The Pennsylvania State University
The Graduate School

AUTOMATIC KEYPHRASE EXTRACTION FROM SCHOLARLY DOCUMENTS

A Thesis in
Computer Science and Engineering
by
Kyriaki Zafeiroudi

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

December 2019

The thesis of Kyriaki Zafeiroudi was reviewed and approved* by the following:

Clyde Lee Giles
David Reese Professor at the College of Information Sciences and Technology
Thesis Adviser

Daniel Kifer
Associate Professor of Computer Science and Engineering

Chitaranjan Das
Distinguished Professor of Computer Science and Engineering
Department Head of Computer Science and Engineering

*Signatures are on file in the Graduate School.

Abstract

Keyphrase extraction is a major natural language processing and information retrieval task, that although important for the advancement of many core tasks in the respective fields, is lacking a true solution. Different approaches have been researched, both supervised and unsupervised. We are proposing a new automatic unsupervised approach to perform keyphrase extraction from scholarly documents by leveraging graph-based ranking algorithms (PageRank) and the progress of word representation in a high dimensional space in the form of word embeddings. Our method is novel, while relying on the foundation of previous successful methods, and aiming to achieve better performance by combining methods used in other natural language processing tasks.

Keywords: Keyphrase extraction, PageRank, word embeddings

Table of Contents

List of Figures

List of Tables

List of Symbols

α	The damping factor in the calculation of PageRank, p. 7
$\widetilde{p}_i$	The position-biased normalized score of a candidate keyphrase c_i, p. 12
$\widetilde{t}_i$	The theme-biased normalized score of a candidate keyphrase c_i, p. 13
$S_{pos}(c_i)$	The position-biased PageRank score of a candidate keyphrase c_i, p. 14
$S_{theme}(c_i)$	The theme-biased PageRank score of a candidate keyphrase c_i, p. 15
$S(c_i)$	The final PageRank score of a candidate keyphrase c_i, biased by both the position and the theme, p. 16

Acknowledgments

I am very grateful to Dr. Cornelia Caragea for the idea of incorporating two existing methods in the field of keyphrase extraction. Her insightful idea laid the foundation for this Thesis. I am also thankful to Debanjan Mahata for pointing me in the right direction with regards to NLP toolkits and existing language models. Their continuous support was incredibly valuable, with many references to recent work helping make this work well-rounded. Finally, but most importantly, Dr. C. Lee Giles' guidance was invaluable throughout the time that I worked under his supervision.

Chapter 1
Introduction

Keyphrase extraction is a core natural language processing task that concerns *the automatic selection of important and topical phrases from the body of a document* [2]. Automatically extracting a small set of representative keyphrases from a document is a challenging task that has received a lot of attention due to its importance in facilitating various natural language processing and information retrieval tasks. In this chapter we are presenting the research conducted in the area of automatic keyphrase extraction and the different approaches that have been developed to-date, followed by an introduction to two other relevant topics in the NLP field that can be leveraged in the task of keyphrase extraction: ranking methods and word embeddings.

1.1 Keyphrase Extraction

Keyphrase extraction has applications in various natural language processing tasks, such as text summarization, text categorization, opinion mining, and document indexing [3]. The most common use of keyphrases exists in scholarly documents for the purpose of identifying the main topics discussed in the document [4]. Although the importance of the task is unquestionable, the proposed solutions have not yet reached the same level of performance that other natural language processing tasks have achieved.

There are two main approaches in the study of keyphrase extraction: supervised and unsupervised methods. Results from supervised methods applied on scientific documents are consistently showing higher accuracy than the state-of-the-art unsupervised methods [1]. These methods are more powerful, usually approaching the keyphrase extraction task as a binary classification task (labeling candidate phrases as keyphrases versus non-keyphrases). The drawback of the supervised approach is that it needs an existing corpus of annotated data, which is hard and costly to generate. The unsupervised

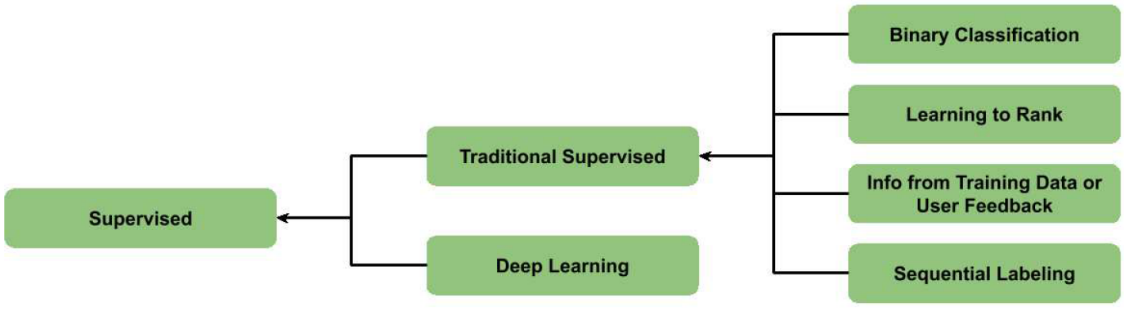

Figure 1.1. Categorization of supervised approaches to keyphrase extraction [1].

approaches are gaining popularity since there is no need for labeled data and thus they can be applied to documents from different domains, due to the independence of their methodology.

1.1.1 Supervised Methods

According to a survey conducted by Hasan and Ng [3], supervised approaches can be categorized in the following two groups:

- Task Reformulation: assessing a keyphrase extraction problem as a classification problem [2, 5], and furthermore expand this approach to a ranking method applied to keyphrases
- Features: representing an instance by using features that are computed from:
  - Within the collection of the training documents, e.g. statistical, structural and syntactic features, or
  - External resources, e.g. measuring the keyphraseness of a candidate by mapping it to Wikipedia [6] or search engine query logs [7]

A different, more recent, survey conducted by Papagiannopoulou and Tsoumakas [1], identified a different structure of supervised keyphrase extraction methods, as shown in Fig. 1.1.

For the present Thesis, we are going to focus solely on unsupervised approaches to keyphrase extraction.

1.1.2 Unsupervised Methods

In unsupervised approaches, the keyphrase extraction process usually comprises of the following steps:

1. Identify candidate phrases from the document based on heuristics, e.g. choose phrases that occur with specific part-of-speech (POS) tags, exclude determiners, punctuation, phrases from a predefined set of stopwords.

2. Rank the selected candidate phrases and select the top-ranked ones as keyphrases according to a user-defined threshold.

The unsupervised part of each approach applies both on the selection of the candidate phrases by automatically choosing them based on predefined in the system heuristics, and on the ranking algorithm that ultimately selects the top candidates to return to the user as keyphrases for a given document.

The aforementioned survey from Hasan and Ng [3] categorized unsupervised approaches in the following four groups:

- Graph-based: representing the candidate keyphrases from a document as nodes in a graph and applying graph ranking algorithms [see Section 1.2] to compute the importance of each candidate keyphrase [8–10]

- Topic-based: identifying topics in a document and assign each candidate keyphrase to one topic [11]

- Simultaneous Learning: as previously mentioned, the task of text summarization can leverage keyphrase extraction, making the assumption that a sentence is important analogously to the importance of the words it contains, and vice versa [12], thus suggesting the combination of the two tasks

- Language Modeling: computing the phraseness and informativeness of phrases in the document following the approach used in language models.

Interestingly, the survey conducted by Papagiannopoulou and Tsoumakas [1], is proposing again a different structure of unsupervised keyphrase extraction methods, as shown in Fig. 1.2, where the graph-based methods are considered to be a more inclusive umbrella category that also incorporates the topic-based methods, as described by Hasan and Ng [3].

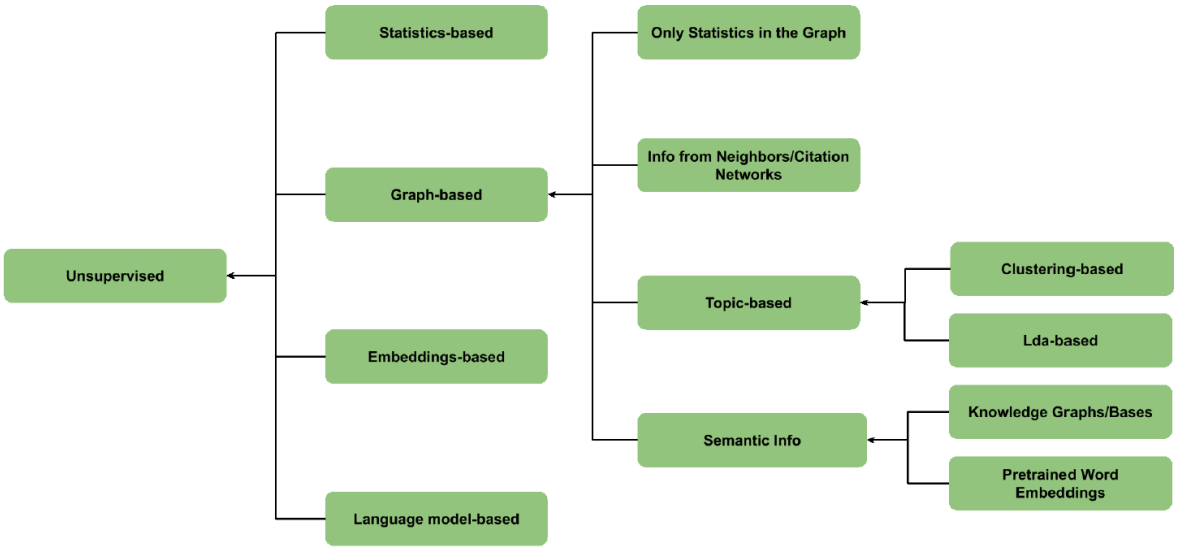

Figure 1.2. Categorization of unsupervised approaches to keyphrase extraction [1].

We will now dive deeper into heuristics commonly used for candidate selection, followed by graph-based ranking methods, in order to build the foundation for our research proposal.

1.1.2.1 Heuristics for Candidate Selection

In order to perform keyphrase extraction, first we need to preprocess the given document in search of candidate phrases. The most common heuristics that we have identified in previously proposed systems [9, 10], are the following:

- Filter out fully numeric lexical units and named entities belonging to the following categories: DATE, TIME, PERCENT, MONEY, QUANTITY, ORDINAL, CARDINAL
- Filter out a standard set of stopwords
- Remove final punctuation, such as period, exclamation point, question mark
- Remove determiners from phrases
- Remove words that fall under a specific set of parts of speech, such as interjection, auxiliary, pronouns
- Normalize the text, e.g. non-matching parentheses, quotes, whitespace

1.1.2.2 Graph-based Ranking Methods

Incorporating a graph-based ranking algorithm for the task of keyphrase extraction is an idea implemented in many previous systems [8–10, 13–15]. Keyphrase extraction focuses on identifying the important words and phrases in a document, where the importance of one candidate keyphrase can be measured in terms of relatedness to other candidates in the same context. The relatedness between candidates is usually calculated in terms of co-occurrence of the candidates in a predefined window of subsequent words in the document [8, 9, 13, 14], and more recently as the semantic similarity of the candidate keyphrases, or their similarity to a main theme identified for each document [10, 15].

The idea of incorporating graph-based ranking for the task of keyphrase extraction relies on the creation of a graph from a given document after identifying a set of candidate phrases. Those candidate phrases will become later the nodes in the graph, connected to one-another through edges in the graph, which indicate that two candidate keyphrases are related, usually because of some word co-occurrence metric. The edges can be unweighted [8], or weighted [9, 10, 13–15], depending on the way the relation between two candidate phrases is determined. In the case of SingleRank [13], weighted edges are introduced to capture the relation between candidate words that co-occur in a window of size $w \geq 2$, while in ExpandRank [13] a similar idea is implemented for neighboring documents. An extension of ExpandRank introduced in [14] integrates additional information using citation networks. In PositionRank [9], the edges of the graph are initialized to weights indicating the co-occurrence count of two candidate keyphrases c_i^d and c_j^d within a window w of successive words in d. The method introduced in [15] assigns weights depending on the semantic relatedness of the candidate phrases, exploiting information extracted from Wikipedia, while in Key2Vec [10], the weight of an edge between two candidate keyphrases c_i^d and c_j^d is calculated based on their semantic similarity $semantic(c_i^d, c_j^d)$ (a metric incorporating their cosine distance) shown in Eq. (1.1) and their co-occurrence $cooccur(c_i^d, c_j^d)$ (a metric incorporating their Point-wise Mutual Information) shown in Eq. (1.2). The edge weight connecting candidate keyphrases c_i^d and c_j^d is finally computed by Eq. (1.3).

$$semantic(c_i^d, c_j^d) = \frac{1}{1 - cosine(c_i^d, c_j^d)}, \tag{1.1}$$

$$cooccur(c_i^d, c_j^d) = PMI(c_i^d, c_j^d), \tag{1.2}$$

$$sr(c_i^d, c_j^d) = semantic(c_i^d, c_j^d) \times cooccur(c_i^d, c_j^d) \tag{1.3}$$

After building the graph, we identify the best existing graph-based ranking algorithm to calculate the score of each node and choose the top-ranked keyphrases for the document. Some common graph-based ranking approaches are PageRank [16], Positional Function [17], and Hypertext Induced Topic Search (HITS) [18], with PageRank being the most commonly used for the task of keyphrase extraction. We will discuss more about each ranking algorithm in Section 1.2.

1.2 Ranking Algorithms

Various ranking algorithms have been proposed in literature with the property of converging when applied iteratively over a graph. This research generally exists within the subfield of computer science that focuses on network analysis. The following three approaches have been identified in literature related to keyphrase extraction, when the documents are represented as graphs:

- PageRank [16]: the most popular algorithm to-date, exploiting the flow of the World Wide Web viewed as a graph, to identify the importance of each website and rank them accordingly
- Hypertext Induced Topic Search (HITS) or Hubs and Authorities [18]: a link analysis algorithm with the purpose of calculating the value of each node as a hub and as an authority
- Positional Power Function [17]: a digraph algorithm calculating the power of each node by both the number of its successors and the power of its successors iteratively

We will focus on the PageRank algorithm and how it can be leveraged to achieve better accuracy in a keyphrase extraction task.

1.2.1 PageRank

The PageRank algorithm was developed in order to be used by the Google search engine by ranking the websites that a query returns based on their importance. The importance of each web page is calculated with regards to the link structure of the web and more precisely, the sum of the back-links of the page. The algorithm is based on the random surfer model, according to which we work on the assumption that a user follows links existing in a page by choosing them randomly, without any preexisting bias towards a particular web page.

The scoring method of the nodes of a graph given by PageRank is given in Eq. (1.4), where $S(v_i)$ is the calculated score of a node, α is the damping factor that allows the algorithm to converge faster, $N(v_i)$ is the set of nodes neighboring v_i. This is the simplest form of PageRank, where the edges in the graph are not weighted.

Eq. (1.5) takes into account the weight w_{ij} of the edge between two nodes v_i and v_j, while Eq. (1.6) introduces a bias in the calculation of PageRank in the form of p_i for a node v_i, which indicates the probability of randomly jumping to v_i. It is usually preferred for the values of p_i and w_{ij} to be normalized.

$$S(v_i) = (1 - \alpha) + \alpha \cdot \sum_{j \in N(v_i)} \frac{1}{N(v_i)} S(v_j), \tag{1.4}$$

$$S(v_i) = (1 - \alpha) + \alpha \cdot \sum_{j \in N(v_i)} \frac{w_{ji}}{N(v_j)} S(v_j), \tag{1.5}$$

$$S(v_i) = (1 - \alpha) \cdot p_i + \alpha \cdot \sum_{j \in N(v_i)} \frac{w_{ji}}{N(v_j)} S(v_j) \tag{1.6}$$

1.3 Word Embeddings

Recent advancements in the area of natural language processing have been enabled by the introduction of the notion of word embeddings; word embeddings are arithmetic vector representations of words, defined by the exploration of very large datasets. This technique captures the semantic and syntactic similarities between words, and leveraging them has proven to be extremely significant for multiple natural language processing tasks, including that of keyphrase extraction.

Based on this idea of generating word vectors that have the ability of describing accurately the semantic and syntactic existence of a word in a high dimensional space, multiple approaches to train word embeddings have been proposed in the recent years:

- Word2Vec [19] and GloVe [20] were the first generation of representing words in a high dimensional vector space. These models focus on the idea that identifying word co-occurrence in context can capture semantic and syntactic similarities between words.

- FastText [21] introduced the idea of considering the sequence of the occurrence of words, additionally capturing morphological similarities between words.

- BERT [22] extends FastText by incorporating a bi-directional architecture, introducing the idea of contextual embeddings.
- SciBERT [23] and BioBERT [24] are the newest domain-specific language models. Utilizing the BERT architecture, SciBERT and BioBERT are trained on scientific papers in both the computer science and biomedical domain, and only the biomedical domain, respectively.

The presented Thesis takes advantage of the language model introduced by SciBERT, which captures the semantic, contextual and morphological similarities between words, as captured from pre-training on scientific papers.

Chapter 2
Proposed Model

In this chapter, we are presenting our proposed model, **K**ey-**O**ption **R**ank (KOR). KOR is a fully unsupervised, graph-based model, that exercises the option of biasing the importance of each word towards the word's position or towards the contextual semantic similarity between the word and the main theme of the document, in order to calculate a biased PageRank score for each candidate keyphrase.

2.1 Overview

The idea of graph-based ranking algorithms like PageRank [16] is to assign an importance score to each node by performing a random walk recursively over the graph in order to incorporate all information encoded within the graph from all nodes and edges. Given a bias, we choose how much to shift the focus from the phrase's position towards the phrase's semantic relatedness to the thematic vector of the document. The weight of

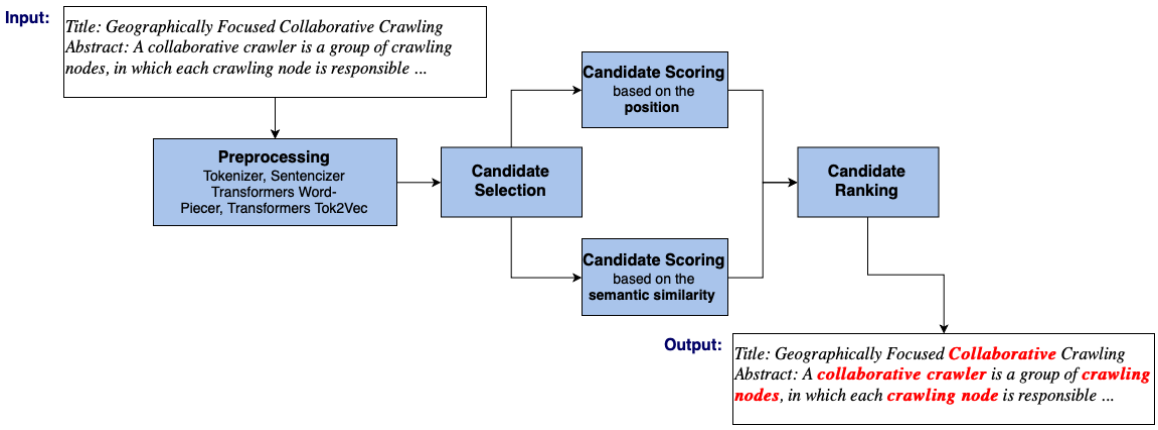

Figure 2.1. The KOR keyphrase extraction system.

each candidate keyphrase is then calculated by incorporating information from all the different positions and contexts the phrase occurs in, partially based on the distance of the phrase from the beginning of the document and partially on the cosine similarity between the contextual phrase embedding of each phrase's occurrence and the thematic vector of the document. Each partial weight is then inserted into a biased PageRank algorithm, which in turn computes an "authoritative" score to each candidate phrase based on the approach. Figure 2.1 showcases the four steps in KOR's pipeline: Initially, we are preprocessing the input text, followed by the selection of candidate phrases that we then score and rank before returning a list of the top ranked (predicted) keyphrases.

2.2 Text Preprocessing

Given an input document d, we follow a specific set of text processing steps. Using the spaCy[1] and textacy[2] NLP toolkits to parse the input text, we first apply a tokenizer, followed by a sentencizer in order to assign segment boundaries on the token and sentence level respectively. The choice of spaCy is based on the several implementations that can be conveniently incorporated into their parsing pipeline. We specifically use a language model, SciBERT [23], that makes use of Transformer, an attention mechanism whose encoder reads the full text in a bidirectional manner, therefore learning the contextual relations between words and phrases in text [25]. The model architecture is based on a multi-layer bidirectional Transformer, trained to:

- Predict randomly masked tokens.
- Predict whether two sentences follow each other.

We include spaCy's wrapping library of Transformer models in our implementation in order to incorporate the SciBERT language model. SciBERT is based on the BERT [22] architecture and trained on a domain specific corpus, specifically a corpus of 1.14 million scientific articles, making it the most suitable language model for the task of keyphrase extraction from scholarly documents. The choice of SciBERT as the preferred language model also arises from recent work [26] on keyphrase extraction that uses contextual embeddings, where SciBERT showed a consistently high F1-score on three separate datasets.

[1] `https://spacy.io`
[2] `https://chartbeat-labs.github.io/textacy/index.html`

Transformer models like SciBERT are usually trained on text that has been preprocessed with the WordPiece algorithm [27], limiting the vocabulary size, which is one of the key challenges that neural language models are facing [28]. Consequently, we are choosing to incorporate Transformers Word-Piecer as part of our text preprocessing. Lastly, we are including Transformers Tok2Vec which assigns token vector and in accordance to SciBERT, lets us predict contextual token representations.

2.3 Candidate Selection

Incorporated in spaCy's default model, we extract the POS tags of each token in d. In order to then extract candidate keyphrases, we follow steps frequently used in previous work [9, 10]:

- Extract noun phrases from d using the following regular expression:

  <ADV>?(<ADJ>|<NOUN>)*(<NOUN><PART>)?(<ADJ>|<NOUN>)*<NOUN>

- Include unigrams with a POS tag included in the following set:

  NN, NNS, NNP, NNPS, JJ

- Filter out fully numeric lexical units and named entities belonging to the following categories:

  DATE, TIME, PERCENT, MONEY, QUANTITY, ORDINAL, CARDINAL

- Filter tokens that were identified as stopwords by spaCy.

- Remove final punctuation, such as period, exclamation point, question mark.

- Remove determiners from phrases.

- Remove words that fall under a specific set of parts of speech, such as interjection, auxiliary, pronouns

- Normalize the text, e.g. non-matching parentheses, quotes, whitespace.

After the selection process, we have chosen a set of candidate phrases for document d:

$$C_d = \{c_1, c_2, ..., c_n\}$$

2.4 Candidate Scoring

After extracting candidate keyphrases from the document, we assign a score to each candidate using two separate approaches:

- A weight capturing the position the candidate occurs in the document.
- A score signifying the semantic relatedness of the candidate to the theme vector of the document.

2.4.1 Position-biased Candidate Scoring

The idea of introducing a score for each candidate phrase based on its position is attributed to the PositionRank method [9]. Under the assumption that words that occur frequently and early in a document d are more likely to be keyphrases, we assign to each candidate a score analogous to the sum of the inverse of each position the phrase occurs in. If a candidate phrase c_i occurs in positions $l_1, l_2, ..., l_n$, then the weight of $c_i \in C_d$ is calculated as:

$$p_i = \frac{1}{l_1} + \frac{1}{l_2} + ... + \frac{1}{l_n} \tag{2.1}$$

The positions $l_1, l_2, ..., l_n$ have been identified for each phrase in the document before filtering out words. Finally, we normalize the scores over all candidates, resulting in $\widetilde{p_i}$:

$$\widetilde{p_i} = \frac{p_i}{p_1 + p_2 + ... + p_{|V|}} \tag{2.2}$$

where $|V|$ is the number of unique candidate phrases and refers to their later representation as nodes in the graph.

2.4.2 Theme-biased Candidate Scoring

According to a separate methodology, Key2Vec [10], we are looking at the semantic similarity of a candidate phrase to the theme of the document. For a document d we choose τ_d to be the thematic vector of d. The span of sentences chosen to be the *theme excerpt* for a given document, similarly to [10], is transformed to a dense vector representation using the Transformers model as mentioned in Section 2.2. This contextual theme embedding represents the thematic vector of d.

The thematic vector is then used to calculate the cosine similarity between a candidate phrase and the theme of the document. Since we are utilizing SciBERT, which offers contextual embeddings, the embedding representation of the same phrase occurring in different contexts will be different, and thus we average the phrase embedding vectors for a candidate phrase before calculating the cosine similarity to the thematic vector resulting in the score t_i for candidate $c_i \in C_d$:

$$t_i = cosine(c_i, \tau_d) \tag{2.3}$$

The cosine distance between two vectors assigns a score in $[0, 1]$ to each candidate, with 1 indicating a complete similarity and 0 a complete dissimilarity. We will again normalize the scores for each candidate, so as to more accurately capture the difference between the candidates closest to the thematic vector of d and candidates farthest, in case the scores show a more dense probability at intermediate values, resulting in scores $\widetilde{t_i}$ for each candidate c_i.

2.5 Candidate Ranking

The idea of incorporating graph-based ranking for the task of keyphrase extraction relies on the creation of a graph from a given document after identifying the set of candidate phrases. After calculating personalized scores for each candidate phrase following the two approaches described above, we are constructing two separate graphs that incorporate the position- and theme-biased scores respectively. The candidates we have extracted become the nodes of each graph, connected to one-another if the co-occur in a window of w successive words.

For a document d, we build two graphs $G = (V, E)$ with the selected candidate keyphrases, such that each unique phrase represents a node $v_i \in V$ within the graph G, $v_i = c_i$. Two nodes v_i and v_j, where $v_i = c_i$ and $v_j = c_j$, are connected by an edge $e_{ij} = (v_i, v_j) = (c_i, c_j) \in E$ depending on whether the two phrases co-occur in a window of size w. The edges are weighted w_{ij}, sr_{ij} depending on the bias parameter of the model, oscillating between a raw co-occurrence indicator [9] and a weight partially influenced by the semantic relatedness of c_i and c_j [10]. The graphs are chosen to be undirected, since it has been shown in [8] that whether the edges are directed or undirected, there is no significant difference in the performance of a keyphrase extraction method.

Below we are describing how each edge weight is assigned for the position-biased PageRank calculated on graph G_1 and the theme-biased PageRank on graph G_2. The

nodes V in each graph remain the same.

2.5.1 Position-biased Candidate Ranking

For the graph $G_1 = (V, E_1)$, the edge weight connecting candidates c_i and c_j is calculated on the frequency of the two phrases co-occurring in a window w of contiguous tokens in d, resulting in w_{ij}. We then calculate a biased PageRank as described in Section 1.2, and particularly by using Equation 1.6. $S_{pos}(c_i)$ is the resulting score for a candidate phrase c_i, which is calculated iteratively using:

$$S_{pos}(c_i) = (1 - \alpha) \cdot \widetilde{p}_i + \alpha \cdot \sum_{c_j \in Adj(c_i)} \frac{w_{ij}}{O(c_j)} S_{pos}(c_j) \tag{2.4}$$

where $O(c_j) = \sum_{c_k \in Adj(c_j)} w_{jk}$ and $\widetilde{p}_i$ is the normalized initial score for c_i as described in Eq. 2.2.

2.5.2 Theme-biased Candidate Ranking

For the graph $G_2 = (V, E_2)$, the edge weight sr_{ij} connecting two candidate phrases c_i and c_j is computed similarly to the Key2Vec model [10], as described in Section 1.1.2.2. We first compute the semantic similarity of the two candidate phrases c_i and c_j as the cosine similarity of their contextual representation, as shown in Eq. 2.5:

$$semantic(c_i, c_j) = cosine(c_i, c_j) \tag{2.5}$$

Then, we calculate the co-occurrence of c_i and c_j in terms of Point-wise Mutual Information (PMI). The question that PMI between two words tries to address is whether they co-occur more often than if they were independent [29]. PMI between phrases c_i and c_j is calculated by Eq. 2.6:

$$PMI(c_i, c_j) = log_2 \frac{P(c_i, c_j)}{P(c_i) \cdot P(c_j)} \tag{2.6}$$

As the probability of a phrase occuring, we are calculating the number of sentences in a document d that c_i occurs in over the number of sentences that exist in the document, as described by Eq. 2.7.

$$P(c_i) = \frac{\#\ sentences\ c_i\ occurs}{\#\ sentences\ in\ d} \tag{2.7}$$

Accordingly, the probability of two words co-occurring is given by Eq. 2.8.

$$P(c_i, c_j) = \frac{\#\ sentences\ c_i\ and\ c_j\ occur}{\#\ sentences\ in\ d} \tag{2.8}$$

The co-occurrence similarity of the two candidate phrases c_i and c_j is then calculated according to Eq. 2.9.

$$cooccur(c_i, c_j) = PMI(c_i, c_j) \tag{2.9}$$

The final weight of the edge connecting two candidate phrases c_i and c_j is given by Eq. 2.10.

$$sr(c_i, c_j) = semantic(c_i, c_j) \times cooccur(c_i, c_j) \tag{2.10}$$

We then calculate a biased PageRank, similarly to the aforementioned position-biased candidate ranking. $S_{theme}(c_i)$ is the resulting score for a candidate phrase c_i, which is calculated iteratively using:

$$S_{theme} = (1 - \alpha) \cdot \widetilde{t_i} + \alpha \cdot \sum_{c_j \in Adj(c_i)} \frac{sr_{ij}}{O(c_j)} S_{theme}(c_j) \tag{2.11}$$

where $O(c_j) = \sum_{c_k \in Adj(c_j)} sr_{jk}$ and $\widetilde{t_i}$ is the normalized initial score for c_i as calculated during the theme-biased candidate scoring step.

2.5.3 Combination of Position- and Theme-biased Candidate Ranking

After performing a random walk following the two separate approaches described in Sections 2.5.1 and 2.5.2, we are now calculating a ranking score that incorporates both techniques. For that purpose we are introducing a *bias* parameter, that will bias the resulting ranking towards the position-based score over the theme-biased. The final score

for a candidate phrase c_i is computed according to Eq. 2.12.

$$S(c_i) = bias \cdot S_{pos}(c_i) + (1 - bias) \cdot S_{theme}(c_i) \tag{2.12}$$

where $bias$ takes values in the range $[0, 1]$, and the importance of the parameter will be shown experimentally in the next section, amongst other parameters.

After ranking the candidate phrases, KOR returns the top-k ranked keyphrases. In the following chapter, we will present the setting of our experiments that finally lead to the performance evaluation of KOR in Chapter 4.

Chapter 3
Experiments

The KOR model predicts suitable keyphrases for a given document d, ranked by decreasing order of probability. In this chapter, we will describe the datasets used to test the performance of our system, followed by the metrics used to evaluate the predictions on each document in the dataset.

3.1 Datasets

The top-k ranked keyphrases predicted by KOR are evaluated on four publicly available datasets with manually annotated gold standard keyphrases, used frequently for the task of evaluating keyphrases extraction models: *Nguyen*, *KDD*, *WWW* and *Inspec*. The reason for using a variety of datasets is to showcase the performance of KOR on documents with different properties and keyphrase expectations.

The *Nguyen* dataset [30] originally contains 211 scholarly documents, automatically converted to plain text. The topics of the documents cover a variety of disciplines. For our evaluation, we used the title and abstract from each document, similarly to [9]. The gold standard keyphrases are the ones annotated by the authors. We excluded documents that had an empty set of annotated keyphrases, and filtered out keyphrases from the set of gold standard that were not extractive from the document; in general we only focus on extractive keyphrases since our method is solely extractive and it is unfair to test the system on abstractive keyphrases that KOR does not try to generate. Table 3.1 shows that only 53% of the author-input keyphrases are present in the documents. After the filtering, we used a set of 172 documents, averaging at 2.47 keyphrases per document. The Pearson correlation coefficient between the number of words and number of keyphrases in each document is calculated at -0.12 for this dataset.

The *KDD* and *WWW* datasets were derived from the CiteSeerX digital library [31],

containing scientific papers from the ACM Conference and Knowledge Discovery in Data Mining (*KDD*) and the World Wide Web Conference (*WWW*). Both datasets were first introduced by Gollapali and Caragea [14] for the purpose of evaluating systems on the task of keyphrase extraction.

The *KDD* dataset originally contains 834 scholarly documents, and specifically their titles, abstracts and author-assigned keyphrases. As mentioned before, we filter out documents depending on whether the assigned keyphrases are extractive or abstractive. We identified that 48% of the author-input keyphrases are present in the documents, leading us to use 653 documents, with an average of 2.28 keyphrases per document. Measuring the correlation between the length of the document and the amount of keyphrases, we computed $r = 0.18$ for *KDD* dataset.

The *WWW* dataset originally contains 1350 research documents, including their titles, abstracts and author-input keyphrases. The extractive keyphrases account for only 44% of the gold standard keyphrases, and after extracting the documents with only abstractive keyphrases, we gathered a set of 1167 documents, with an average of 2.39 keyphrases per document. The Pearson correlation coefficient was calculated at 0.12 for *WWW*.

The *Inspec* dataset [5] is comprised by 2000 abstracts and titles with their respective keyphrases, divided in sets of 1000, 500 and 500 for the purpose of training, validation and testing respectively. The list of keyphrases accompanying each document is divided in two separate sets: keyphrases assigned by the authors (controlled set) and keyphrases assigned by annotators (uncontrolled set). For our evaluation, we are using the subset of the 500 documents comprising the testing dataset, and we keep a combination of the controlled and uncontrolled set of keyphrases. The final set of documents used in our evaluation after excluding abstractive keyphrases that comprise the 41% of the dataset, included 497 documents. The main difference between this dataset compared to the previous three described in this section is the number of keyphrases accompanying each document. We are now observing an average of 8.30 keyphrases per document, resulting in a correlation coefficient of $r = 0.57$.

Overall, all datasets are comprised by title, abstract and a list of gold standard keyphrases. For the theme-biased candidate scoring and ranking, we are considering the vector representation of the title as the thematic vector of each document, computed based on the SciBERT language model.

A summarization of all the datasets used in our evaluation is shown in Table 3.1.

Dataset	Nguyen	KDD	WWW	Inspec
#Docs	172	653	1167	497
Avg Words	170.62	185.43	156.45	124.02
Avg Unique Words	98.66	107.69	92.60	74.76
Avg Sentences	8.59	10.15	8.62	6.55
r	-0.12	0.18	0.12	0.57
Keyphrases	425	1488	2793	4124
Avg Keyphrases	2.47	2.28	2.39	8.30
% Present	53.19	48.11	43.61	59.21
% Absent	46.81	51.89	56.39	40.79

Table 3.1. General statistics of the processed datasets used in out experiments. r is the Pearson correlation between the length of a document and the number of gold standard keyphrases.

3.2 Evaluation Metrics

In order to evaluate the performance of KOR in each aforementioned dataset, we calculate Precision, Recall and F1-score, which are the most frequently used evaluation metrics for the task of keyphrase extraction [5, 8–10, 13], borrowed from the area of Information Retrieval. We calculate Precision according to Eq. 3.1, while the number of predicted keyphrases is provided as a parameter to test the system's performance when producing top-k keyphrases, for $k = 2, 4, 6, 8$.

$$Precision = \frac{\#\ of\ correct\ predicted\ keyphrases}{\#\ of\ predicted\ keyphrases} \tag{3.1}$$

Similarly, we compute Recall according to Eq. 3.2 and F1-score ($\beta = 1$) according to Eq. 3.3.

$$Recall = \frac{\#\ of\ correct\ predicted\ keyphrases}{\#\ of\ gold\ standard\ keyphrases} \tag{3.2}$$

$$F1 - score = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{3.3}$$

For the purpose of comparing with previous work, we calculate both micro-averaged

Precision, Recall and F1 and macro-averaged, since we have identified several systems that use one or the other to present their performance.

Additionally, we use Mean Reciprocal Rank (MRR) curves to illustrate the performance of KOR against baselines, and when we experimentally test the effect of the parameters of the system. This metric shows the averaged ranking of the first correct prediction, following Eq. 3.4.

$$MRR = \frac{1}{|D|} \sum_{d \in D} \frac{1}{r_d} \tag{3.4}$$

where D is the set of documents used in our experiments and r_d is the rank of the highest ranked predicted keyphrase that falls in the intersection of KOR predictions for d and the gold standard keyphrases accompanying d.

In the next chapter we are presenting the results of applying KOR on the datasets introduced here, using the evaluation metrics we described.

Chapter 4
Results

In this chapter, we are evaluating the performance of KOR on the datasets that we described in Chapter 3. We are first experimentally identifying the best values for the system's parameters, *bias* and window w. Finally, we include a comparison of KOR with other existing state-of-the-art models.

4.1 Varying window

One parameter hard-coded in our model is the size of the window w of successive words that we account for when assigning edges during the construction of both graphs for the calculation of the position-biased and the theme-biased PageRank, to capture the relation between two candidate phrases. We experimented with values of w in $\{2, 4, 6, 8, 10\}$, on all four datasets.

Figure 4.1 shows the MRR curves of KOR for different values of w, separately for each one of the four datasets. As it was illustrated in previous work [9], the size of the window w does not significantly affect the performance of our system. Thus, for the rest of our experiments, we are setting the window size as $w = 10$ which is a frequently used window size for similar systems incorporating PageRank for ranking candidate phrases.

4.2 Varying bias

The bias parameter shifts the focus of the system from the position of a candidate keyphrase to the similarity of the candidate phrase to the theme of the document. A low bias, indicates a strong preference towards the phrase's similarity to the theme, where $bias = 0$ means a complete focus on the thematic similarity. On the other hand, a high

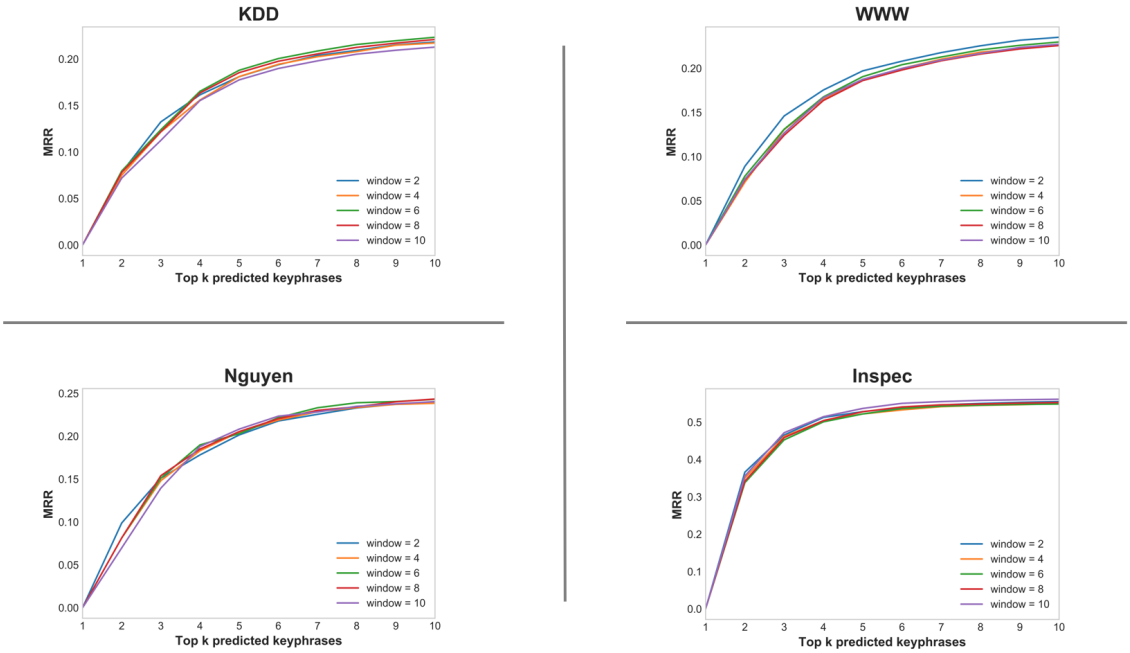

Figure 4.1. MRR curves for the top-k predictions when using different values for the window size w.

bias shows a strong preference towards the importance of a candidate's position, where $bias = 1$ means that the system only takes into account the position of each candidate.

We experimented with different values for the bias, particularly $bias \in \{0, 0.4, 0.6, 1\}$. In this line of testing, we observed significant differences depending on the value of the $bias$. Figure 4.2 illustrates the performance of KOR in terms of MRR given different values of $bias$ against all four datasets.

In order to illustrate our results better, we are additionally including Figures 4.3, 4.4 and 4.5 that showcase the curves for KOR in terms of macro-averaged Precision, Recall and F1-score respectively. The curves for the micro-averaged metrics are similar, and thus not included here.

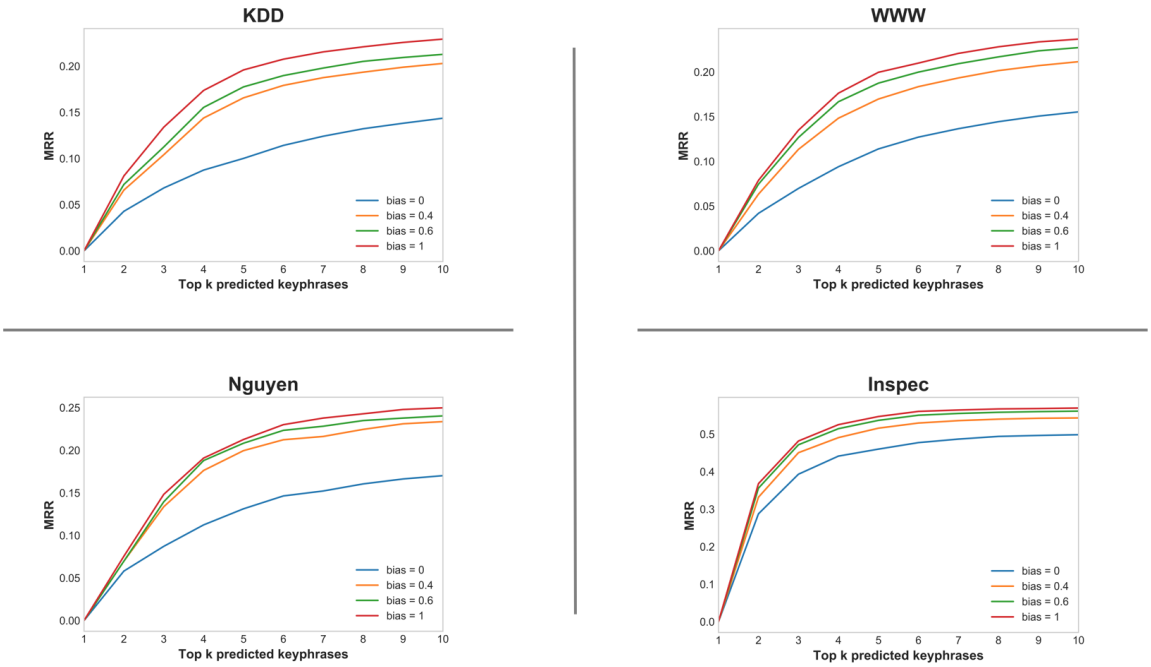

Figure 4.2. MRR curves for the top-k predictions when using different values for bias.

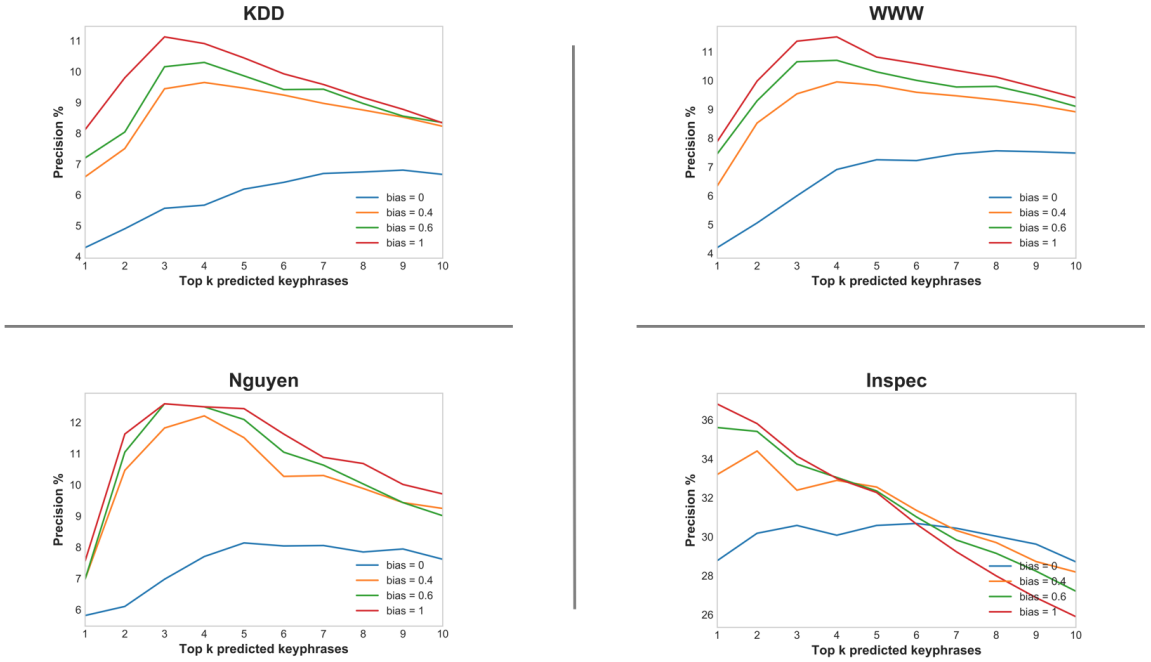

Figure 4.3. Precision curves for the top-k predictions when using different values for bias.

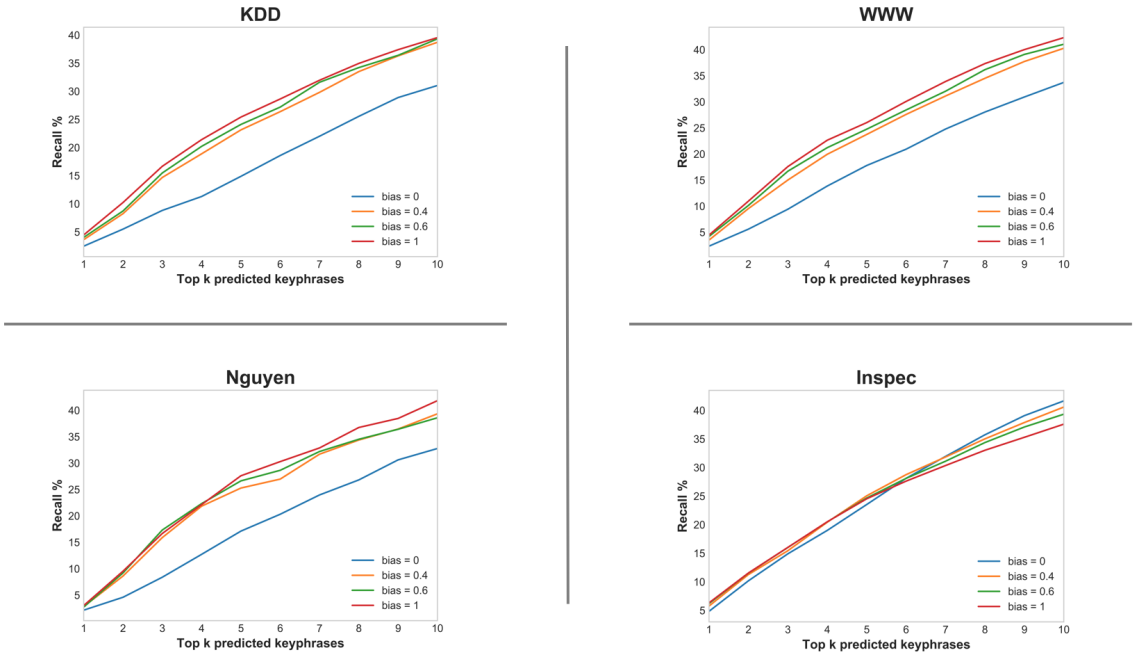

Figure 4.4. Recall curves for the top-k predictions when using different values for bias.

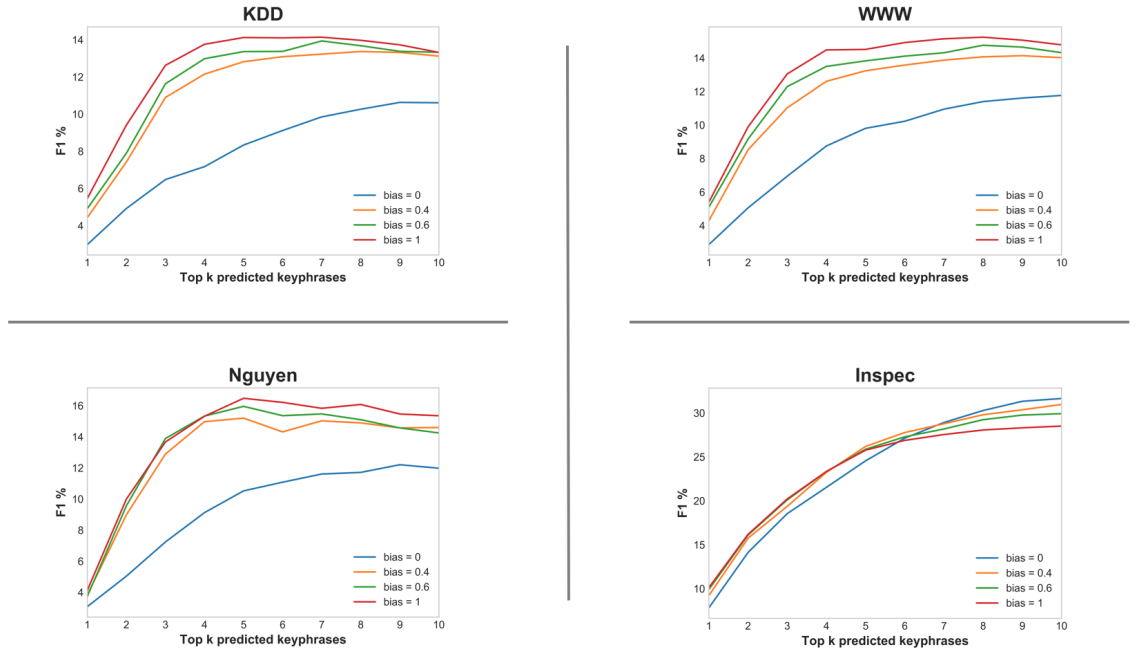

Figure 4.5. F1-score ($\beta = 1$) curves for the top-k predictions when using different values for bias.

4.3 Testing against baselines

Our system aimed to incorporate a position-biased and a theme-biased PageRank ranking model, in order to extract keyphrases more effectively compared to other state-of-the-art methods. To that regard, we run experiments on all four presented datasets for KOR with $w = 10$ and $bias = 0.6$.

We are presenting our results in terms of MRR, and Precision, Recall and F1-score when applying KOR and three other baseline methods, specifically TextRank [8], PositionRank [9] and SingleRank [13], on the four datasets by performing the evaluation we presented previously, where we are focusing solely on the extractive keyphrases. We are also introducing a RandomChoice method, which picks randomly the top-k predicted keyphrases after following the same procedure as KOR to extract candidate phrases. Figure 4.6 shows the MRR curves of all models for the four datasets. As suggested by the curves, KOR outperforms all baselines, including PositionRank that we based our position-biased methodology on, signifying that our proposed model improved on the previous work.

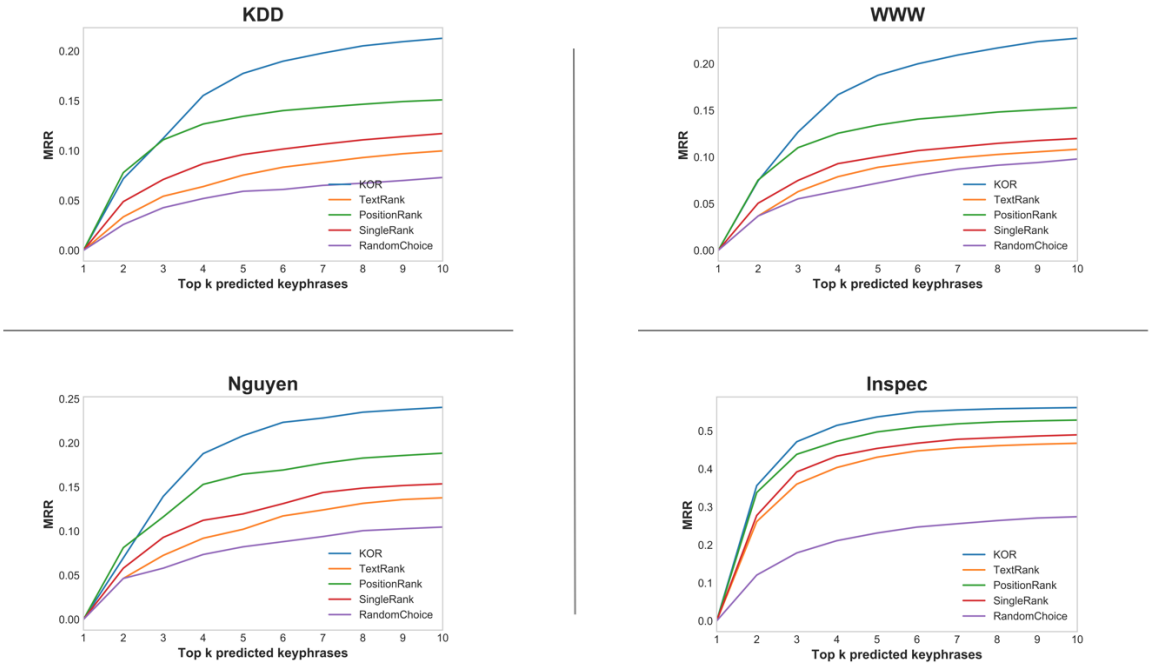

Figure 4.6. MRR curves for KOR and baselines on the four datasets.

Table 4.1 shows the performance of all methods on all four datasets for the top-k predicted keyphrases in terms of micro-averaged Precision, Recall and F1-score. The

macro-averaged Precision, Recall and F1-score results are shown in Table 4.2. KOR outperforms significantly the state-of-the-art baselines, for all evaluation metrics computed here.

Dataset	Method	Top-2			Top-4			Top-6			Top-8		
		P%	R%	F1%	P%	R%	F1%	P%	R%	F1%	P%	R%	F1%
KDD	KOR	**52.5**	**46.05**	**49.07**	**27.79**	**45.29**	**34.44**	**19.59**	**47.13**	**27.67**	**15.83**	**51.38**	**24.2**
	TextRank	53.06	49.06	50.98	29.87	47.06	36.54	20.27	44.96	27.94	15.13	44.01	22.52
	PositionRank	51.6	43.3	47.09	28.42	44.32	34.63	20.33	45.1	28.03	15.69	45.81	23.37
	SingleRank	54.1	46.48	50	28.86	44.79	35.1	20.02	44.76	27.67	15.47	44.98	23.02
	RandomChoice	51.28	39.6	44.69	27.3	40.49	32.61	18.71	43.31	26.13	14.67	44.51	22.07
WWW	KOR	**51.91**	**44.93**	**48.17**	**27.91**	**44.34**	**34.25**	**19.84**	**46.35**	**27.79**	**15.93**	**49.2**	**24.07**
	TextRank	50.96	44.17	47.32	28.11	44.89	34.57	20.01	45.16	27.74	15.4	45.65	23.03
	PositionRank	52.96	47.48	50.07	28.56	45.29	35.03	20.06	46.12	27.96	15.47	45.48	23.09
	SingleRank	51.72	43.8	47.43	28.71	44.61	34.94	20.16	44.99	27.85	15.44	45.5	23.06
	RandomChoice	52.33	37.34	43.58	27.23	38.8	32	18.61	38.79	25.15	14.59	40.9	21.5
Nguyen	KOR	**52.78**	**38**	**44.19**	**28.67**	**44.33**	**34.82**	**20.43**	**46.72**	**28.43**	**16.43**	**50.36**	**24.78**
	TextRank	55.88	39.58	46.34	31.62	44.79	37.07	20.68	46.21	28.57	15.43	44.74	22.94
	PositionRank	51.92	36.49	42.86	30.19	45.71	36.36	20.51	44.94	28.17	15.98	46.86	23.83
	SingleRank	54.55	37.5	44.44	31.76	44.76	37.15	20.83	45.73	28.63	15.74	45.36	23.37
	RandomChoice	50	32.43	39.34	27.88	35.8	31.35	19.82	41.12	26.75	15.1	41.13	22.1
Inspec	KOR	**60.27**	**14.02**	**22.75**	**41.06**	**19.37**	**26.32**	**34.41**	**24.27**	**28.46**	**31.09**	**29.45**	**30.25**
	TextRank	60.31	14.39	23.24	40.92	19.41	26.33	35.11	24.9	29.14	31.47	29.56	30.49
	PositionRank	60.82	14.14	22.95	41.24	19.33	26.32	34.94	24.94	29.1	31.34	29.5	30.39
	SingleRank	59.72	14.4	23.2	40.68	19.28	26.16	35.08	24.88	29.11	31.14	29.42	30.26
	RandomChoice	53.81	11.67	19.18	30.46	13.35	18.57	22.82	14.86	18	19.07	17.03	17.99

Table 4.1. KOR against TextRank, PositionRank and SingleRank as baselines, along with a RandomChoice implementation algorithm. The results are shown in terms of Precision, Recall and F1-score (micro-averaged). Best results are indicated by **bold blue.**

Dataset	Method	Top-2			Top-4			Top-6			Top-8		
		P%	R%	F1%	P%	R%	F1%	P%	R%	F1%	P%	R%	F1%
KDD	KOR	**8.04**	**8.73**	**7.9**	**10.3**	**20.22**	**12.98**	**9.42**	**27.2**	**13.38**	**8.96**	**34.23**	**13.68**
	TextRank	3.98	4.78	4.09	4.66	8.68	5.74	4.82	11.89	6.51	4.81	15.03	6.9
	PositionRank	7.43	7.83	7.18	6.5	12.22	8.05	5.89	14.87	8.01	5.45	17.22	7.85
	SingleRank	5.05	5.45	4.92	5.31	9.74	6.52	5.08	12.75	6.89	5.03	15.83	7.22
	RandomChoice	3.06	2.83	2.79	3.18	5.91	3.91	2.81	8.08	3.97	2.76	10.25	4.18
WWW	KOR	**9.3**	**10.15**	**9.16**	**10.7**	**21.23**	**13.51**	**10.01**	**28.46**	**14.12**	**9.8**	**36.19**	**14.77**
	TextRank	4.54	4.88	4.46	5.14	9.67	6.39	5.01	12.89	6.82	4.81	15.54	6.91
	PositionRank	7.67	8.35	7.58	6.61	12.41	8.22	5.95	15.6	8.15	5.55	18.07	8.01
	SingleRank	5.14	5.32	4.97	5.39	10.04	6.66	5.16	13.08	7.01	4.97	15.97	7.15
	RandomChoice	3.86	3.51	3.45	3.66	6.57	4.43	4.05	10.49	5.54	4.02	13.57	5.92
Nguyen	KOR	**11.05**	**9.2**	**9.57**	**12.5**	**22.34**	**15.33**	**11.05**	**28.64**	**15.36**	**10.03**	**34.53**	**15.1**
	TextRank	5.52	4.85	4.78	6.25	9.74	7.28	6.49	15.6	8.86	6.19	19.49	9.1
	PositionRank	7.85	6.65	6.78	9.3	15.36	11.15	7.75	18.5	10.57	7.06	22.9	10.44
	SingleRank	6.98	5.58	5.87	6.83	10.66	7.97	7.27	17.34	9.9	6.41	19.87	9.42
	RandomChoice	3.49	2.85	2.96	4.22	5.97	4.76	4.26	9.86	5.75	4.22	12.94	6.15
Inspec	KOR	**35.41**	**11.53**	**16.11**	**33.05**	**20.5**	**23.35**	**31.02**	**28.11**	**27.29**	**29.15**	**34.38**	**29.24**
	TextRank	27.67	9.15	12.8	28.57	17.93	20.4	29.18	26.22	25.56	28.33	32.56	28.18
	PositionRank	32.8	10.76	15.02	30.53	19.08	21.67	29.88	26.83	26.14	28.66	32.87	28.46
	SingleRank	30.28	9.97	14.03	28.97	17.69	20.35	29.58	26.37	25.86	28.28	32.43	28.09
	RandomChoice	12.78	3.73	5.45	12.63	7.56	8.71	12.44	10.67	10.59	12.55	14.66	12.49

Table 4.2. KOR against TextRank, PositionRank and SingleRank as baselines, along with a RandomChoice implementation algorithm. The results are shown in terms of Precision, Recall and F1-score (macro-averaged). Best results are indicated by **bold blue.**

4.3.1 Discussion

A very interesting observation that we made from experimentally testing our system for different values of the *bias* parameter, is that the position-biased scoring and ranking results in better keyphrase extraction performance than the combination of the two methods. For the *Inspec* dataset, we are observing some increase in performance in terms of Precision, Recall and F1-score for the top-k predicted keyphrases, for $k > 7$. As we will discuss in more detail in the next chapter, this is a topic that we would like to explore further in the future, in order to more effectively incorporate a language model in our system.

The goal of this work was to incorporate both position- and theme-biased PageRank to improve on existing models [9,10]. Therefore, the experiments comparing our proposed model to state-of-the-art are not considering bias extremes, instead we are using $bias \in (0, 1)$, although our model performs better for $bias = 1$.

As mentioned briefly in Section 4.3, KOR outperforms all state-of-the-art models when tested on the exact same datasets, focusing on extractive keyphrases. It is important to note that all systems included in our comparison perform keyphrase extraction and thus the results introduced here for these models show an improvement on the same metrics than those presented in the respective papers, an improvement arising by not including the abstractive keyphrases in our evaluation.

Overall, we can confidently say that the candidate selection process introduced by KOR has resulted in a significant performance boost.

Chapter 5
Conclusion and Future Implications

We introduced a novel unsupervised graph-based methodology for the task of keyphrase extraction, named KOR. Building on previous work that incorporated various techniques in collaboration with the application of the PageRank algorithm to perform a random walk and rank candidate keyphrases, we combined two different approaches, improving on each one experimentally, and achieving a better performance.

We incorporated the position and frequency of each word, in accordance with the semantic similarity of words with each other, but also with a main theme dominating a document, in order to calculate scores from two separate biased PageRank implementations. The results are promising, outperforming state-of-the-art methods on benchmark datasets and building the foundation for a possible breakthrough in the area of keyphrase extraction.

In the future, we are planning to use different language models that can improve the performance of our model for the cases where we are lowering the bias. It will be interesting to introduce a language model that is trained in a more diverse set of topics of scholarly documents. Additionally, we would like to test our system against datasets where the full body of the scientific paper is included and improve our approach on lengthier documents.

Bibliography

[1] Papagiannopoulou, E. and G. Tsoumakas (2019) "A Review of Keyphrase Extraction." *CoRR*, **abs/1905.05044**.
URL http://dblp.uni-trier.de/db/journals/corr/corr1905.html#abs-1905-05044

[2] Turney, P. D. (2000) "Learning Algorithms for Keyphrase Extraction," *Information Retrieval*, **2**(4), pp. 303–336.
URL https://doi.org/10.1023/A:1009976227802

[3] Hasan, K. S. and V. Ng (2014) "Automatic Keyphrase Extraction: A Survey of the State of the Art," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Association for Computational Linguistics, Baltimore, Maryland, pp. 1262–1273.
URL http://www.aclweb.org/anthology/P14-1119

[4] Meng, R., S. Zhao, S. Han, D. He, P. Brusilovsky, and Y. Chi (2017) "Deep Keyphrase Generation." in *ACL (1)* (R. Barzilay and M.-Y. Kan, eds.), Association for Computational Linguistics, pp. 582–592.
URL http://dblp.uni-trier.de/db/conf/acl/acl2017-1.html#MengZHHBC17

[5] Hulth, A. (2003) "Improved automatic keyword extraction given more linguistic knowledge," *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 03)*, pp. 216–223.

[6] Medelyan, O., E. Frank, and I. H. Witten (2009) "Human-competitive tagging using automatic keyphrase extraction," in *Internat. Conference of Empirical Methods in Natural Language Processing, EMNLP-2009,*.
URL http://www.cs.waikato.ac.nz/\~{}ihw/papers/09-OM-EF-IHW-Humancompetitive\%20tag.pdf

[7] tau Yih, W., J. Goodman, and V. R. Carvalho (2006) "Finding advertising keywords on web pages," in *WWW '06: Proceedings of the 15th international conference on World Wide Web*, ACM, New York, NY, USA, pp. 213–222.
URL http://portal.acm.org/citation.cfm?id=1135777.1135813&coll=Portal&dl=ACM&CFID=6067738&CFTOKEN=82120793

[8] Mihalcea, R. and P. Tarau (2004) "TextRank: Bringing Order into Texts," in *Proceedings of EMNLP-04and the 2004 Conference on Empirical Methods in Natural Language Processing*.

[9] Florescu, C. and C. Caragea (2017) "PositionRank: An Unsupervised Approach to Keyphrase Extraction from Scholarly Documents." in *ACL (1)* (R. Barzilay and M.-Y. Kan, eds.), Association for Computational Linguistics, pp. 1105–1115.
URL http://dblp.uni-trier.de/db/conf/acl/acl2017-1.html#FlorescuC17

[10] Mahata, D., J. Kuriakose, R. R. Shah, and R. Zimmermann (2018) "Key2Vec: Automatic Ranked Keyphrase Extraction from Scientific Articles using Phrase Embeddings." in *NAACL-HLT (2)* (M. A. Walker, H. Ji, and A. Stent, eds.), Association for Computational Linguistics, pp. 634–639.
URL http://dblp.uni-trier.de/db/conf/naacl/naacl2018-2.html#MahataKSZ18

[11] Liu, Z., P. Li, Y. Zheng, and M. Sun (2009) "Clustering to Find Exemplar Terms for Keyphrase Extraction." in *EMNLP*, ACL, pp. 257–266.
URL http://dblp.uni-trier.de/db/conf/emnlp/emnlp2009.html#LiuLZS09

[12] Zha, H. (2002) "Generic summarization and keyphrase extraction using mutual reinforcement principle and sentence clustering." in *SIGIR* (K. Järvelin, M. Beaulieu, R. A. Baeza-Yates, and S.-H. Myaeng, eds.), ACM, pp. 113–120.
URL http://dblp.uni-trier.de/db/conf/sigir/sigir2002.html#Zha02

[13] Wan, X. and J. Xiao (2008) "Single Document Keyphrase Extraction Using Neighborhood Knowledge," in *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 2*, AAAI'08, AAAI Press, pp. 855–860.
URL http://dl.acm.org/citation.cfm?id=1620163.1620205

[14] Gollapalli, S. D. and C. Caragea (2014) "Extracting Keyphrases from Research Papers Using Citation Networks," in *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, AAAI Press, pp. 1629–1635.
URL http://dl.acm.org/citation.cfm?id=2892753.2892779

[15] Grineva, M., M. Grinev, and D. Lizorkin (2009) "Extracting Key Terms from Noisy and Multitheme Documents," in *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, ACM, New York, NY, USA, pp. 661–670.
URL http://doi.acm.org/10.1145/1526709.1526798

[16] Brin, S. and L. Page (1998) "The Anatomy of a Large-scale Hypertextual Web Search Engine," *Comput. Netw. ISDN Syst.*, **30**(1-7), pp. 107–117.
URL http://dx.doi.org/10.1016/S0169-7552(98)00110-X

[17] Herings, P. J.-J., G. van der Laan, and D. Talman (2001) *Measuring the Power of Nodes in Digraphs*, *Tinbergen Institute Discussion Papers 01-096/1*,

Tinbergen Institute.
URL https://ideas.repec.org/p/tin/wpaper/20010096.html

[18] Kleinberg, J. M. (1999) "Authoritative Sources in a Hyperlinked Environment," *J. ACM*, **46**(5), pp. 604–632.
URL http://doi.acm.org/10.1145/324133.324140

[19] Mikolov, T., I. Sutskever, K. Chen, G. S. Corrado, and J. Dean (2013) "Distributed Representations of Words and Phrases and their Compositionality," in *Advances in Neural Information Processing Systems 26* (C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, eds.), Curran Associates, Inc., pp. 3111–3119.
URL http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-p
pdf

[20] Pennington, J., R. Socher, and C. Manning (2014) "Glove: Global Vectors for Word Representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Association for Computational Linguistics, Doha, Qatar, pp. 1532–1543.
URL https://www.aclweb.org/anthology/D14-1162

[21] Bojanowski, P., E. Grave, A. Joulin, and T. Mikolov (2016) "Enriching Word Vectors with Subword Information," *CoRR*, **abs/1607.04606**, 1607.04606.
URL http://arxiv.org/abs/1607.04606

[22] Devlin, J., M. Chang, K. Lee, and K. Toutanova (2018) "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *CoRR*, **abs/1810.04805**, 1810.04805.
URL http://arxiv.org/abs/1810.04805

[23] Beltagy, I., A. Cohan, and K. Lo (2019) "SciBERT: Pretrained Contextualized Embeddings for Scientific Text," *CoRR*, **abs/1903.10676**, 1903.10676.
URL http://arxiv.org/abs/1903.10676

[24] Lee, J., W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So, and J. Kang (2019) "BioBERT: a pre-trained biomedical language representation model for biomedical text mining," *Bioinformatics*.
URL https://doi.org/10.1093/bioinformatics/btz682

[25] Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin (2017) "Attention is All You Need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, Curran Associates Inc., USA, pp. 6000–6010.
URL http://dl.acm.org/citation.cfm?id=3295222.3295349

[26] Sahrawat, D., D. Mahata, M. Kulkarni, H. Zhang, R. Gosangi, A. Stent, A. Sharma, Y. Kumar, R. R. Shah, and R. Zimmermann (2019), "Keyphrase

Extraction from Scholarly Articles as Sequence Labeling using Contextualized Embeddings," 1910.08840.

[27] Wu, Y., M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, Łukasz Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean (2016) "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation," *CoRR*, **abs/1609.08144**.
URL http://arxiv.org/abs/1609.08144

[28] Yang, Z., Z. Dai, R. Salakhutdinov, and W. W. Cohen (2017) "Breaking the Softmax Bottleneck: A High-Rank RNN Language Model," *CoRR*, **abs/1711.03953**, 1711.03953.
URL http://arxiv.org/abs/1711.03953

[29] Church, K. W. and P. Hanks (1989) "Word Association Norms, Mutual Information, and Lexicography," in *27th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, Vancouver, British Columbia, Canada, pp. 76–83.
URL https://www.aclweb.org/anthology/P89-1010

[30] Nguyen, T. D. and M.-Y. Kan (2007) "Keyphrase Extraction in Scientific Publications," in *Proceedings of the 10th International Conference on Asian Digital Libraries: Looking Back 10 Years and Forging New Frontiers*, ICADL'07, Springer-Verlag, Berlin, Heidelberg, pp. 317–326.
URL http://dl.acm.org/citation.cfm?id=1780653.1780707

[31] Giles, C. L., K. D. Bollacker, and S. Lawrence (1998) "CiteSeer: An Automatic Citation Indexing System," in *Proceedings of the Third ACM Conference on Digital Libraries*, DL '98, ACM, New York, NY, USA, pp. 89–98.
URL http://doi.acm.org/10.1145/276675.276685