



Article

# Unsupervised Keyphrase Extraction for Web Pages

Tim Haarman <sup>1,2,\*</sup>, Bastiaan Zijlema <sup>2</sup> and Marco Wiering <sup>1,\*</sup>

<sup>1</sup> Bernoulli Institute, Department of Artificial Intelligence, University of Groningen, PO Box 407, 9700 AK Groningen, The Netherlands

<sup>2</sup> Dataprovider.com, Helperpark 292, 9723 ZA Groningen, The Netherlands

\* Correspondence: t.r.haarman@gmail.com (T.H.); m.a.wiering@rug.nl (M.W.)

Received: 28 June 2019; Accepted: 26 July 2019; Published: 31 July 2019



**Abstract:** Keyphrase extraction is an important part of natural language processing (NLP) research, although little research is done in the domain of web pages. The World Wide Web contains billions of pages that are potentially interesting for various NLP tasks, yet it remains largely untouched in scientific research. Current research is often only applied to clean corpora such as abstracts and articles from academic journals or sets of scraped texts from a single domain. However, textual data from web pages differ from normal text documents, as it is structured using HTML elements and often consists of many small fragments. These elements are furthermore used in a highly inconsistent manner and are likely to contain noise. We evaluated the keyphrases extracted by several state-of-the-art extraction methods and found that they did not transfer well to web pages. We therefore propose WebEmbedRank, an adaptation of a recently proposed extraction method that can make use of structural information in web pages in a robust manner. We compared this novel method to other baselines and state-of-the-art methods using a manually annotated dataset and found that WebEmbedRank achieved significant improvements over existing extraction methods on web pages.

**Keywords:** unsupervised keyphrase extraction; sequence embeddings; web pages; WebEmbedRank

## 1. Introduction

Keyphrases are short phrases of one or more words that exemplify a particular document. Automatically generating fitting keyphrases for documents is an important task, as it can be used to index and categorize large corpora of documents. This can in turn enable various applications, such as fast and accurate searching through large collections of documents. The World Wide Web is one of the domains wherein effective keyphrase generation could be highly relevant, as its size and inconsistent nature makes accurate indexation difficult. It is also one of the largest sources of publicly available textual data, with estimates of the size of the indexed web ranging up to 50 billion pages [1]. Despite its size, sets of web pages are infrequently used in natural language processing (NLP) tasks such as keyphrase extraction.

One important factor that differentiates web pages from more commonly used sources like scientific articles or news reports is the presence of structural information in the form of HTML elements. The extra information that these elements provide can be used to determine the most relevant keyphrases on the page. For instance, phrases found in the title of a web page are intuitively more likely to be relevant than phrases taken from a paragraph at the bottom of a page. An initial method to retrieve useful terms could therefore be to extract phrases from the title. Although this seems like an effective strategy, it has been found that more than one third of web pages have a title that is not relevant [2]. This includes titles that contain a placeholder, irrelevant or generic terms, or are empty or not even specified to begin with. Making assumptions based on the positional information alone can therefore lead to poor keyphrase extraction, as these data are not consistently available or reliable. This illustrates the difficulty that the inconsistency of web pages poses for keyphrase extraction.

To avoid this inconsistency, most research on keyphrase extraction is applied to clean documents with a predetermined structure, such as journal papers and abstracts [3–5]. Research that does use web content mostly uses data from specific web pages with a known structure for each page, like news articles scraped from a single domain [6,7]. This is not representative for the web as a whole, and it is questionable how well the methods from these studies transfer to a noisy and inconsistent domain like that of the World Wide Web.

The majority of the state-of-the-art methods for unsupervised keyphrase extraction are graph-based ranking methods [8]. These methods formalize a document as a graph, often representing the words as the vertices with edges based on the co-occurrence frequencies between words. The vertices are ranked using a graph-based ranking algorithm like PageRank [9]. Phrases of multiple words are generally reconstructed after the ranking process. While graph-based methods make use of statistical information, such as the word co-occurrence frequencies, the semantics of words are not explicitly taken into account. Recently, a new keyphrase extraction method called EmbedRank was proposed that moved away from graph-based methods [5]. EmbedRank achieved state-of-the-art results on multiple benchmarks using pre-trained text embeddings to find relevant candidates. The usage of text embeddings allowed EmbedRank to rank candidate phrases based on their semantic similarity to the document, instead of their relation to other words in the document alone.

In this paper, we propose WebEmbedRank, a weighted adaptation of EmbedRank suited for keyphrase extraction from web pages. By incorporating a bias towards more important hypertext elements in the EmbedRank implementation, WebEmbedRank utilizes the structural information present in hypertext documents while remaining robust against noise. Furthermore, we investigated how well several state-of-the-art keyphrase extraction methods transferred to web documents from multiple sources, and found that most methods performed poorly when web pages are treated like standard text documents.

The paper is structured as follows. In Section 2, we give a brief overview of the field of unsupervised keyphrase extraction and web-related NLP tasks. The WebEmbedRank method is described in Section 3, followed by an explanation of how the method was implemented and evaluated in Section 4. We present our results and findings in Section 5 and conduct a final discussion in Section 6.

## 2. Related Work

In general, keyphrase extraction can be divided into supervised and unsupervised methods. The supervised field usually treats keyphrase extraction as a binary classification problem, where a set of candidate phrases are classified as either a keyphrase or a non-keyphrase [8]. An example of this is the research by Yih et al., who proposed a supervised approach to extract keyphrases from web pages [10]. Their system classified keyphrases based on several features in order to show more relevant advertisements to the user, using a manually annotated dataset as training data. They found multiple features that contributed to the classification, such as the location of a phrase in the document, statistical features based on the term and document frequencies, and query log information. Though a supervised approach can be effective, it requires a large set of annotated data, which is very laborious to construct and as far as we know there is no suitable large dataset for this task openly available. Additionally, the World Wide Web spans many different languages. Extending a supervised system to another language often requires re-annotating and retraining new classifiers for each language. Considering this, we focused on unsupervised extraction methods that can extract keyphrases based solely on the information available on the web page.

### 2.1. Hypertext Documents

Although previous research on unsupervised keyphrase extraction from web pages is limited, it has repeatedly been shown that the HTML element in which a word or phrase appears correlates with its relevance for the topic of a page [11–13]. Thomaidou and Vazirgiannis proposed an unsupervised keyphrase recommendation system for web pages that used hypertext location information [14].

Their extraction method calculated a relevance score for each phrase, which was based on word frequency information and an additional weighting scheme considering the element it was extracted from. They based the weighting factors on the locations web page designers place the most important information on websites, although they provide no further justification for the exact values they chose.

This knowledge has furthermore been used in other NLP tasks using hypertext documents, such as classification. Riboni introduced a method they referred to as the Structure-oriented Weighting Technique (SWT) for web page classification [15]. The idea behind this weighting was to assign higher weights to texts in elements that were more suitable to represent a web page, such as the title and meta description. Their results showed higher classification performance when the extra weights were applied, indicating that the title and meta description may contain more important terms than the rest of the page. Similar to this, Kwon and Lee divided elements into three groups of varying significance [16]. The highest weighted group contained elements such as the title, meta description and headings. These weighting factors were used in combination with a  $k$ -nearest neighbor classifier to effectively assign web pages to several categories.

This notion that the relevance of phrases may be tied to their location in the hypertext can be used in the selection of keyphrases. By preferring phrases from elements that have a tendency to be more descriptive for the content on a web page, we can potentially improve the quality of the selected keyphrases.

## 2.2. Graph-Based Ranking

Most state-of-the-art unsupervised keyphrase extraction methods are graph based, many of which are based on the PageRank algorithm [9] that was originally used by Google to rank websites. In this ranking system, a directional graph is created wherein web pages are represented as vertices and the connections are determined by the in and outgoing links. The PageRank algorithm is then iteratively applied to rank each vertex. This method was first adapted to be used for keyphrase extraction by Mihalcea and Tarau, who proposed the TextRank method [4]. In TextRank, an undirected graph is constructed to represent documents where the vertices are all the unique words. The edges are determined by passing a window of a fixed size over the text, creating undirected edges between the vertices of all words that occur in the same window. A weighted version of the PageRank algorithm is used to rank each vertex, after which the words are selected from the highest scoring vertices and reassembled into phrases by combining high-scoring adjacent words.

Several methods have since been proposed that built and improved on the TextRank method. Wan and Xiao proposed Singlerank [17], which is a slight adaptation of the original TextRank. It weights the edges based on the co-occurrence frequency of the words that correspond to their vertices. This ensures that words that occur together more frequently in the document are given a higher weight. Additionally, they proposed to construct phrases by filtering for noun phrases, and scoring the phrases by summing the scores of individual words to obtain a score for the phrase. While this method showed increased scores over TextRank, Bougouin et al. suggested that there is no justification for this scoring method [6]. They additionally argued that summing the individual scores to obtain phrase scores leads to longer phrases getting higher scores, even when some of the words in the phrase are irrelevant. To solve this, Bougouin et al. proposed TopicRank. In TopicRank, phrases are first clustered into topics based on their similarity. The authors suggested placing two phrases into the same cluster if at least 25% of their words overlap. TextRank's ranking model is then applied to a complete graph where the vertices are the topics, in order to assign a score to each cluster. The keyphrases are selected by taking only the most representative candidate from each topic cluster, starting with the cluster that got the highest score. This clustering attempts to avoid overgeneration of similar keywords, however it can not effectively deal with synonyms that are semantically similar, but not syntactically.

Recently, Boudin proposed a variation of TopicRank using multipartite graphs [7], which is commonly referred to as MultipartiteRank. Arguing that a downside of TopicRank is that all phrases

in a single topic are seen as equally important, they proposed to model the topics in a multipartite graph where the vertices are the words and the partitions the topics. This modification allowed the graph to model the relations between all words while still being able to represent the different clusters. They also added an additional weighting step, where the edges between words that first occurred early on in the text were given a higher weight. The idea of including a weighting procedure to increase the influence of earlier words was not new, and was earlier used by Florescu and Caragea in the PositionRank method [3]. PositionRank introduced a positional bias to TextRank's ranking procedure to favor words at the start of the document and showed that this significantly improved the quality of the extracted keyphrases.

### 2.3. Embedding-Based Ranking

Ever since the introduction of Word2Vec in 2013 [18], word embeddings have been adopted on a large scale in a variety of NLP tasks. Before the popularity of word embeddings, words were often represented as large one-hot encoded vectors. In this encoding method words are represented as vectors with a separate dimension for each unique word, where only the dimension that corresponds to the word is set to 1. With large corpora of text, this can result in very high-dimensional and sparse word representations. Word embedding methods map these one-hot encoded vectors to a low-dimensional continuous vector space. Most recent word embedding techniques are based on neural networks that are trained to perform language modelling tasks [19]. In such tasks, a model is trained to predict the words that surround a given target word. Because of this, the model has to learn how context words affect the meaning of the sentence. The resulting vector space model containing the word embeddings can therefore represent the relations and semantic similarities between words. Since the introduction of Word2Vec, various other embedding methods have been introduced, including methods to create embeddings for sequences of texts like Doc2Vec [20] and Sent2Vec [21].

With the increase of text embedding techniques, new possibilities also emerged for keyphrase extraction. Using the semantic representation of text in the form of vectors can be very helpful in determining whether a word is relevant to a document. Recently, Bennani-Smires et al. proposed a new method called EmbedRank that improved on the state-of-the-art of unsupervised keyphrase extraction using word sequence embeddings [5]. They introduced a relatively simple yet effective algorithm that extracts phrases based only on the document itself and pre-trained sequence embeddings, requiring no corpus. Similar to most other methods, EmbedRank starts with selecting candidate phrases. After tokenizing all text in the document, part-of-speech (POS) tags are assigned to each token. Only phrases that consist of one or more nouns and any adjectives that directly precede them are considered candidates. The candidates are then ranked based on their similarity with the document as a whole. To find this similarity, sequence embeddings of all the candidates and one of the complete document are made. Bennani-Smires et al. compared the aforementioned Doc2Vec and Sent2Vec to create these embeddings, and found that overall Sent2Vec had a higher accuracy while also being faster. The similarity between each candidate embedding and the general document embedding is then calculated using the cosine similarity between the candidate and document embedding vectors. The final keyphrases are determined by selecting the candidates with the highest similarity to the document as a whole. This ensures that keyphrases were selected based on the meaning of a phrase, rather than term frequency or context related statistics from the document alone.

The authors of EmbedRank furthermore proposed a second version they named EmbedRank++, which adds a diversity mechanism to decrease the amount of semantically equivalent keyphrases being chosen. This diversity mechanism is an adapted version of the Maximal Marginal Relevance (MMR) metric, which weighs diversity against similarity. It does this by taking into account the cosine similarities between all the candidate embeddings. A parameter  $\lambda$  is used to determine the balance between the relevance (similarity with the document) and the diversity (similarity with other candidates). This parameter can range between 0 and 1, where a higher value indicates an increased importance of the relevance versus the diversity, and vice versa. With a balanced value for  $\lambda$ , candidates

that are too similar to other candidates but not as relevant are excluded. Although this version scored lower on their evaluation set, they also showed the results of a pilot user study that suggested that users overall preferred the sets of phrases with diversity.

### 3. WebEmbedRank

While the EmbedRank method has been shown to be capable of extracting state-of-the-art keyphrases, it does not use the positional information from the text. In the original proposal, the method is implemented for extracting keyphrases from scientific documents and news articles. As documents from these domains consist of a single segment of continuous text, there is limited structural information available. When web pages are treated in the same way, valuable structural information from the hypertext is ignored. To effectively make use of this information, we propose WebEmbedRank, an extension to the EmbedRank algorithm that adds an extra weighting scheme to allow it to take advantage of the extra structural information present in web pages.

WebEmbedRank works similar to EmbedRank, in that this method calculates the cosine distance between a document embedding and the candidate embeddings. Both the candidate and document embeddings were calculated using Sent2vec [21], as Bennani-Smires et al. have shown that it generally worked better and faster than Doc2vec [5]. We slightly altered the way the document embedding was calculated to deal with the highly varying length of text on web pages. Instead of basing the document embedding on all candidates, which can sometimes be scarce, we based it on all text on the page after stop-word removal. The stop-words were removed, as these yield no semantic value and therefore only add noise to the embedding. While lexical items other than adjectives and nouns may generally not make the best keyphrases, they can contribute to the overall meaning of the page, and we found that using all text improved the performance of WebEmbedRank on our evaluation set.

As the resulting cosine similarity scores can be negative, the similarity scores for all candidates are normalized between 0 and 1 before the weighting. The bonus weight is determined based on the elements a phrase occurs in. As a phrase that appears in multiple key elements is likely to be more important than one that only occurs in one, we propose a cumulative weighting system. This entails that, for each candidate phrase, the bonus weight is determined as the sum of all weights of the elements the phrase occurs in. Earlier work has shown that the hostname can be a strong descriptor for a web page [22,23], and often consists of the most relevant words for the page such as the brand name, or the main product or service it provides. We therefore included a weight to increase the score for terms that occurred in the hostname. We also included a small penalty for long phrases. As the document embedding is essentially a mean of the word vectors in the documents, a phrase embedding has a tendency to move more towards the mean of all word vectors as the number of words in the phrase increases. For this reason, we noticed that EmbedRank had a tendency to generate many long phrases, which may also have been a cause for the lack of diversity described in the EmbedRank paper [5]. By adding a slight penalty to long phrases, we bias the ranking by preferring shorter phrases, while still allowing longer phrases to be selected if the added words significantly contribute to the relevance of the phrase. The proposed weights were determined based on an exhaustive grid search on our manually annotated dataset, and can be found in Table 1. To illustrate how this bonus weight is calculated, consider a phrase that consists of two words and appears in the title and the description, but not in hostname or headings. In this case, the bonus weight is  $1.5 + 0.5 - 0.25 \times (2 - 1) = 1.75$ . The scoring  $S(p_i)$  is calculated as follows, where  $\text{sim}(p_i, d)$  is the cosine similarity between a candidate phrase embedding  $p_i \in P$  and the document embedding  $d$ , and  $w_i$  the bonus weight corresponding to  $p_i$ :

$$\widetilde{\text{sim}}(p_i, d) = \frac{\text{sim}(p_i, d) - \min_{p_j \in P} \text{sim}(p_j, d)}{\max_{p_j \in P} \text{sim}(p_j, d) - \min_{p_j \in P} \text{sim}(p_j, d)}, \quad (1)$$

$$S(p_i) = (1 + w_i) \times \widetilde{\text{sim}}(p_i, d). \quad (2)$$

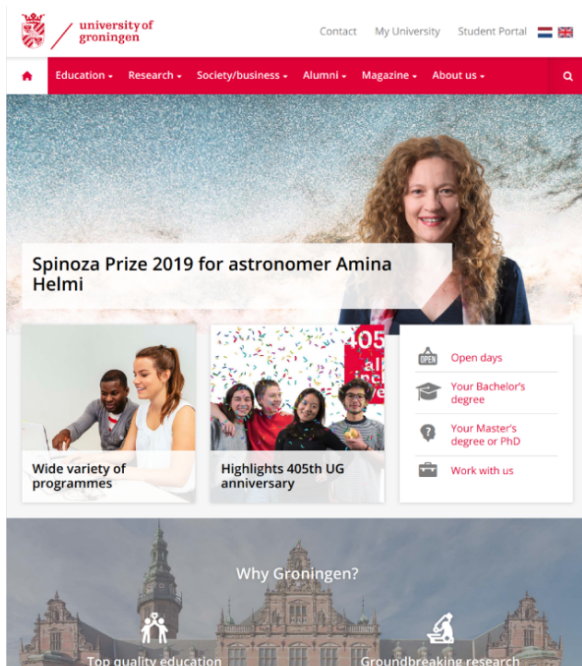


**Table 1.** The bonus weighting scheme for candidate phrases using WebEmbedRank.  $n$  is the number of words in the phrase.

Element	Weight
Hostname	4
Title	1.5
Headings	1
Description	0.5
Others	0
Multi-word penalty	$-0.25 \times (n - 1)$

This scoring method allows a candidate phrase that is semantically similar to the complete document to get a significant boost in the ranking if it appears in a key location. At the same time, phrases that are highly dissimilar to the document will get a cosine similarity score closer to zero, making the bonus multiplier much less effective. This ensures that semantically representative keyphrases in good locations will get a high rank, while still being robust against irrelevant words that are placed in the important HTML elements. For completeness, we also evaluated a variant of the WebEmbedRank method with the MMR diversity mechanism from EmbedRank++ as explained in Section 2.3, which we refer to as WebEmbedRank++. In this case, the MMR filtering is applied using the relevance scores as computed in Equation (2).

By combining both a priori knowledge about the importance of specific elements and recently developed text sequence embedding methods, we aim to provide a new method that combines the best of both worlds. WebEmbedRank uses semantic information from the complete web page, and is strengthened with structural information where available. This combination makes it more robust to the inconsistent nature of web pages, where it can not be assumed that any of the elements are always present. Figure 1 shows an example of this, illustrating the keyphrases extracted by WebEmbedRank from the homepage of the University of Groningen along with the text from the bonus elements. Most selected keyphrases appear often in the bonus elements, but not all phrases in those elements are given high scores. Similarly, the word ‘Faculty’ only appeared on the bottom of the page (not pictured) and not in any of the HTML elements that provide a bonus weight, but still got extracted due to its high similarity to the document.



**Keyphrases:** University, Groningen, UG, Library, Faculty, Top quality education, Research, Academic teaching, Vibrant student city, Degree programmes

**Hostname:** www.rug.nl

**Title:** Top 100 University | University of Groningen

**Description:** The University of Groningen has a high reputation for academic teaching and research and a large international network and offers degree programmes at Bachelor, Master and PhD levels in virtually every field.

**Headings:** 'Wide variety of programmes', 'The nobel prize', 'Over 120 nationalities', 'Groundbreaking research', 'Library', 'Four UG scientists to present lectures at Oerol', 'Cisca Wijmenga receives prestigious genetics award', 'Work with us', 'News and events', 'Pauline Westerman and Cisca Wijmenga nominated for Huibregtsen prize 2019', 'Vibrant student city', 'UG climbs to 114th place in the QS ranking', 'Highlights 405th UG anniversary', 'Top quality education', 'In our magazine: From fish jaws to patented leg prosthesis', 'Why Groningen?', 'Van Rijn's multi-million euro reduction disastrous for University of Groningen', 'Spinoza prize 2019 for astronomer Amina Helmi'

**Figure 1.** Top-10 keyphrases as extracted by WebEmbedRank from [www.rug.nl](http://www.rug.nl), along with the parsed text from the important HTML elements. The keyphrases are marked in red.

## 4. Materials and Methods

### 4.1. Text Extraction and Preprocessing

In order to rank phrases from a web page, the page first has to be parsed to extract all relevant text. This parsing of raw HTML-code was done using the BeautifulSoup (<https://www.crummy.com/software/BeautifulSoup/>) Python package. A potential problem with extracting text from HTML is that some elements are used to stylize words in sentences, or even parts of words. To illustrate this, consider the following snippet of HTML code:

```
<div>
<b>J</b>oe's <b>R</b>estaurant
</div>.
```

In this case, the text 'Joe's Restaurant' should be extracted, though using all elements as sentence separators results in this small text fragment being separated into four words. This happens because the bold element `<b>` is not used as a structural, but as a styling element. Therefore, all elements that were considered style elements were ignored, moving their content directly into the parent node. Furthermore, elements that did not display their content when the page was rendered were removed. As an example, the script element is normally filled with code that is executed when the page is rendered, and does not contain any relevant text. All content within these elements was therefore not considered. Two notable exceptions to this were the title and the description, which are often in the invisible head and meta elements, as these do tend to contain text relevant to the page and company. The complete document was constructed in a hierarchical fashion, starting with the title, followed by the description, headings and finally the rest of the elements. The content from the rest of the elements were placed in order of appearance in the HTML code.

In order to do the weighting for WebEmbedRank, the elements as mentioned in Table 1 were extracted and stored while parsing the page. The matching for assigning the bonus weights was done using only complete words. This meant that, for example, the term 'day' would not get a weight increase when the word 'daycare' was present in the title. Furthermore, before the weighting the hostname was stripped of the protocol specifier and the 'www.' (e.g., 'http://www.'), and the top-level domain (e.g., '.co.uk'). To check whether a multi-word phrase was present in the hostname, the words from the candidate phrase were concatenated before matching.

### 4.2. Evaluation

In order to evaluate the quality of keyphrase extraction methods, there are several annotated datasets available that are frequently used as benchmarks. Examples of these are the Inspec dataset consisting of scientific abstracts [24], the keyphrase extraction part of the SemEval dataset on scientific papers [25], and the set of news articles made by Marujo et al. [26]. However, as WebEmbedRank was specifically designed for web pages, these commonly used datasets can not be used for the evaluation. As far as we know, there are no comparable benchmarks publicly available for web pages. Given this lack of gold-standard datasets, we created our own. To select a set of web pages that was representative of the content on the Web, we used Dataprovider.com's database of domains, which contains over 250 million indexed hostnames. From this database, 105 hostnames were randomly selected, using some filters to improve the quality of the data. These filters constrained the data to web pages where English was detected as the main language, as our tokenizer and embedding model were trained on English texts. We furthermore included the constraints that the homepage still had to be online, was not a placeholder or parked domain, and contained at least some text. The resulting dataset consisted mostly of company web pages, but also included blogs and pages of local communities. Three annotators were asked to annotate this set by going to each web page, and to write down the keyphrases they thought best described the web page. They were given three main guidelines:

- Keyphrases must be present on the homepage (text in title or meta description is also allowed).
- Keyphrases consist of at least 1 and at most 5 words.
- Choose at least 3 and at most 8 keyphrases per page.

Two of the websites were further rejected for not containing enough content, leaving a final of 103 annotated web pages. Similar to [5,8], the union of the three sets of keyphrases for each web page was used for the evaluation. The final combined dataset contained an average of 10.02 keyphrases per document, while each annotator assigned on average 5.19 keywords per document. The amount of text on the web pages after parsing the HTML varied between 38 to 4244 words, with an average of 519 words. After filtering out phrases that did not fit the part-of-speech criterion, the number of candidate keyphrases per page ranged from 29 to 808, averaging 174 candidates. To detect wrong or misspelled annotations, all annotations that were not in the extracted text from the page were removed. The text extraction and preprocessing were performed only once, after which all parsed content was stored in page snapshots. This ensured that all methods were evaluated using identical web pages as input. To avoid copyright infringement, we can not share the exact snapshots of the web pages used for the evaluation; however, we published (<https://gitlab.com/Tim-Haarman/WebEmbedRank>) the set of annotations in the hope that other researchers can use this for their experiments, and can hopefully extend it.

#### 4.3. Comparison

To evaluate the performance of our keyphrase extraction method, we compared it to several baselines and state-of-the-art extraction methods. Similar to earlier research [3,4,7,24], the methods were evaluated based on their precision, recall and  $F_1$  score with respect to the web page annotations. In the matching of the keyphrases, we only accepted exact matches of the complete phrase. We will briefly describe the evaluated methods.

The first method was the term frequency-inverse document frequency (TF-IDF) [27], a frequently used and robust baseline that uses frequency information from a corpus to determine the relevance of one or more words. TF-IDF works based on the idea that words that are relevant for a document occur frequently in that document, yet infrequently in other documents from a large corpus. It consists of two parts, being the term frequency and the inverse document frequency. By multiplying the frequency of the word in a document by an inverse of how frequently it appears in other documents, a relative frequency score is calculated. The top keyphrases were selected by taking the phrases with the highest TF-IDF scores.

The Location method was a simple heuristic baseline that was based on the aforementioned theory that certain HTML elements have varying levels of relevance. It applied a part-of-speech (POS) tagger to all texts, and selected all the longest chains of zero or more adjectives followed by one or more nouns as candidates. The candidates were ranked based on which HTML element they appeared in. This hierarchical ranking method ranked candidates that occurred in the hostname (URL) the highest, followed by the title, headings, description, and all remaining elements respectively. If multiple candidates were present in an element, they were ranked in the order of appearance in the original HTML code.

We further compared WebEmbedRank and WebEmbedRank++ to four state-of-the-art graph based models as described in Section 2, being TextRank [4], TopicRank [6], PositionRank [3] and MultipartiteRank [7]. TextRank, TopicRank, PositionRank and MultipartiteRank were implemented using the Python keyphrase extraction (PKE) toolkit [28]. Lastly, we also compared it to the original implementation of EmbedRank [5], using both the standard version (EmbedRank) and the version with diversity mechanism (EmbedRank++), each using Sent2vec as embedding method. EmbedRank and EmbedRank++ were tested using the original implementation of the authors: (<https://github.com/swisscom/ai-research-keyphrase-extraction>). For all methods we used the versions and parameters as suggested by the authors. Similar to [5], we used  $\lambda = 0.5$  for EmbedRank++ and WebEmbedRank++, indicating an equal importance of the relevance and diversity of the candidate



phrases. The tokenization and POS-tagging was performed using the Stanford CoreNLP toolkit [29] for all methods.

## 5. Results

As shown in Table 2, WebEmbedRank outperformed all other methods, achieving the highest precision, recall and  $F_1$  score for both the top 5 as well as the top 10 ranked keyphrases. For  $N = 5$ , paired  $t$ -tests showed significant improvements at  $\alpha = 0.05$  of WebEmbedRank over TF-IDF, TextRank, TopicRank, PositionRank, MultipartiteRank, EmbedRank and EmbedRank++, all at  $p < 0.001$ . The increase over WebEmbedRank++ ( $p = 0.12$ ) and Location ( $p = 0.08$ ) were not found to be significant. For  $N = 10$ , paired  $t$ -tests showed a significant increase for WebEmbedRank at  $\alpha = 0.05$  over all other methods at  $p < 0.001$  for all tests. Surprisingly, the simple location based heuristic got the second highest scores. This indicates the importance of considering the hypertext markup in the extraction process. Even the TF-IDF method scores on par or even better than most other state-of-the-art methods. The low scores of most of these extraction methods relative to the baselines may suggest that these methods transition poorly from normal documents to the noisy and inconsistent structure of web pages.

**Table 2.** Scores of WebEmbedRank compared to various other state-of-the-art keyphrase extraction methods on the newly constructed web page dataset for the top 5 and top 10 extracted phrases. **P** denotes the precision, **R** the recall and **F<sub>1</sub>** the micro- $F_1$  score. The highest scores are marked in bold.

Method	$N = 5$			$N = 10$		
	P	R	F <sub>1</sub>	P	R	F <sub>1</sub>
TF-IDF	30.49	16.20	20.78	20.97	21.81	21.00
Location	40.00	21.11	27.09	29.42	30.13	29.29
TextRank	15.73	8.37	10.74	14.94	15.54	14.98
TopicRank	29.51	15.47	19.92	22.04	22.86	22.01
PositionRank	24.85	13.43	17.14	21.36	22.61	21.57
MultipartiteRank	32.82	17.24	22.22	24.47	25.47	24.50
EmbedRank	11.84	6.53	8.24	16.02	17.06	16.22
EmbedRank++ ( $\lambda = 0.5$ )	11.07	5.63	7.34	10.10	10.54	10.09
WebEmbedRank	<b>43.69</b>	<b>23.37</b>	<b>29.94</b>	<b>34.56</b>	<b>36.41</b>	<b>34.81</b>
WebEmbedRank++ ( $\lambda = 0.5$ )	41.36	22.10	28.27	25.83	26.86	25.82

The lowest scores in both tests are from the original EmbedRank and EmbedRank++ methods. By inspecting the generated keyphrases, we found that this may partially have been due to its tendency to generate long keyphrases, as explained in Section 3. The phrases extracted by these methods contained on average 2.43 and 2.13 words, respectively, compared to an average 1.51 words for WebEmbedRank and 1.61 words for the annotated phrases in the gold-standard evaluation set. We observed the same problem with PositionRank, which extracted phrases averaging 2.36 words. As PositionRank constructs phrases by summing the scores of the individual words, a strong bias towards long phrases is created, even when some of the words do not contribute much to the overall score. In the evaluation of the original proposal [3], this problem is largely avoided by limiting the keyphrase generation to at most tri-grams; however, it can be argued that such a hard cut-off is undesirable.

From the state-of-the-art models, MultipartiteRank achieved the highest  $F_1$  scores. This may partially be attributed to the positional component in this method. As the documents were constructed hierarchically, the phrases from the more important elements were effectively given a higher weight, as these were placed at the beginning of the document. Unlike PositionRank, which also incorporates a positional score but got lower scores, MultipartiteRank did not suffer from generating too lengthy keyphrases. The extracted phrases contained on average approximately 1.61 phrases, almost identical to the annotated set.

Finally, the EmbedRank++ and WebEmbedRank++ versions with the diversity mechanism both scored lower than their counterparts without diversity. Interestingly, the scores are closer for the top 5 keyphrases, going from 29.94 to 28.27 for WebEmbedRank and WebEmbedRank++ respectively. Looking at the top 10 phrases the scores are much lower when the diversity method is enabled, with an  $F_1$  score of 34.81 for WebEmbedRank and 25.82 for WebEmbedRank++. A similar effect can be seen between EmbedRank and EmbedRank++.

## 6. Discussion

In this paper, we proposed WebEmbedRank, an extension of EmbedRank [5] for keyphrase extraction from web pages. WebEmbedRank uses text embeddings to extract only the phrases that are relevant to the document, and strengthens the ranking with a weighting procedure based on the structural information in the document. By using the structural information where available to strengthen the ranking process, but not solely relying on it, WebEmbedRank can effectively deal with the inconsistency in web documents.

We furthermore showed that most state-of-the-art unsupervised keyphrase extraction methods transfer poorly to web documents. Multiple methods had a tendency to extract very long phrases, which may explain the lower scores. This might be due to the different nature of the data, as most keyphrase extraction systems are evaluated on datasets of scientific abstracts and articles. These documents often comprise multiple complex topics, which may require longer keyphrases to accurately represent the document compared to web pages. To check this hypothesis, we analyzed the assigned keyphrases in the training part of the Inspec [24] and SemEval 2010 [25] datasets, which are some of the most frequently used datasets to evaluate unsupervised keyphrase extraction methods [8]. We found that they respectively contained on average 2.33 and 2.16 words per phrase, which is substantially more than the average of 1.61 in our web page dataset. This disparity can potentially explain why some methods are more geared towards extracting longer phrases, and hence why their results are poor when transferred to the present domain and evaluation set.

Similar to the results reported in [5], the diversity mechanism that EmbedRank++ and WebEmbedRank++ employ did not increase the scores. However, we did notice a stronger relative decrease of the  $F_1$  score at the top 10 keyphrases compared to the top 5 keyphrases. This may be explained by the lack of relevant and sufficiently diverse keyphrases on a web page, as the aim and topic of a web page tends to be quite specific. This is also shown by the fact that the annotators assigned 5.2 keyphrases on average per web page, even though they could provide eight. Forcing diversity during the generation of 10 keyphrases may result in irrelevant phrases being extracted when there are only a few true main topics.

In this paper, we focused the weighting on what we perceived to be the most important elements, being the hostname, title, headings and description. Earlier work has shown some indications that other elements may also have positive or negative effects on the relevance of their contents, such as text style elements (bold, italic, etc.) or anchor text [11,30]. Additionally, we could experiment with different weights for the different sizes of headers (h1 to h6). We furthermore only focused on web pages with English text. Given the unsupervised nature of WebEmbedRank, it is expected that this method can be generalized to other languages relatively easily. The word embeddings can be trained in the same way for different languages, as long as there is some large set of written text available in that language. Given that the fastText method already provides pre-trained embedding models for 157 languages [31], this is not expected to be a problem. Different languages will also require an automatic POS-tagger trained for that language; however, the Stanford CoreNLP POS-tagger is available in most major languages [29]. Whether the same rules about what constitutes a good keyphrase also apply to different languages remains to be answered, which we will leave for future work.

**Author Contributions:** Conceptualization, T.H.; methodology, T.H., B.Z. and M.W.; software, T.H.; validation, B.Z. and M.W.; formal analysis, T.H.; investigation, T.H.; resources, B.Z. and M.W.; data curation, T.H.; writing—original draft preparation, T.H.; writing—review and editing, B.Z. and M.W.; visualization, T.H.; supervision, B.Z. and M.W.; project administration, T.H., B.Z. and M.W.

**Funding:** This research received no external funding.

**Acknowledgments:** We thank the annotators for their help with constructing the evaluation set presented in this research. We furthermore thank the anonymous reviewers for their helpful feedback.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Van den Bosch, A.; Bogers, T.; De Kunder, M. Estimating search engine index size variability: A 9-year longitudinal study. *Scientometrics* **2016**, *107*, 839–856. [[CrossRef](#)] [[PubMed](#)]
2. Hu, Y.; Xin, G.; Song, R.; Hu, G.; Shi, S.; Cao, Y.; Li, H. Title extraction from bodies of HTML documents and its application to web page retrieval. In Proceedings of the 28th Annual International Acm Sigir Conference on Research and Development in Information Retrieval, Salvador, Brazil, 15–19 August 2005; pp. 250–257.
3. Florescu, C.; Caragea, C. PositionRank: An unsupervised approach to keyphrase extraction from scholarly documents. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, Vancouver, BC, Canada, 30 July–4 August 2017; Volume 1, pp. 1105–1115.
4. Mihalcea, R.; Tarau, P. TextRank: Bringing Order into Text. In Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing, Barcelona, Spain, 25–26 July 2004; Association for Computational Linguistics: Barcelona, Spain, 2004; pp. 404–411.
5. Bennani-Smires, K.; Musat, C.; Hossmann, A.; Baeriswyl, M.; Jaggi, M. Simple Unsupervised Keyphrase Extraction using Sentence Embeddings. In Proceedings of the 22nd Conference on Computational Natural Language Learning, Brussels, Belgium, 31 October–1 November 2018; pp. 221–229.
6. Bougouin, A.; Boudin, F.; Daille, B. TopicRank: Graph-based topic ranking for keyphrase extraction. In Proceedings of the International Joint Conference on Natural Language Processing (IJCNLP), Nagoya, Japan, 14–18 October 2013; pp. 543–551.
7. Boudin, F. Unsupervised Keyphrase Extraction with Multipartite Graphs. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, New Orleans, LA, USA, 1–6 June 2018; Association for Computational Linguistics: New Orleans, LA, USA, 2018; Volume 2, pp. 667–672. [[CrossRef](#)]
8. Hasan, K.S.; Ng, V. Automatic keyphrase extraction: A survey of the state of the art. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, Baltimore, MD, USA, 22–27 June 2014; Volume 1, pp. 1262–1273.
9. Brin, S.; Page, L. The anatomy of a large-scale hypertextual web search engine. *Comput. Netw. ISDN Syst.* **1998**, *30*, 107–117. [[CrossRef](#)]
10. Yih, W.T.; Goodman, J.; Carvalho, V.R. Finding advertising keywords on web pages. In Proceedings of the 15th International Conference on World Wide Web, Scotland, UK, 23–26 May 2006; pp. 213–222.
11. Choi, B.; Yao, Z. web page classification. In *Foundations and Advances in Data Mining*; Springer: Heidelberg/Berlin, Germany, 2005; pp. 221–274.
12. Hammouda, K.M.; Kamel, M.S. Efficient phrase-based document indexing for web document clustering. *IEEE Trans. Knowl. Data Eng.* **2004**, *16*, 1279–1296. [[CrossRef](#)]
13. Zhang, Y.Z.; Zincir-Heywood, N.; Milios, E. Summarizing web sites automatically. In Proceedings of the Conference of the Canadian Society for Computational Studies of Intelligence, Halifax, NS, Canada, 11–13 June 2003; Springer: Heidelberg/Berlin, Germany, 2003; pp. 283–296.
14. Thomaidou, S.; Vazirgiannis, M. Multiword keyword recommendation system for online advertising. In Proceedings of the 2011 International Conference on Advances in Social Networks Analysis and Mining, Kaohsiung, Taiwan, 25–27 July 2011; pp. 423–427.
15. Riboni, D. Feature selection for web page classification. In Proceedings of the Workshop on Web Content Mapping: A Challenge to ICT, Shiraz, Iran, 29–31 October 2002.

16. Kwon, O.W.; Lee, J.H. Text categorization based on k-nearest neighbor approach for web site classification. *Inf. Proc. Manag.* **2003**, *39*, 25–44. [[CrossRef](#)]
17. Wan, X.; Xiao, J. Single Document Keyphrase Extraction Using Neighborhood Knowledge. In Proceedings of the 23rd National Conference on Artificial Intelligence, Chicago, IL, USA, 13–17 July 2008; Volume 8, pp. 855–860.
18. Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.S.; Dean, J. Distributed representations of words and phrases and their compositionality. In Proceedings of the 26th International Conference on Neural Information Processing Systems, Lake Tahoe, NV, USA, 5–10 December 2013; pp. 3111–3119.
19. Almeida, F.; Xexéo, G. Word Embeddings: A Survey. *arXiv* **2019**, arXiv:1901.09069. Available online: <https://arxiv.org/abs/1901.09069> (accessed on 25 June 2019).
20. Le, Q.; Mikolov, T. Distributed representations of sentences and documents. In Proceedings of the International conference on Machine Learning, Beijing, China, 21–26 June 2014; pp. 1188–1196.
21. Pagliardini, M.; Gupta, P.; Jaggi, M. Unsupervised Learning of Sentence Embeddings using Compositional n-Gram Features. In Proceedings of the NAACL 2018—Conference of the North American Chapter of the Association for Computational Linguistics, New Orleans, LA, USA, 2–7 February 2018.
22. Kan, M.Y. web page classification without the web page. In Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers & Posters, New York, NY, USA, 19–21 May 2004; pp. 262–263.
23. Kan, M.Y.; Thi, H.O.N. Fast webpage classification using URL features. In Proceedings of the 14th ACM International Conference on Information and Knowledge Management, Bremen, Germany, 31 October–5 November 2005; pp. 325–326.
24. Hulth, A. Improved automatic keyword extraction given more linguistic knowledge. In Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing, EMNLP 2003, Sapporo, Japan, 11–12 July 2003; pp. 216–223.
25. Kim, S.N.; Medelyan, O.; Kan, M.Y.; Baldwin, T. Semeval-2010 task 5: Automatic keyphrase extraction from scientific articles. In Proceedings of the 5th International Workshop on Semantic Evaluation, Uppsala, Sweden, 15–16 July 2010; pp. 21–26.
26. Marujo, L.; Gershman, A.; Carbonell, J.; Frederking, R.; Neto, J.P. Supervised topical key phrase extraction of news stories using crowdsourcing, light filtering and co-reference normalization. In Proceedings of the Eighth International Language Resources and Evaluation (LREC), Istanbul, Turkey, 21–27 May 2012.
27. Salton, G.; Buckley, C. Term-weighting approaches in automatic text retrieval. *Inf. Process. Manag.* **1988**, *24*, 513–523. [[CrossRef](#)]
28. Boudin, F. PKE: An open source Python-based keyphrase extraction toolkit. In Proceedings of the 26th International Conference on Computational Linguistics: System Demonstrations, Osaka, Japan, 11–16 December 2016; pp. 69–73.
29. Manning, C.; Surdeanu, M.; Bauer, J.; Finkel, J.; Bethard, S.; McClosky, D. The Stanford CoreNLP natural language processing toolkit. In Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, Baltimore, MD, USA, 22–27 June 2014; pp. 55–60.
30. Craven, T.C. HTML tags as extraction cues for web page description construction. *Inf. Sci.* **2003**, *6*, 1–12. [[CrossRef](#)]
31. Grave, E.; Bojanowski, P.; Gupta, P.; Joulin, A.; Mikolov, T. Learning Word Vectors for 157 Languages. In Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018), Miyazaki, Japan, 7–12 May 2018.

