset consists of 59 clusters of documents, where documents in the same cluster are describing a common event or person. On average, each cluster contains about ten documents. In the experiments reported below, we consider each cluster separately as a corpus $\mathcal{D} = \{d_1, \dots, d_n\}$ and we calculate the graph $G_{\mathcal{D}}$ as the union of the graphs of all documents (see Section 5.1). Next, we will derive a subgraph $G_{(k)}^{\bullet}$ from $G_{\mathcal{D}}$ with an appropriate choice of k .

²See <http://duc.nist.gov>

6.1. Vertex Rank Similarity

In the first experiment, the impact of the graph composition of $G_{\mathcal{D}}$ on $G_{(k)}^{\bullet}$ is studied. More specifically, it is investigated how the design choices to generate the graphs G_i (and as a consequence also the graph $G_{\mathcal{D}}$) influence the resulting subgraph $G_{(k)}^{\bullet}$. We therefore consider the following two alternative design choices:

- **Edge weighting** The first design choice is whether or not to weigh edges. If edges are weighted, the distance between two adjacent vertices becomes inversely proportional to the number of edges between those vertices. In Figure 2, the distance between vertices “ski-masks” and “guns” equals $1/2$ in the weighted case due to the two edges between those vertices. In the unweighted case, the distance between two adjacent vertices is always 1. By applying weights to the edges, we model that two vertices have lower distance if there exist more relationships between them. This will have an increasing effect on the closeness of these vertices. The application of weights thus has the effect that rank method cls will prefer pairs of vertices with many mutual relationships. For simplicity, we model the effect of multiple relationships in an inverse linear manner.
- **Edge direction** The second design choice is whether or not to consider directed edges. In Figure 2, the undirected case is shown, thereby losing the information of the order in which the vertices occur in the text.

For a given vertex rank method r , these design choices lead to four different variations. In the following, we shall denote the design choices by means of a two-bit mask. More specifically, we denote a vertex rank method as $r_{b_1 b_2}$ where b_1 is a bit that indicates whether edges are weighted ($b_1 = 1$) or not ($b_1 = 0$) and b_2 is a bit that indicates whether edges are directed ($b_2 = 1$) or not ($b_2 = 0$). For example, cls_{01} denotes the closeness rank method used on a graph with non-weighted, directed edges.

The main question that we try to answer in this section is to what extent the design choices result in a different subgraph $G_{(k)}^{\bullet}$. Therefore, for each cluster in the dataset, the similarity will be measured between two subgraphs that stem from different design choices. Hereby, similarity between two graphs is calculated as the Jaccard index of the vertex sets. More specifically, for two graphs G_1 and G_2 , we denote the similarity of these two graphs as:

$$\text{sim}(G_1, G_2) = J(V_1, V_2) = \frac{|V_1 \cap V_2|}{|V_1 \cup V_2|}.$$

Next, we can calculate the mean similarity over all clusters. Figure 4 shows the similarities between cls and deg for the different choices of design as a heatmap, where a dark color indicates high similarity.

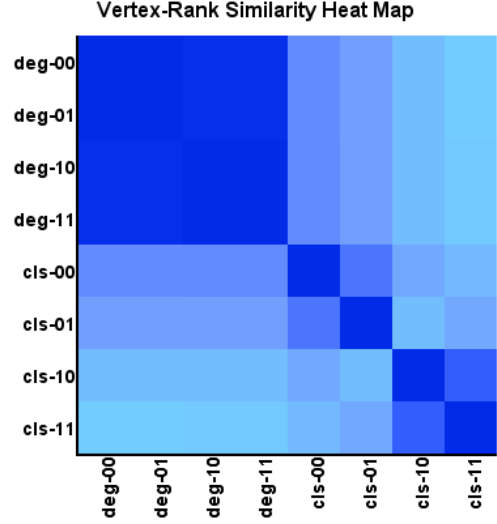


Figure 4: Similarity heatmap for $\text{cls}_{b_1 b_2}$ and $\text{deg}_{b_1 b_2}$ for $k = 10$

An interesting observation is the high mutual similarity between subgraphs obtained by vertex rank method deg. This is however not surprising, as it is known that the degree of a vertex is not influenced by weighting the edges. We see however that the subgraphs obtained by using deg and cls show lower mutual similarity. Also, for vertex rank method cls, the different design choices lead to different subgraphs. This indicates that it makes sense to consider different vertex rank methods and in case of cls, it makes sense to consider different design choices. Because of the minimal impact of the design choices for deg, we shall consider only deg_{00} in the remainder of this paper.

6.2. Multi-Document Summarization

In the second experiment, it is investigated to what extent the graph decomposition method can be used in a Multi-Document Summarization task. More specifically, it is studied to what extent the labels of the vertices of a subgraph $G_{(k)}^{\bullet}$ can serve as the basis for generating a text summary. As such, it is evaluated to what extent a subgraph $G_{(k)}^{\bullet}$ can serve as a content selection tool. As ground truth, the handwritten summaries (multiple summaries per cluster) available in the DUC-2002 dataset are used. For our experiments, we use the 50-word length summaries.

In order to evaluate the accuracy of the content provided by a subgraph $G_{(k)}^{\bullet}$, the ROUGE- N evaluation metric [17] is used. This is a recall-based method that is generally accepted as a good accuracy metric for Multi-Document Summarization. The ROUGE- N metric compares a generated summary with a set of reference summaries by calculating the ratio of N -grams in the reference summaries that also occur in the generated summary. Because our method produces a graph rather than a

summary (i.e., a text), the vertices of the produced subgraph $G_{(k)}^\bullet$ are tokenized and collected in a list. Next, as prescribed in [18], we use Porter stemming to preprocess the tokens and calculate the ROUGE-1 scores for the different vertex rank methods. Because the ROUGE-1 is biased to content with a high amount of tokens, the parameter k (i.e., the number of vertices in $G_{(k)}^\bullet$) is ranged from 5 to 50 in steps of five in order to obtain a correct evaluation of our approach. In addition, the ILP method described in [18] is used as a baseline method³.

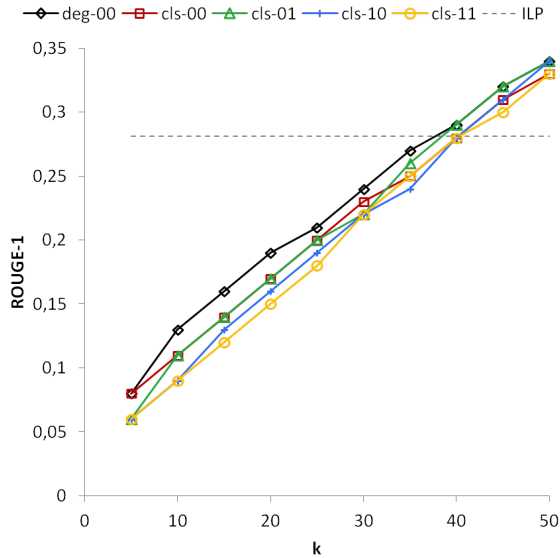


Figure 5: Average ROUGE-1 scores for five vertex rank methods and baseline ILP (dotted line)

Figure 5 shows the ROUGE-1 scores for the baseline method and the five vertex rank methods under consideration. The first observation is that vertex ranker deg_{00} is consistently better than cls in all its variants. The second observation is that the ROUGE-1 increases linearly with an increasing amount of vertices in the subgraph. It can be seen that for $k = 40$ (i.e., 40 vertices in the subgraph), the ROUGE-1 score of all vertex rank methods is comparable to that of the baseline ILP method. This effect is due to an increasing number of vertices in the created subgraph. Indeed, taking into account that $k_1 \leq k_2 \Rightarrow G_{(k_1)}^\bullet \subseteq G_{(k_2)}^\bullet$, we have that adding more vertices to the subgraph increases the probability that the subgraph will contain tokens that appear also in the reference summaries. In other words, the bigger the subgraph, the higher its recall with respect to reference summaries.

7. Future work

Within the scope of this paper, we have studied a novel, graph-based approach for the representation of a text. Although the usability of this model was

³It was verified that the experiment setup is equal to the one in [18]

briefly illustrated in the context of Multi-Document Summarization, it is our goal to further investigate this model and its applications. Some ideas to guide this future research are the following. First, there exist many other vertex rank methods besides degree and closeness. Investigating these rank methods will be a useful contribution. Also other graph reduction methods such as the network Pathfinder algorithm [19] can be taken into account. Second, other applications of the model, such as sentiment analysis, will be explored. Third, as the proposed model relies on POS-tagging, which is inherently language dependent, the model should be evaluated on other languages.

8. Conclusions

In this paper we have introduced a novel graph-model for representing a text. The principle underlying our approach is that tokens associated with a text can be mapped onto vertices and edges of a graph by considering the syntactic function of the token. Therefore our approach uses Part-Of-Speech tagging to annotate each token in a text with its correct lexical category. Next, a text is processed as a stream of tokens, and each token is attributed to either a vertex or an edge in a dynamically generated graph structure. The main advantage of this approach is that vertices and edges are labeled with N -grams where N is not a constant value within the same graph. Finally, it was shown how our graph-model can be used in the context of Multi-Document Summarization by applying the operations of graph union and vertex ranking. Several vertex ranking methods were evaluated by means of the well-known ROUGE- N score and compared to a baseline technique from the literature. The evaluations have shown that, in terms of the ROUGE-1 score, our approach is competitive with the baseline technique.

References

- [1] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern information retrieval*. ACM Press, 1999.
- [2] Karen Spärck Jones, Steve Walker, and Stephen Robertson. A probabilistic model of information retrieval: Development and comparative experiments (parts 1 and 2). *Information Processing and Management*, 36(6):779–840, 2000.
- [3] Svetlana Hensman. *Automatic Construction of Conceptual Graphs from Texts using Computational Linguistics Techniques*. PhD thesis, School of Computer Science and Informatics, University College Dublin, 2004.
- [4] Wei Jin and Rohini Srihari. Graph-based text representation and knowledge discovery. In

- Proceedings of the SAC conference*, pages 807–811, Seoul, Korea, 2007.
- [5] Faguo Zhou, Fan Zhang, and Bingru Yang. Graph-based text representation model and its realization. In *Natural Language Processing and Knowledge Engineering (NLP-KE), 2010 International Conference on*, pages 1–8, 2010.
 - [6] Chuntao Jiang, Frans Coenen, Robert Sanderson, and Michele Zito. Text classification using graph mining-based feature extraction. *Knowledge-Based Systems*, 23(4):302 – 308, 2010.
 - [7] Lu Zhang, Chunping Li, Jun Liu, and Hui Wang. Graph-based text similarity measurement by exploiting wikipedia as background knowledge. *World Academy of Science, Engineering and Technology*, pages 1548 – 1553, 2011.
 - [8] Eric Brill. *A Corpus-Based Approach to Language Learning*. PhD thesis, Department of Computer and Information Science, University of Pennsylvania, 1993.
 - [9] Barbara Greene and Gerald Rubin. *Automatic Grammatical Tagging of English*. Department of Linguistics, Brown University, 1971.
 - [10] Lalit Bahl and Robert Mercer. Part-of-speech assignment by a statistical decision algorithm. In *IEEE International Symposium on Information Theory*, pages 88–89, 1976.
 - [11] Kenneth Church. A stochastic parts program and noun phrase parser for unrestricted text. In *Proceedings of the Third Conference on Applied Natural Language Processing*, pages 136–143, 1988.
 - [12] Steven DeRose. Grammatical category disambiguation by statistical optimization. *Computational Linguistics*, 14(1):31–39, 1988.
 - [13] Helmut Schmid. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of International Conference on New Methods in Language Processing*, pages 44–49, 1994.
 - [14] Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313 – 330, 1993.
 - [15] Chin-Yew lin. *Robust Automated Topic Identification*. PhD thesis, Faculty of the Graduate School, University of Southern California, 1997.
 - [16] Louis Hakimi. On realizability of a set of integers as degrees of the vertices of a linear graph. *Journal of the Society for Industrial and Applied Mathematics*, 10:496–506, 1962.
 - [17] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Proceedings of the Workshop on Text Summarization Branches Out*, pages 74–81, 2004.
 - [18] Ryan MacDonald. A study of global inference algorithms in multi-document summarization. In *Proceedings of the 29th European conference on IR research*, pages 557–564, 2007.
 - [19] Roger Schvaneveldt, Donald Dearholt, and Francis Durso. Graph theoretic foundations of pathfinder networks. *Computers and Mathematics with Applications*, 15(4):337–345, 1988.