TextRank: Bringing Order into Texts

Rada Mihalcea and Paul Tarau
Department of Computer Science
University of North Texas
{rada,tarau}@cs.unt.edu

Abstract

In this paper, we introduce TextRank – a graph-based ranking model for text processing, and show how this model can be successfully used in natural language applications. In particular, we propose two innovative unsupervised methods for keyword and sentence extraction, and show that the results obtained compare favorably with previously published results on established benchmarks.

1 Introduction

Graph-based ranking algorithms like Kleinberg's HITS algorithm (Kleinberg, 1999) or Google's PageRank (Brin and Page, 1998) have been successfully used in citation analysis, social networks, and the analysis of the link-structure of the World Wide Web. Arguably, these algorithms can be singled out as key elements of the paradigm-shift triggered in the field of Web search technology, by providing a Web page ranking mechanism that relies on the collective knowledge of Web architects rather than individual content analysis of Web pages. In short, a graph-based ranking algorithm is a way of deciding on the importance of a vertex within a graph, by taking into account global information recursively computed from the entire graph, rather than relying only on local vertex-specific information.

Applying a similar line of thinking to lexical or semantic graphs extracted from natural language documents, results in a graph-based ranking model that can be applied to a variety of natural language processing applications, where knowledge drawn from an entire text is used in making local ranking/selection decisions. Such text-oriented ranking methods can be applied to tasks ranging from automated extraction of keyphrases, to extractive summarization and word sense disambiguation (Mihalcea et al., 2004).

In this paper, we introduce the TextRank graph-based ranking model for graphs extracted from natural language texts. We investigate and evaluate the application of TextRank to two language processing tasks consisting of unsupervised keyword and sentence extraction, and show that the results obtained with TextRank are competitive with state-of-the-art systems developed in these areas.

2 The TextRank Model

Graph-based ranking algorithms are essentially a way of deciding the importance of a vertex within a graph, based on global information recursively drawn from the entire graph. The basic idea implemented by a graph-based ranking model is that of "voting" or "recommendation". When one vertex links to another one, it is basically casting a vote for that other vertex. The higher the number of votes that are cast for a vertex, the higher the importance of the vertex. Moreover, the importance of the vertex casting the vote determines how important the vote itself is, and this information is also taken into account by the ranking model. Hence, the score associated with a vertex is determined based on the votes that are cast for it, and the score of the vertices casting these votes.

Formally, let $G = (V, E)$ be a directed graph with the set of vertices V and set of edges E, where E is a subset of $V \times V$. For a given vertex V_i, let $In(V_i)$ be the set of vertices that point to it (predecessors), and let $Out(V_i)$ be the set of vertices that vertex V_i points to (successors). The score of a vertex V_i is defined as follows (Brin and Page, 1998):

$$S(V_i) = (1 - d) + d * \sum_{j \in In(V_i)} \frac{1}{|Out(V_j)|} S(V_j)$$

where d is a damping factor that can be set between 0 and 1, which has the role of integrating into the model the probability of jumping from a given vertex to another random vertex in the graph. In the context of Web surfing, this graph-based ranking algorithm implements the "random surfer model", where a user clicks on links at random with a probability d, and jumps to a completely new page with probability $1 - d$. The factor d is usually set to 0.85 (Brin and Page, 1998), and this is the value we are also using in our implementation.

Starting from arbitrary values assigned to each node in the graph, the computation iterates until convergence below a given threshold is achieved [1]. After running the algorithm, a score is associated with each vertex, which represents the "importance" of the vertex within the graph. Notice that the final values obtained after TextRank runs to completion are not affected by the choice of the initial value, only the number of iterations to convergence may be different.

It is important to notice that although the TextRank applications described in this paper rely on an algorithm derived from Google's PageRank (Brin and Page, 1998), other graph-based ranking algorithms such as e.g. HITS (Kleinberg, 1999) or Positional Function (Herings et al., 2001) can be easily integrated into the TextRank model (Mihalcea, 2004).

2.1 Undirected Graphs

Although traditionally applied on directed graphs, a recursive graph-based ranking algorithm can be also applied to undirected graphs, in which case the out-degree of a vertex is equal to the in-degree of the vertex. For loosely connected graphs, with the number of edges proportional with the number of vertices, undirected graphs tend to have more gradual convergence curves.

Figure 1 plots the convergence curves for a randomly generated graph with 250 vertices and 250 edges, for a convergence threshold of 0.0001. As the connectivity of the graph increases (i.e. larger number of edges), convergence is usually achieved after fewer iterations, and the convergence curves for directed and undirected graphs practically overlap.

2.2 Weighted Graphs

In the context of Web surfing, it is unusual for a page to include multiple or partial links to another page, and hence the original PageRank definition for graph-based ranking is assuming unweighted graphs.

However, in our model the graphs are build from natural language texts, and may include multiple or partial links between the units (vertices) that are extracted from text. It may be therefore useful to indicate and incorporate into the model the "strength" of the connection between two vertices V_i and V_j as a weight w_{ij} added to the corresponding edge that connects the two vertices.

[1]Convergence is achieved when the error rate for any vertex in the graph falls below a given threshold. The error rate of a vertex V_i is defined as the difference between the "real" score of the vertex $S(V_i)$ and the score computed at iteration k, $S^k(V_i)$. Since the real score is not known apriori, this error rate is approximated with the difference between the scores computed at two successive iterations: $S^{k+1}(V_i) - S^k(V_i)$.

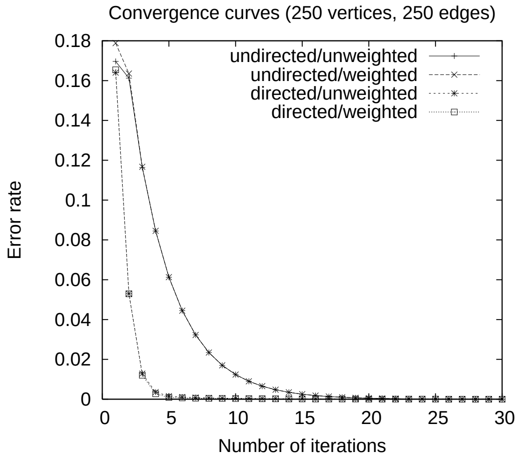


Figure 1: Convergence curves for graph-based ranking: directed/undirected, weighted/unweighted graph, 250 vertices, 250 edges.

Consequently, we introduce a new formula for graph-based ranking that takes into account edge weights when computing the score associated with a vertex in the graph. Notice that a similar formula can be defined to integrate vertex weights.

$$WS(V_i) = (1-d) + d * \sum_{V_j \in In(V_i)} \frac{w_{ji}}{\sum_{V_k \in Out(V_j)} w_{jk}} WS(V_j)$$

Figure 1 plots the convergence curves for the same sample graph from section 2.1, with random weights in the interval 0–10 added to the edges. While the final vertex scores (and therefore rankings) differ significantly as compared to their unweighted alternatives, the number of iterations to convergence and the shape of the convergence curves is almost identical for weighted and unweighted graphs.

2.3 Text as a Graph

To enable the application of graph-based ranking algorithms to natural language texts, we have to build a graph that represents the text, and interconnects words or other text entities with meaningful relations. Depending on the application at hand, text units of various sizes and characteristics can be added as vertices in the graph, e.g. words, collocations, entire sentences, or others. Similarly, it is the application that dictates the type of relations that are used to draw connections between any two such vertices, e.g. lexical or semantic relations, contextual overlap, etc.

Regardless of the type and characteristics of the elements added to the graph, the application of graph-based ranking algorithms to natural language texts consists of the following main steps:

1. *Identify text units that best define the task at hand, and add them as vertices in the graph.*
2. *Identify relations that connect such text units, and use these relations to draw edges between vertices in the graph. Edges can be directed or undirected, weighted or unweighted.*
3. *Iterate the graph-based ranking algorithm until convergence.*
4. *Sort vertices based on their final score. Use the values attached to each vertex for ranking/selection decisions.*

In the following, we investigate and evaluate the application of TextRank to two natural language processing tasks involving ranking of text units: (1) A keyword extraction task, consisting of the selection of keyphrases representative for a given text; and (2) A sentence extraction task, consisting of the identification of the most "important" sentences in a text, which can be used to build extractive summaries.

3 Keyword Extraction

The task of a keyword extraction application is to automatically identify in a text a set of terms that best describe the document. Such keywords may constitute useful entries for building an automatic index for a document collection, can be used to classify a text, or may serve as a concise summary for a given document. Moreover, a system for automatic identification of important terms in a text can be used for the problem of terminology extraction, and construction of domain-specific dictionaries.

The simplest possible approach is perhaps to use a frequency criterion to select the "important" keywords in a document. However, this method was generally found to lead to poor results, and consequently other methods were explored. The state-of-the-art in this area is currently represented by supervised learning methods, where a system is trained to recognize keywords in a text, based on lexical and syntactic features. This approach was first suggested in (Turney, 1999), where parametrized heuristic rules are combined with a genetic algorithm into a system for keyphrase extraction - GenEx - that automatically identifies keywords in a document. A different learning algorithm was used in (Frank et al., 1999), where a Naive Bayes learning scheme is applied on the document collection, with improved results observed on the same data set as used in (Turney, 1999). Neither Turney nor Frank report on the recall of their systems, but only on precision: a 29.0% precision is achieved with GenEx (Turney, 1999) for five keyphrases extracted per document, and 18.3% precision achieved with Kea (Frank et al., 1999) for fifteen keyphrases per document.

More recently, (Hulth, 2003) applies a supervised learning system to keyword extraction from abstracts, using a combination of lexical and syntactic features, proved to improve significantly over previously published results. As Hulth suggests, keyword extraction from abstracts is more widely applicable than from full texts, since many documents on the Internet are not available as full-texts, but only as abstracts. In her work, Hulth experiments with the approach proposed in (Turney, 1999), and a new approach that integrates part of speech information into the learning process, and shows that the accuracy of the system is almost doubled by adding linguistic knowledge to the term representation.

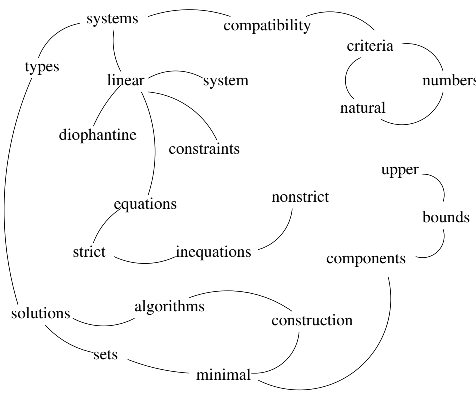
In this section, we report on our experiments in keyword extraction using TextRank, and show that the graph-based ranking model outperforms the best published results in this problem. Similar to (Hulth, 2003), we are evaluating our algorithm on keyword extraction from abstracts, mainly for the purpose of allowing for a direct comparison with the results she reports with her keyphrase extraction system. Notice that the size of the text is not a limitation imposed by our system, and similar results are expected with TextRank applied on full-texts.

3.1 TextRank for Keyword Extraction

The expected end result for this application is a set of words or phrases that are representative for a given natural language text. The units to be ranked are therefore sequences of one or more lexical units extracted from text, and these represent the vertices that are added to the text graph. Any relation that can be defined between two lexical units is a potentially useful connection (edge) that can be added between two such vertices. We are using a *co-occurrence* relation, controlled by the distance between word occurrences: two vertices are connected if their corresponding lexical units co-occur within a window of maximum N words, where N can be set anywhere from 2 to 10 words. Co-occurrence links express relations between syntactic elements, and similar to the semantic links found useful for the task of word sense disambiguation (Mihalcea et al., 2004), they represent cohesion indicators for a given text.

The vertices added to the graph can be restricted with syntactic filters, which select only lexical units of a certain part of speech. One can for instance consider only nouns and verbs for addition to the graph, and consequently draw potential edges based only on relations that can be established between nouns and verbs. We experimented with various syntactic filters, including: all open class words, nouns and verbs only, etc., with best results observed for nouns and adjectives only, as detailed in section 3.2.

The TextRank keyword extraction algorithm is fully unsupervised, and proceeds as follows. First, Compatibility of systems of linear constraints over the set of natural numbers. Criteria of compatibility of a system of linear Diophantine equations, strict inequations, and nonstrict inequations are considered. Upper bounds for components of a minimal set of solutions and algorithms of construction of minimal generating sets of solutions for all types of systems are given. These criteria and the corresponding algorithms for constructing a minimal supporting set of solutions can be used in solving all the considered types systems and systems of mixed types.



Keywords assigned by TextRank:
linear constraints; linear diophantine equations; natural numbers; nonstrict inequations; strict inequations; upper bounds

Keywords assigned by human annotators:
linear constraints; linear diophantine equations; minimal generating sets; non–strict inequations; set of natural numbers; strict inequations; upper bounds

Figure 2: Sample graph build for keyphrase extraction from an *Inspec* abstract

the text is tokenized, and annotated with part of speech tags – a preprocessing step required to enable the application of syntactic filters. To avoid excessive growth of the graph size by adding all possible combinations of sequences consisting of more than one lexical unit (ngrams), we consider only single words as candidates for addition to the graph, with multi-word keywords being eventually reconstructed in the post-processing phase.

Next, all lexical units that pass the syntactic filter are added to the graph, and an edge is added between those lexical units that co-occur within a window of N words. After the graph is constructed (undirected unweighted graph), the score associated with each vertex is set to an initial value of 1, and the ranking algorithm described in section 2 is run on the graph for several iterations until it converges – usually for 20-30 iterations, at a threshold of 0.0001.

Once a final score is obtained for each vertex in the graph, vertices are sorted in reversed order of their score, and the top T vertices in the ranking are retained for post-processing. While T may be set to any fixed value, usually ranging from 5 to 20 keywords (e.g. (Turney, 1999) limits the number of keywords extracted with his GenEx system to five), we are using a more flexible approach, which decides the number of keywords based on the size of the text. For the data used in our experiments, which consists of relatively short abstracts, T is set to a third of the number of vertices in the graph.

During post-processing, all lexical units selected as potential keywords by the TextRank algorithm are marked in the text, and sequences of adjacent keywords are collapsed into a multi-word keyword. For instance, in the text *Matlab code for plotting ambiguity functions*, if both *Matlab* and *code* are selected as potential keywords by TextRank, since they are adjacent, they are collapsed into one single keyword *Matlab code*.

Figure 2 shows a sample graph built for an abstract from our test collection. While the size of the abstracts ranges from 50 to 350 words, with an average size of 120 words, we have deliberately selected a very small abstract for the purpose of illustration. For this example, the lexical units found to have higher "importance" by the TextRank algorithm are (with the TextRank score indicated in parenthesis): numbers (1.46), inequations (1.45), linear (1.29), diophantine (1.28), upper (0.99), bounds (0.99), strict (0.77). Notice that this ranking is different than the one rendered by simple word frequencies. For the same text, a frequency approach provides the following top-ranked lexical units: systems (4), types (3), solutions (3), minimal (3), linear (2), inequations (2), algorithms (2). All other lexical units have a frequency of 1, and therefore cannot be ranked, but only listed.

3.2 Evaluation

The data set used in the experiments is a collection of 500 abstracts from the *Inspec* database, and the corresponding manually assigned keywords. This is the same test data set as used in the keyword extraction experiments reported in (Hulth, 2003). The *Inspec* abstracts are from journal papers from Computer Science and Information Technology. Each abstract comes with two sets of keywords assigned by professional indexers: controlled keywords, restricted to a given thesaurus, and uncontrolled keywords, freely assigned by the indexers. We follow the evaluation approach from (Hulth, 2003), and use the uncontrolled set of keywords.

In her experiments, Hulth is using a total of 2000 abstracts, divided into 1000 for training, 500 for development, and 500 for test[2]. Since our approach is completely unsupervised, no training/development data is required, and we are only using the test docu-

[2] Many thanks to Anette Hulth for allowing us to run our algorithm on the data set used in her keyword extraction experiments, and for making available the training/test/development data split.

Method	Assigned Total	Assigned Mean	Correct Total	Correct Mean	Precision	Recall	F-measure
TextRank							
Undirected, Co-occ.window=2	6,784	13.7	2,116	4.2	**31.2**	43.1	**36.2**
Undirected, Co-occ.window=3	6,715	13.4	1,897	3.8	28.2	38.6	32.6
Undirected, Co-occ.window=5	6,558	13.1	1,851	3.7	28.2	37.7	32.2
Undirected, Co-occ.window=10	6,570	13.1	1,846	3.7	28.1	37.6	32.2
Directed, forward, Co-occ.window=2	6,662	13.3	2,081	4.1	31.2	42.3	35.9
Directed, backward, Co-occ.window=2	6,636	13.3	2,082	4.1	31.2	42.3	35.9
Hulth (2003)							
Ngram with tag	7,815	15.6	1,973	3.9	25.2	**51.7**	33.9
NP-chunks with tag	4,788	9.6	1,421	2.8	29.7	37.2	33.0
Pattern with tag	7,012	14.0	1,523	3.1	21.7	39.9	28.1

Table 1: Results for automatic keyword extraction using TextRank or supervised learning (Hulth, 2003)

ments for evaluation purposes.

The results are evaluated using precision, recall, and F-measure. Notice that the maximum recall that can be achieved on this collection is less than 100%, since indexers were not limited to keyword extraction – as our system is – but they were also allowed to perform keyword generation, which eventually results in keywords that do not explicitly appear in the text.

For comparison purposes, we are using the results of the state-of-the-art keyword extraction system reported in (Hulth, 2003). Shortly, her system consists of a supervised learning scheme that attempts to learn how to best extract keywords from a document, by looking at a set of four features that are determined for each "candidate" keyword: (1) within-document frequency, (2) collection frequency, (3) relative position of the first occurrence, (4) sequence of part of speech tags. These features are extracted from both training and test data for all "candidate" keywords, where a candidate keyword can be: *Ngrams* (unigrams, bigrams, or trigrams extracted from the abstracts), *NP-chunks* (noun phrases), *patterns* (a set of part of speech patterns detected from the keywords attached to the training abstracts). The learning system is a rule induction system with bagging.

Our system consists of the TextRank approach described in Section 3.1, with a co-occurrence window-size set to two, three, five, or ten words. Table 1 lists the results obtained with TextRank, and the best results reported in (Hulth, 2003). For each method, the table lists the total number of keywords assigned, the mean number of keywords per abstract, the total number of correct keywords, as evaluated against the set of keywords assigned by professional indexers, and the mean number of correct keywords. The table also lists precision, recall, and F-measure.

Discussion. TextRank achieves the highest precision and F-measure across all systems, although the recall is not as high as in supervised methods – possibly due the limitation imposed by our approach on the number of keywords selected, which is not made in the supervised system[3]. A larger window does not seem to help – on the contrary, the larger the window, the lower the precision, probably explained by the fact that a relation between words that are further apart is not strong enough to define a connection in the text graph.

Experiments were performed with various syntactic filters, including: all open class words, nouns and adjectives, and nouns only, and the best performance was achieved with the filter that selects nouns and adjectives only. We have also experimented with a setting where no part of speech information was added to the text, and all words - except a predefined list of stopwords - were added to the graph. The results with this setting were significantly lower than the systems that consider part of speech information, which corroborates with previous observations that linguistic information helps the process of keyword extraction (Hulth, 2003).

Experiments were also performed with directed graphs, where a direction was set following the natural flow of the text, e.g. one candidate keyword "recommends" (and therefore has a directed arc to) the candidate keyword that follows in the text, keeping the restraint imposed by the co-occurrence relation. We have also tried the reversed direction, where a lexical unit points to a previous token in the text. Table 1 includes the results obtained with directed graphs for a co-occurrence window of 2. Regardless of the direction chosen for the arcs, results obtained with directed graphs are worse than results obtained with undirected graphs, which suggests that despite a natural flow in running text, there is no natural "direction" that can be established between co-

[3]The fact that the supervised system does not have the capability to set a cutoff threshold on the number of keywords, but it only makes a binary decision on each candidate word, has the downside of not allowing for a precision-recall curve, which prohibits a comparison of such curves for the two methods.

occurring words.

Overall, our TextRank system leads to an F-measure higher than any of the previously proposed systems. Notice that TextRank is completely unsupervised, and unlike other supervised systems, it relies exclusively on information drawn from the text itself, which makes it easily portable to other text collections, domains, and languages.

4 Sentence Extraction

The other TextRank application that we investigate consists of sentence extraction for automatic summarization. In a way, the problem of sentence extraction can be regarded as similar to keyword extraction, since both applications aim at identifying sequences that are more "representative" for the given text. In keyword extraction, the candidate text units consist of words or phrases, whereas in sentence extraction, we deal with entire sentences. TextRank turns out to be well suited for this type of applications, since it allows for a ranking over text units that is recursively computed based on information drawn from the entire text.

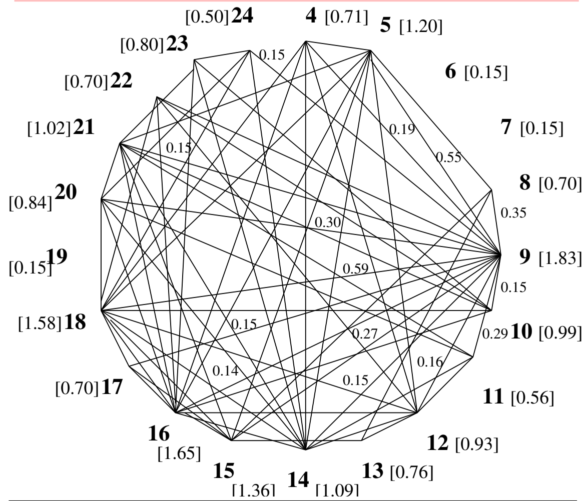
4.1 TextRank for Sentence Extraction

To apply TextRank, we first need to build a graph associated with the text, where the graph vertices are representative for the units to be ranked. For the task of sentence extraction, the goal is to rank entire sentences, and therefore a vertex is added to the graph for each sentence in the text.

The co-occurrence relation used for keyword extraction cannot be applied here, since the text units in consideration are significantly larger than one or few words, and "co-occurrence" is not a meaningful relation for such large contexts. Instead, we are defining a different relation, which determines a connection between two sentences if there is a "similarity" relation between them, where "similarity" is measured as a function of their content overlap. Such a relation between two sentences can be seen as a process of "recommendation": a sentence that addresses certain concepts in a text, gives the reader a "recommendation" to refer to other sentences in the text that address the same concepts, and therefore a link can be drawn between any two such sentences that share common content.

The overlap of two sentences can be determined simply as the number of common tokens between the lexical representations of the two sentences, or it can be run through syntactic filters, which only count words of a certain syntactic category, e.g. all open class words, nouns and verbs, etc. Moreover, to avoid promoting long sentences, we are using a normalization factor, and divide the content overlap

3: BC−HurricaineGilbert, 09−11 339
4: BC−Hurricaine Gilbert, 0348
5: Hurricaine Gilbert heads toward Dominican Coast
6: By Ruddy Gonzalez
7: Associated Press Writer
8: Santo Domingo, Dominican Republic (AP)
9: Hurricaine Gilbert Swept towrd the Dominican Republic Sunday, and the Civil Defense alerted its heavily populated south coast to prepare for high winds, heavy rains, and high seas.
10: The storm was approaching from the southeast with sustained winds of 75 mph gusting to 92 mph.
11: "There is no need for alarm," Civil Defense Director Eugenio Cabral said in a television alert shortly after midnight Saturday.
12: Cabral said residents of the province of Barahona should closely follow Gilbert's movement.
13: An estimated 100,000 people live in the province, including 70,000 in the city of Barahona, about 125 miles west of Santo Domingo.
14. Tropical storm Gilbert formed in the eastern Carribean and strenghtened into a hurricaine Saturday night.
15: The National Hurricaine Center in Miami reported its position at 2 a.m. Sunday at latitude 16.1 north, longitude 67.5 west, about 140 miles south of Ponce, Puerto Rico, and 200 miles southeast of Santo Domingo.
16: The National Weather Service in San Juan, Puerto Rico, said Gilbert was moving westard at 15 mph with a "broad area of cloudiness and heavy weather" rotating around the center of the storm.
17. The weather service issued a flash flood watch for Puerto Rico and the Virgin Islands until at least 6 p.m. Sunday.
18: Strong winds associated with the Gilbert brought coastal flooding, strong southeast winds, and up to 12 feet to Puerto Rico's south coast.
19: There were no reports on casualties.
20: San Juan, on the north coast, had heavy rains and gusts Saturday, but they subsided during the night.
21: On Saturday, Hurricane Florence was downgraded to a tropical storm, and its remnants pushed inland from the U.S. Gulf Coast.
22: Residents returned home, happy to find little damage from 90 mph winds and sheets of rain.
23: Florence, the sixth named storm of the 1988 Atlantic storm season, was the second hurricane.
24: The first, Debby, reached minimal hurricane strength briefly before hitting the Mexican coast last month.



TextRank extractive summary
Hurricane Gilbert swept toward the Dominican Republic Sunday, and the Civil Defense alerted its heavily populated south coast to prepare for high winds, heavy rains and high seas. The National Hurricane Center in Miami reported its position at 2 a.m. Sunday at latitude 16.1 north, longitude 67.5 west, about 140 miles south of Ponce, Puerto Rico, and 200 miles southeast of Santo Domingo. The National Weather Service in San Juan, Puerto Rico, said Gilbert was moving westward at 15 mph with a "broad area of cloudiness and heavy weather" rotating around the center of the storm. Strong winds associated with Gilbert brought coastal flooding, strong southeast winds and up to 12 feet to Puerto Rico's south coast.

Manual abstract I
Hurricane Gilbert is moving toward the Dominican Republic, where the residents of the south coast, especially the Barahona Province, have been alerted to prepare for heavy rains, and high wind and seas. Tropical storm Gilbert formed in the eastern Carribean and became a hurricane on Saturday night. By 2 a.m. Sunday it was about 200 miles southeast of Santo Domingo and moving westward at 15 mph with winds of 75 mph. Flooding is expected in Puerto Rico and in the Virgin Islands. The second hurricane of the season, Florence, is now over the southern United States and down−graded to a tropical storm.

Manual abstract II
Tropical storm Gilbert in the eastern Carribean strenghtened into a hurricane Saturday night. The National Hurricane Center in Miami reported its position at 2 a.m. Sunday to be about 140 miles south of Puerto Rico and 200 miles southeast of Santo Domingo. It is moving westward at 15 mph with a broad area of cloudiness and heavy weather with sustained winds of 75 mph gusting to 92 mph. The Dominican Republic's Civil Defense alerted that country's heavily populated south coast and the National Weather Service in San Juan, Puerto Rico issued a flood watch for Puerto Rico and the Virgin Islands until at least 6 p.m. Sunday.

Figure 3: Sample graph build for sentence extraction from a newspaper article. Manually assigned summaries and TextRank extractive summary are also shown.

of two sentences with the length of each sentence. Formally, given two sentences S_i and S_j, with a sentence being represented by the set of N_i words that appear in the sentence: $S_i = w_1^i, w_2^i, ..., w_{N_i}^i$, the similarity of S_i and S_j is defined as:

$$Similarity(S_i, S_j) = \frac{|\{w_k | w_k \in S_i \& w_k \in S_j\}|}{log(|S_i|) + log(|S_j|)}$$

Other sentence similarity measures, such as string kernels, cosine similarity, longest common subsequence, etc. are also possible, and we are currently evaluating their impact on the summarization performance.

The resulting graph is highly connected, with a weight associated with each edge, indicating the strength of the connections established between various sentence pairs in the text. The text is therefore represented as a weighted graph, and consequently we are using the weighted graph-based ranking formula introduced in Section 2.2.

After the ranking algorithm is run on the graph, sentences are sorted in reversed order of their score, and the top ranked sentences are selected for inclusion in the summary.

Figure 3 shows a text sample, and the associated weighted graph constructed for this text. The figure also shows sample weights attached to the edges connected to vertex 9[4], and the final TextRank score computed for each sentence. The sentences with the highest rank are selected for inclusion in the abstract. For this sample article, the sentences with id-s 9, 15, 16, 18 are extracted, resulting in a summary of about 100 words, which according to automatic evaluation measures, is ranked the second among summaries produced by 15 other systems (see Section 4.2 for evaluation methodology).

4.2 Evaluation

We evaluate the TextRank sentence extraction algorithm on a single-document summarization task, using 567 news articles provided during the Document Understanding Evaluations 2002 (DUC, 2002). For each article, TextRank generates an 100-words summary — the task undertaken by other systems participating in this single document summarization task.

For evaluation, we are using the ROUGE evaluation toolkit, which is a method based on Ngram statistics, found to be highly correlated with human evaluations (Lin and Hovy, 2003). Two manually produced reference summaries are provided, and used in the evaluation process[5].

Fifteen different systems participated in this task, and we compare the performance of TextRank with the top five performing systems, as well as with the baseline proposed by the DUC evaluators – consisting of a 100-word summary constructed by taking the first sentences in each article. Table 2 shows the results obtained on this data set of 567 news articles, including the results for TextRank (shown in bold), the baseline, and the results of the top five performing systems in the DUC 2002 single document summarization task (DUC, 2002).

System	ROUGE score – Ngram(1,1) basic (a)	stemmed (b)	stemmed no-stopwords (c)
S27	0.4814	0.5011	0.4405
S31	0.4715	0.4914	0.4160
TextRank	**0.4708**	**0.4904**	**0.4229**
S28	0.4703	0.4890	0.4346
S21	0.4683	0.4869	0.4222
Baseline	*0.4599*	*0.4779*	*0.4162*
S29	0.4502	0.4681	0.4019

Table 2: Results for single document summarization: TextRank, top 5 (out of 15) DUC 2002 systems, and baseline. Evaluation takes into account (a) all words; (b) stemmed words; (c) stemmed words, and no stopwords.

Discussion. TextRank succeeds in identifying the most important sentences in a text based on information exclusively drawn from the text itself. Unlike other supervised systems, which attempt to learn what makes a good summary by training on collections of summaries built for other articles, TextRank is fully unsupervised, and relies only on the given text to derive an extractive summary, which represents a summarization model closer to what humans are doing when producing an abstract for a given document.

Notice that TextRank goes beyond the sentence "connectivity" in a text. For instance, sentence 15 in the example provided in Figure 3 would not be identified as "important" based on the number of connections it has with other vertices in the graph, but it is identified as "important" by TextRank (and by humans – see the reference summaries displayed in the same figure).

Another important aspect of TextRank is that it gives a ranking over all sentences in a text – which means that it can be easily adapted to extracting very short summaries (headlines consisting of one

[4]Weights are listed to the right or above the edge they correspond to. Similar weights are computed for each edge in the graph, but are not displayed due to space restrictions.

[5]ROUGE is available at http://www.isi.edu/~cyl/ROUGE/.

The evaluation is done using the Ngram(1,1) setting of ROUGE, which was found to have the highest correlation with human judgments, at a confidence level of 95%. Only the first 100 words in each summary are considered.

sentence), or longer more explicative summaries, consisting of more than 100 words. We are also investigating combinations of keyphrase and sentence extraction techniques as a method for building short/long summaries.

Finally, another advantage of TextRank over previously proposed methods for building extractive summaries is the fact that it does not require training corpora, which makes it easily adaptable to other languages or domains.

5 Why TextRank Works

Intuitively, TextRank works well because it does not only rely on the local context of a text unit (vertex), but rather it takes into account information recursively drawn from the entire text (graph).

Through the graphs it builds on texts, TextRank identifies connections between various entities in a text, and implements the concept of *recommendation*. A text unit recommends other related text units, and the strength of the recommendation is recursively computed based on the importance of the units making the recommendation. For instance, in the keyphrase extraction application, co-occurring words recommend each other as important, and it is the common context that enables the identification of connections between words in text. In the process of identifying important sentences in a text, a sentence recommends another sentence that addresses similar concepts as being useful for the overall understanding of the text. The sentences that are highly recommended by other sentences in the text are likely to be more informative for the given text, and will be therefore given a higher score.

An analogy can be also drawn with PageRank's "random surfer model", where a user surfs the Web by following links from any given Web page. In the context of text modeling, TextRank implements what we refer to as "text surfing", which relates to the concept of text cohesion (Halliday and Hasan, 1976): from a certain concept C in a text, we are likely to "follow" links to connected concepts – that is, concepts that have a relation with the current concept C (be that a lexical or semantic relation). This also relates to the "knitting" phenomenon (Hobbs, 1974): facts associated with words are shared in different parts of the discourse, and such relationships serve to "knit the discourse together".

Through its iterative mechanism, TextRank goes beyond simple graph connectivity, and it is able to score text units based also on the "importance" of other text units they link to. The text units selected by TextRank for a given application are the ones most recommended by related text units in the text, with preference given to the recommendations made by most influential ones, i.e. the ones that are in turn highly recommended by other related units. The underlying hypothesis is that in a cohesive text fragment, related text units tend to form a "Web" of connections that approximates the model humans build about a given context in the process of discourse understanding.

6 Conclusions

In this paper, we introduced TextRank – a graph-based ranking model for text processing, and show how it can be successfully used for natural language applications. In particular, we proposed and evaluated two innovative unsupervised approaches for keyword and sentence extraction, and showed that the accuracy achieved by TextRank in these applications is competitive with that of previously proposed state-of-the-art algorithms. An important aspect of TextRank is that it does not require deep linguistic knowledge, nor domain or language specific annotated corpora, which makes it highly portable to other domains, genres, or languages.

References

S. Brin and L. Page. 1998. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7).

DUC. 2002. Document understanding conference 2002. http://www-nlpir.nist.gov/projects/duc/.

E. Frank, G. W. Paynter, I. H. Witten, C. Gutwin, and C. G. Nevill-Manning. 1999. Domain-specific keyphrase extraction. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*.

M. Halliday and R. Hasan. 1976. *Cohesion in English*. Longman.

P.J. Herings, G. van der Laan, and D. Talman. 2001. Measuring the power of nodes in digraphs. Technical report, Tinbergen Institute.

J. Hobbs. 1974. A model for natural language semantics. Part I: The model. Technical report, Yale University.

A. Hulth. 2003. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, Japan, August.

J.M. Kleinberg. 1999. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632.

C.Y. Lin and E.H. Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of Human Language Technology Conference (HLT-NAACL 2003)*, Edmonton, Canada, May.

R. Mihalcea, P. Tarau, and E. Figa. 2004. PageRank on semantic networks, with application to word sense disambiguation. In *Proceedings of the 20st International Conference on Computational Linguistics (COLING 2004)*, Geneva, Switzerland.

R. Mihalcea. 2004. Graph-based ranking algorithms for sentence extraction, applied to text summarization. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Lingusitics (ACL 2004) (companion volume)*, Barcelona, Spain.

P. Turney. 1999. Learning to extract keyphrases from text. Technical report, National Research Council, Institute for Information Technology.