



iJRASET

International Journal For Research in
Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 11 Issue: V Month of publication: May 2023

DOI: <https://doi.org/10.22214/ijraset.2023.50786>

www.ijraset.com

Call:  08813907089

E-mail ID: ijraset@gmail.com

Text Summarizer Using ML

Prof. Pushkar Joglekar¹, Om Randhave², Kshitij Meshram³, Harshwardhan Patil⁴, Aditya Nagrale⁵
Vishwakarma Institute of Technology (VIT Pune), Pune, Maharashtra, 411017, India

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.*

I. INTRODUCTION

Some of the various Graph-based ranking algorithms like Kleinberg's HITS algorithm or Google's PageRank have been successfully used in various citation analysis, social networks, and the analysis of the link-structure of the World Wide Web (WWW). Also, these algorithms can be singled out as the key elements of the paradigm-shift triggered in the field of Web search technology, by providing a convenient Web page ranking mechanism that also relies on the collective knowledge of Web architects rather than individual content analysis of various 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 that is calculated from the entire graph, rather than just relying on local vertex for specific information. When Applying a similar line of thinking to lexical as well as semantic graphs extracted from natural language documents, results in a graph based ranking model that can also be applied to a variety of natural language processing (NLP) applications, where knowledge drawn from an entire text is used in constructing various local ranking/selection decisions. Such text-oriented ranking methods can be applied to tasks ranging from automated extraction of various keyphrases, to extractive summarization and word sense disambiguation.

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

II. THE TEXTRANK MODEL

Graph based ranking algorithms that are essentially a suitable way of deciding the importance of a vertex within graphs, 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 vertex, it is basically casting a vote for that other vertex. The higher the number of votes cast for a vertex, the higher the importance of the vertex based on it too. Moreover, the importance of vertex casting the vote also determines that how important the vote itself is and this information is also taken into account by the ranking model. Hence, the score that is associated with a vertex is also determined based on the votes that are cast for it and the score of the vertices casting these votes too. Formally, let be a directed graph with the set of various vertices and set of edges linked to them. For a given specific vertex, let there be a set of vertices that point to it (predecessors), and let be the set of vertices that vertex points to (successors). The score of a vertex is defined as follows 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 specific vertex to another random vertex in graph. In the context of Web surfing, this graph based ranking algorithm implements the "random surfer model", where a user when clicks on links at random with a specific probability, and also jumps to a completely new page with probability. The factor is usually set to 0.85 and this is the value that we are also using in our implementation.

Starting from various arbitrary values that are specifically assigned to each node in the graph, the computation also iterates until convergence below a specific threshold is achieved too. 0.16

After executing the algorithm, a score is associated with each vertex, which also represents the "importance" of the vertex within the graph.

It is important to notice that although the TextRank applications described in this paper rely on an algorithm derived from Google's PageRank, other graph-based ranking algorithms such as e.g. HITS or Positional

A. Undirected Graphs

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

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.

B. Weighted Graphs

It may be therefore useful to indicate and incorporate into the model the “strength” of the connection between two vertices and as a weight added to the corresponding edge that connects the two vertices.

Convergence is achieved when the error rate for any given vertex in the graph falls below a given threshold. The error rate of a vertex is defined as the difference between the “real” score of the vertex and the score computed at iteration i . Since the real score is not known a priori, this error rate is approximated with the difference between the scores computed at two successive iterations:

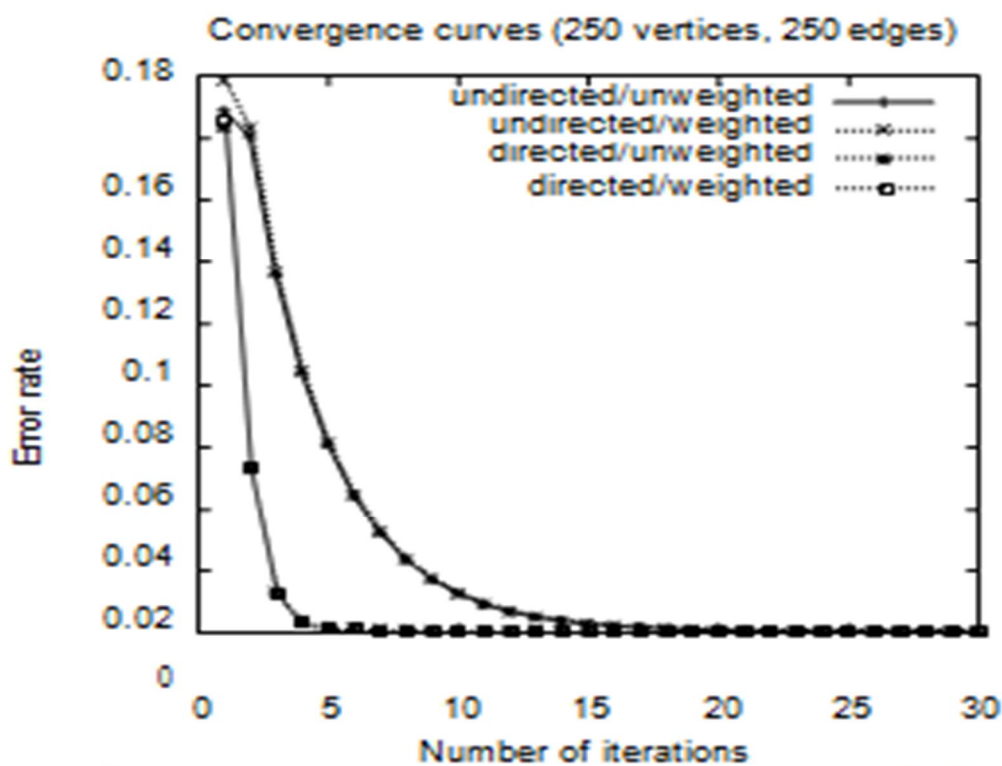


Figure 1: Convergence curves for graph based ranking: directed/undirected, weighted/unweighted graph, 250 vertices and 250 edges.

Consequently, we can also introduce a new formula based on graph ranking that takes into account edge that weights when computing the score associated with a specific vertex in the graph.

C. Text as a Graph

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

e.g. lexical/semantic relations, contextual overlaps, etc.

Regardless of the various types and characteristics of the elements that are added to the graph, the application of graph based ranking algorithms to natural language texts consists of the following main steps:

- 1) Identify the various text units that suitably define the task at hand, and add them as the vertices in the graph.
- 2) Also Identify the relations that connect such text units, and utilize these relations to draw the edges between the vertices in the graph. Edges can be also directed or undirected, weighted or unweighted too.
- 3) Recapitulate the graph based ranking algorithm until convergence.

1. Sort vertices based on their final score. Use the values attached to each vertex for ranking/selection decisions.

We also investigate and evaluate the applications of TextRank related to two natural language processing tasks which involves ranking of text units: (1) A keyword extraction task, which consists 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.

III. 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 that are helpful for building an automatic index for a document collection that can also be used to classify a text, or may also serve as a concise summary for a given document. Moreover, a system functioning for automatic identification of important terms in a text can be suitably 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. A different learning algorithm with a different approach was also used, where a Naive Bayes learning based scheme is applied on the document collection, with improved results observed on the same data set that was also 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.

In this given section, we can report on our experiments in keyword extraction using TextRank algorithm, and demonstrate that the graph based ranking model outperforms the best published results in this problem. We are evaluating our algorithm on keyword extraction from abstracts, mainly for the purpose of allowing for a direct comparison with the results it reports with its keyphrase extraction system. Notice that the size of the text is not a limitation that is imposed by our system, and similar results are also expected with TextRank applied on full-texts.

A. TextRank for Keyword Extraction

The expected end result for this specific application is a set of words or some phrases that are also 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. Cooccurrence links express various relations between syntactic elements and similar to the semantic links which are found useful for the task of word sense disambiguation, they represent cohesion indicators for a given text. The vertices added to the graph can also be restricted with various syntactic filters, which also 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 completely unsupervised, and proceeds as follows. First, the text is tokenized in a suitable manner and also annotated with various parts of speech tags – a preprocessing step required to enable the application of various 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 some of the candidates for addition to the graph, with various multiword keywords being eventually reconstructed in the post processing phase too.

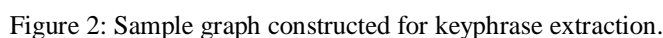


Figure 2 shows a sample graph built for an abstract from our test collection. While the size of the abstracts ranges from 50 to 350 words, with an average size of 120 words, we have deliberately selected a very small abstract for the purpose of illustration. For this example, the lexical units found 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), dio-phantine (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 of the other lexical units also have a specific frequency of 1, and therefore they cannot be ranked, but only listed.

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 also comes with two sets of the keywords which are assigned by professional indexers: i.e. controlled keywords, restricted to a given thesaurus, as well as uncontrolled keywords, freely assigned by the indexers too. Since our approach is completely unsupervised, no training/development data is required, and we are only using the test documents for evaluation purposes.

Method	Assigned		Correct		Precision	Recall	F-measure
	Total	Mean	Total	Mean			
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 the automatic keyword extraction using TextRank Algorithm or Supervised Learning.

The maximum recall that can also be achieved in this collection is also less than 100%, since various indexers which were not limited to keyword extraction as our system is but they were allowed to perform keyword generation, which eventually results in a suitable way where the 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 by looking at a set of the four features that are determined for each of the “candidate” keyword: (1) within-document frequency, (2) collection frequency, (3) relative position of the first occurrence, (4) sequence of part of speech tags. These some of the features are extracted from both training as well as test data for all of the “candidate” keywords, where a candidate keyword can also be: *NP-chunks also the* (noun phrases), *patterns also* (a set of part of speech patterns which are also detected from the keywords that are attached to the training abstracts).

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 of the demonstrated method, the table also lists the total number of keywords that are assigned, the mean number of the keywords per abstract, the total number of the correct keywords, as also evaluated against the set of the keywords assigned by professional indexers as well as the mean number of correct keywords.

TextRank achieves the highest precision and F-measure across all systems, although the recall is not as high as in supervised methods – possibly due to the limitations which are imposed by our approach on the number of keywords which are selected, which is not constructed in the supervised system. 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. 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, 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-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.

IV. SENTENCE EXTRACTION

In a suitable way, the issue of sentence extraction can be regarded as similar to keyword extraction, since both of the applications aim at identifying sequences that are more “representative” for the given text too. In keyword extraction approach, the candidate text units consist of various words or phrases, whereas in sentence extraction approach, we deal with entire sentences. TextRank also turns out to be well mannerly suited for these types of applications, since it also allows for a ranking over the text units that is recursively computed which are based on information drawn from the entire text.

A. 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 functioning of sentence extraction, the goal is to rank the entire sentences, and therefore a vertex too is also added to the graph for each of the 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 the two of the sentences can suitably be seen as a process of “recommendation” where: a sentence that addresses certain types of concepts in a text.

The overlapping of the two sentences can also be determined as simply as the number of common tokens between the lexical representations of the two sentences, and it can be processed through the syntactic filters, that only count words of a certain syntactic category, e.g. all of the open class words, nouns and verbs too, etc. Moreover, to also avoid promoting long sentences, we are using a normalization factor, as well as divide the content overlapping of the two sentences with length of each given sentence. Formally, given two sentences and, with a sentence being represented by the set of words that appear in the sentence: the similarity of and is defined as:

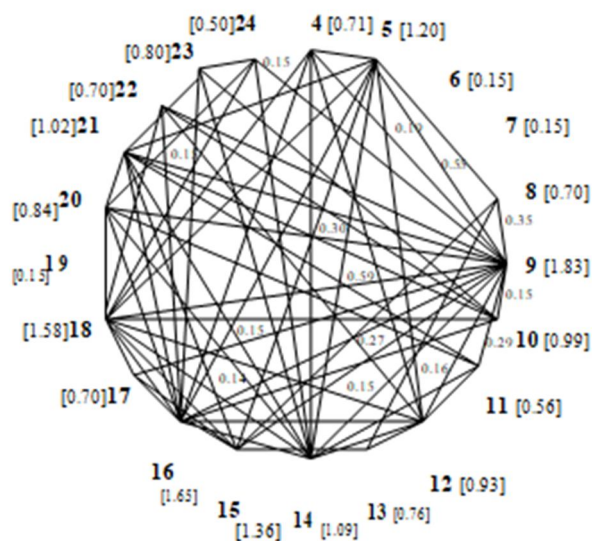


Figure 3: Sample graph constructed for sentence extraction from a newspaper article.

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

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 procedure of the ranking algorithm is executed on the graph, sentences are then sorted in the reversed order of their scores, and the top of the ranked sentences are also selected for the procedure of inclusion in the summary.

The figure also shows sample weights attached to the edges connected to vertex 9⁴, and the final TextRank score computed for each sentence. The given sentences with the highest rank are then selected for inclusion.

B. Evaluation

For each article, TextRank generates an 100-words summary — the task undertaken by other systems participating in this single document summarization task.

Two manually produced reference summaries are provided, and used in the evaluation process.

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 below also shows the results obtained on this data set of the various 567 news articles, including the results for TextRank also (shown in bold), baseline, and the various results of the top five performing systems in the DUC 2002 single document summarization task.

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
<i>Baseline</i>	<i>0.4599</i>	<i>0.4779</i>	<i>0.4162</i>
S29	0.4502	0.4681	0.4019

Table 2: Based on the Results for single document summarization: Evaluation takes into account (a) based on all of the words; (b) based on the stemmed words; (c) based on the stemmed words, and based on the no stop- words.

TextRank succeeds in identifying the most of the important sentences in a text based on the information exclusively drawn from the text itself. Unlike some of the other supervised systems, which also attempt to learn what makes a good summary by training based on the collections of summaries built for various other articles, TextRank is completely 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 foremost important aspect of TextRank is that it allocates a ranking over all of the sentences in a text – which indicates that it can be easily adapted to extracting some of the very short summaries (headlines consisting of one sentences), or more longer explicative summaries which consists 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 of the major advantage of TextRank over previously proposed methods for constructing extractive summaries is the fact that it does not require training corpora, which also makes it easily adaptable to other of the languages or domains too.

V. WHY TEXTRANK WORKS

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 various of the keyphrase extraction applications, co-occurring words also recommend each other as the important ones, and it is the common context that also enables the identification of connections between the words in text. An analogy can be also drawn with the help of the PageRank’s “random surfer model”, where a user surfs the Web followed by the 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. Also through its iterative mechanism, TextRank suitably goes beyond simple graph connectivity, and it is also able to score the text units based on the “importance” of other text units they link to.

The text units that are selected by TextRank for a given application are the ones which are most recommended by the related text units in the text, with foremost preference given to the recommendations made by most influential ones. 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 constructed about a given context in the process of discourse understanding too.

VI. 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

- [1] S. Brin and L. Page. 1998. Computer Networks and ISDN Systems, 30(1–7).
- [2] DUC. 2002. Document understanding conference 2002. <http://www-nlpir.nist.gov/projects/duc/>.
- [3] E. Frank, G. W. Paynter, I. H. Witten, C. Gutwin, and C. G. Nevill-Manning. 1999. Domain-specific keyphrase extraction.
- [4] M. Halliday and R. Hasan. 1976. Cohesion in English. Longman.
- [5] P.J. Herings, G. van der Laan, and D. Talman. 2001. Measuring the power of nodes in digraphs. Technical report, Tinbergen Institute.
- [6] J. Hobbs. 1974. A model for natural language semantics. Part I: The model. Technical report, Yale University.
- [7] A. Hulth. 2003.
- [8] J.M. Kleinberg. 1999. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632.
- [9] R. Mihalcea. 2004. P. Turney. 1999. Learning to extract keyphrases from text.



10.22214/IJRASET



45.98



IMPACT FACTOR:
7.129



IMPACT FACTOR:
7.429



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089  (24*7 Support on Whatsapp)