EmbedRank: Unsupervised Keyphrase Extraction using Sentence Embeddings

Kamil Bennani-Smires[1], **Claudiu Mușat**[1], **Andreea Hossmann**[1], **Michael Baeriswyl**[1], **Martin Jaggi**[2],
Artificial Intelligence and Machine Learning Group, Swisscom [1]
firstname.lastname@swisscom.com
Machine Learning and Optimization Laboratory, EPFL [2]
martin.jaggi@epfl.ch

Abstract

Keyphrase extraction is the task of automatically selecting a small set of phrases that best describe a given free text document. Supervised keyphrase extraction requires large amounts of labeled training data and generalizes very poorly outside the domain of the training data. At the same time, unsupervised systems have poor accuracy, and often do not generalize well, as they require the input document to belong to a larger corpus also given as input. Furthermore, both supervised and unsupervised methods are often *too slow* for real-time scenarios and suffer from *over-generation*.

Addressing these drawbacks, in this paper, we tackle keyphrase extraction from single documents with EmbedRank: a novel unsupervised method, that leverages sentence embeddings. EmbedRank achieves higher F-scores than graph-based state of the art systems on standard datasets and is suitable for real-time processing of large amounts of Web data. With EmbedRank, we also explicitly increase coverage and diversity among the selected keyphrases by introducing an embedding-based maximal marginal relevance (MMR) for new phrases. A user study including over 200 votes showed that, although reducing the phrases' semantic overlap leads to no gains in F-score, our high diversity selection is preferred by humans.

1 Introduction

Document keywords and keyphrases enable faster and more accurate search in large text collections, serve as condensed document summaries, and are used for various other applications, such as categorization of documents. In particular, keyphrase extraction is a crucial component when gleaning real-time insights from large amounts of Web and social media data, which is a now routine task in companies across the world. In this case, it is essential for the extraction to be *fast* and for the keyphrases to be *disjoint*. However, existing systems are complex and slow, and are plagued by over-generation, i.e. extracting redundant keyphrases (e.g., "*European Commission*" and "*Commission*").

Here, we address both these problems with a new unsupervised algorithm.

Unsupervised keyphrase extraction has a series of advantages over supervised methods. Supervised keyphrase extraction always requires the existence of a (large) annotated corpus of both documents and their manually selected keyphrases to train on - a very strong requirement in most cases. In addition, supervised methods perform poorly outside of the domain represented by the training corpus - a big issue, considering that the domain of new documents may not be known at all. Unsupervised keyphrase extraction addresses such information-constrained situations in one of two ways: (a) by relying on in-corpus statistical information (e.g., the inverse document frequency of the words), and the current document; (b) by only using information extracted from the current document.

We propose EmbedRank - an unsupervised method to automatically extract keyphrases from a document, that is both simple and *only requires the current document itself*, rather than an entire corpus that this document may be linked to.

Our method relies on notable new developments in text representation learning [Le *et al.*, 2014; Kiros *et al.*, 2015; Pagliardini *et al.*, 2017], where documents or word sequences of arbitrary length are embedded into the same continuous vector space. This opens the way to computing semantic relatedness among text fragments by using the induced similarity measures in that feature space. Using these semantic text representations, we guarantee the two most challenging properties of keyphrases: *informativeness* obtained by the distance between the embedding of a candidate phrase and that of the full document; *diversity* expressed by the distances among candidate phrases themselves.

In a traditional F-score evaluation, EmbedRank clearly **outperforms the current state of the art** (i.e. complex graph-based methods [Mihalcea and Tarau, 2004; Wan and Xiao, 2008; Rui Wang, Wei Liu, 2015]) on two out of three common datasets for keyphrase extraction; it matches the state of the art on the third dataset. We also evaluated the impact of **ensuring diversity** by conducting a user study, since this aspect cannot be captured by the F-score evaluation. The study showed that users highly prefer keyphrases with the diversity property.

Finally, to the best of our knowledge, we are the first to present an unsupervised method based on phrase and document embeddings for keyphrase extraction, as opposed to standard individual word embeddings.

The paper is organized as follows. Related work on

keyphrase extraction and sentence embeddings is presented in Section 2. In Section 3 we present how our method works. An enhancement of the method allowing us to gain a control over the redundancy of the extracted keyphrases is then described in Section 4. Finally, Section 5 contains the different experiments that we performed.

2 Related Work

A comprehensive, albeit slightly dated survey on keyphrase extraction is available [Hasan and Ng, 2011]. Here, we focus on unsupervised methods, as they are superior in many ways (domain independence, no training data) and represent the state of the art in performance. As EmbedRank relies heavily on (sentence) embeddings, we also discuss the state of the art in this area in the second part of this section.

2.1 Unsupervised Keyphrase Extraction

Unsupervised keyphrase extraction comes in two flavors: corpus-dependent [Wan and Xiao, 2008] and corpus-independent.

Corpus-independent methods, including our proposed method, require no other inputs than the one document from which to extract keyphrases. Most such existing methods are graph-based, with the notable exceptions of KeyCluster [Liu *et al.*, 2009] and TopicRank [Bougouin *et al.*, 2013]. In graph-based keyphrase extraction, first introduced with TextRank [Mihalcea and Tarau, 2004], the target document is a graph, in which nodes represent words and edges represent the co-occurrence of the two endpoints inside some window. The edges may be weighted, like in SingleRank [Wan and Xiao, 2008], using the number of co-occurrences as weights. The words (or nodes) are scored using some node ranking metric, such as degree centrality or PageRank [Page, 1998]. Scores of individual words are then aggregated into scores of multi-word phrases. Finally, sequences of consecutive words which respect a certain sequence of part-of-speech tags are considered as candidate phrases and ranked by their scores.

Recently, WordAttractionRank [Rui Wang, Wei Liu, 2015] followed an approach similar to SingleRank, with the difference of using a new weighting scheme for edges between two words, to incorporate the distance between their word embedding representation.

[Florescu and Caragea, 2017] use node weights, favoring words appearing earlier in the text.

Departing from the popular graph approach, KeyCluster [Liu *et al.*, 2009] introduces a clustering-based approach. The words present in the target document are clustered and, for each cluster, one word is selected as an "exemplar term". Candidate phrases are filtered as before, using the sequence of part-of-speech tags and, finally, candidates which contain at least one exemplar term are returned as the keyphrases.

TopicRank [Bougouin *et al.*, 2013] combines the graph and clustering-based approaches. Candidate phrases are first clustered, then a graph where each node represents a cluster is created. The top N clusters are found using a centrality metric and the keyphrases are computed by selecting the most representative member of each cluster. Clustering the candidate phrases reduces redundancy and improves diversity. However, TopicRank clusters phrases based on the percentage of shared words, resulting in e.g., "*fantastic teacher*" and "*great instructor*" not being clustered together, despite expressing the same idea.

EmbedRank differs from the aforementioned methods as it represents both the document and all candidate phrases as vectors in a high-dimensional space, relying on state-of-the-art semantic document embedding methods beyond simple averaging of word vectors. In the resulting vector space, we can therefore compute meaningful distances between a candidate phrase and the document (for informativeness), as well as the semantic distance between candidates (for diversity).

2.2 Word and Sentence Embeddings

Word2Vec [Mikolov *et al.*, 2013] and related methods marked a very impactful advancement in representing words as vectors in a continuous vector space. Representing words with vectors in moderate dimensions solves several major drawbacks of the classic bag-of-words representation, including the lack of semantic relatedness between words and the very high dimensionality (size of the vocabulary).

Different methods are needed for representing entire sentences or documents. Skip-Thought [Kiros *et al.*, 2015] provides sentence embeddings by means of recurrent neural networks, trained to predict neighboring sentences. Paragraph Vector [Le *et al.*, 2014] finds paragraph embeddings using an unordered list of paragraphs. The method can be generalized to also work on sentences or entire documents, turning paragraph vectors into more generic document vectors [Lau and Baldwin, 2016].

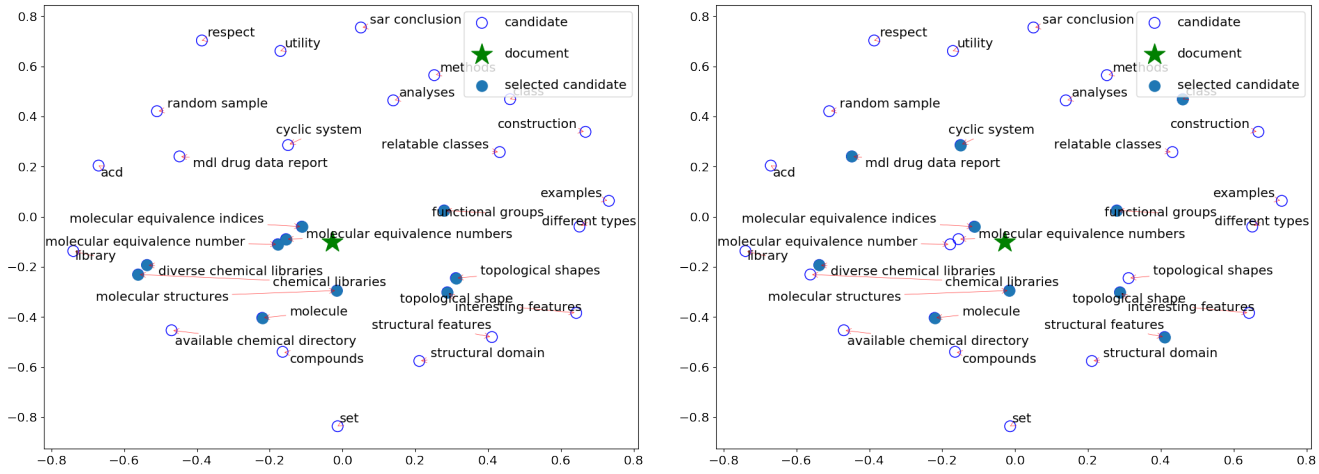
Sent2Vec [Pagliardini *et al.*, 2017] uses word n-gram features to produce sentence embeddings. It produces word and n-gram vectors, which are specifically trained such that they can be additively combined into a sentence vector, as opposed to general word-vectors. Sent2Vec features much faster inference than Paragraph Vector [Le *et al.*, 2014] or Skip-Thought [Kiros *et al.*, 2015]. Similarly to recent word and document embeddings, Sent2Vec reflects semantic relatedness between phrases when using standard similarity measures on the corresponding vectors. This property is at the core of our method, as we show it outperforms competing embedding methods for keyphrase extraction.

3 EmbedRank: From Embeddings to Keyphrases

In this and the next section, we introduce and describe our novel keyphrase extraction method, EmbedRank. The method consists of three main steps, as follows: (1) We extract candidate phrases from the text, based on part-of-speech sequences. More precisely, we keep only those phrases that consist of zero or more adjectives followed by one or multiple nouns [Wan and Xiao, 2008]. (2) We use sentence embeddings to represent (**embed**), both the candidate phrases and the document itself in the same high-dimensional vector space (Sec. 3.1). (3) We **rank** the candidate phrases to select the output keyphrases (Sec. 3.2). In addition, in the next section, we show how to improve the ranking step, by providing a way to tune the diversity of the extracted keyphrases.

3.1 Embedding the Phrases and the Document

State-of-the-art text embeddings (word, sentence, document) have the remarkable property of capturing semantic related-



(a) EmbedRank (without diversity)

(b) EmbedRank++ (with diversity)

Figure 1: Embedding space[1] of a scientific abstract entitled *"Using molecular equivalence numbers to visually explore structural features that distinguish chemical libraries"*

ness via the distances between the corresponding vector representations within the shared vector space. In this step, we use this property to rank the candidate phrases extracted in the first step above, by measuring their distance to the original document. Thus, semantic relatedness between a candidate phrase and its document becomes a proxy for informativeness of the phrase.

Concretely, this second step of our keyphrase extraction method consists of:

(a) Computing the *document embedding*. This includes a noise reduction procedure, where we keep only the adjectives and nouns contained in the input document.

(b) Computing the *embedding of each candidate phrase* separately, again with the same algorithm.

To determine the impact the document embedding method may have on the final outcome, we evaluate keyphrases obtained using both the popular Doc2Vec [Lau and Baldwin, 2016] (denoted EmbedRank d2v) and ones based on the newer Sent2vec [Pagliardini *et al.*, 2017] (denoted EmbedRank s2v).

Both embedding methods **allow us to embed arbitrary-length sequences of words**. To embed both phrases and documents, we employ publicly available pre-trained models of Sent2Vec[2] and Doc2vec[3]. The pre-computed Sent2vec embeddings based on words and n-grams vectors have $Z = Z_s = 700$ dimensions, while for Doc2vec $Z = Z_d = 300$. All embeddings are trained on the large English Wikipedia corpus.[4] EmbedRank s2v is very fast, since Sent2vec infers a document embedding from the pre-trained model, by averaging the pre-computed representations of the text's components (words and n-grams), in a single linear pass through the text. EmbedRank d2v is slower, as Doc2vec uses the embedding network to infer a vector for the whole document. Both methods provide vectors comparable in the same semantic space, no matter if the input "document" is a word, a phrase, a sentence or an entire document.

After this step, we have one Z-dimensional vector representing our document and a Z-dimensional vector for each of our candidate phrases, all sharing the same reference space. Figure 1a shows a concrete example, using EmbedRank s2v, from one of the datasets we used for evaluation (scientific abstracts). As can be seen by comparing document titles and candidate phrases, our initial assumption holds in this example: the closer a phrase is to the document vector, the more informative that phrase is for the document. Therefore, it is sensible to use the cosine similarity between the embedding of the candidate phrase and the document embedding as a measure of informativeness.

3.2 Selecting the Top Candidates

Based on the above, we select the top keyphrases out of the initial set, by ranking the candidate phrases according to their cosine distance to the document embedding. In Figure 1a, this results in ten highlighted keyphrases, which are clearly in line with the document's title.

Nevertheless, it is notable that there can be significant redundancy in the set of top keyphrases. For example, "*molecular equivalence numbers*" and "*molecular equivalence in-*

[1] Visualization based on multidimensional scaling with cosine distance on the original $Z = Z_s = 700$ dimensional embeddings.

[2] https://github.com/epfml/sent2vec

[3] https://github.com/jhlau/doc2vec

[4] The generality of this corpus, as well as the unsupervised embedding method itself ensure that the computed text representations are general-purpose, thus domain-independent.

Dataset	Documents	Avg tok	Avg cand	Keyphrases	Avg kp	Missing kp in doc	Missing kp in cand	Missing due to cand
Inspec	500	134.63	26.39	4903	9.81	21.52%	39.85%	18.34%
Duc	308	850.02	138.47	2479	8.05	2.18%	12.38%	10.21%
Nguyen	209	8448.55	765.56	2272	10.87	14.39%	30.85%	16.46%

Table 1: The three datasets we use. Columns are: number of documents; average number of tokens per document; average number of unique candidates per document; total number of unique keyphrases; average number of unique keyphrases per document; percentage of keyphrases not present in the documents; percentage of keyphrases not present in the candidates; percentage of keyphrases present in the document, but not in the candidates. These statistics were computed after stemming the candidates, the keyphrases and the document.

dices" are both selected as separate keyphrases, despite expressing the same meaning. This problem can be elegantly solved by once again using our phrase embeddings and their cosine similarity as a proxy for semantic relatedness. We describe our proposed solution to this in the next section.

Summarizing this section, we have proposed an unsupervised step-by-step method to extract *informative keyphrases* from a single document by using sentence embeddings.

4 EmbedRank++: Increasing Keyphrase Diversity with MMR

By returning the N candidate phrases closest to the document embedding, EmbedRank only accounts for the phrase informativeness property, leading to redundant keyphrases.

In scenarios where users directly see the extracted keyphrases (e.g. text summarization, tagging for search), this is problematic: redundant keyphrases adversely impact the user's experience. This can deteriorate to the point in which providing keyphrases becomes completely useless.

Moreover, if we extract a fixed number of top keyphrases, redundancy hinders the diversification of the extracted keyphrases. In the document from Figure 1a, the extracted keyphrases include {*topological shape, topological shapes*} and {*molecular equivalence number, molecular equivalence numbers, molecular equivalence indices*}. That is, four out of the ten keyphrase "slots" are taken by redundant phrases.

This problem bears strong resemblance to search result diversification [Drosou and Pitoura, 2010], where a search engine must provide search results which balance query-document relevance and document diversity. One of the simplest and most effective solutions to this is the Maximal Marginal Relevance (MMR) [Goldstein, 1998] metric, which combines in a controllable way the concepts of relevance and diversity. In the following, we show how to adapt MMR to keyphrase extraction, in order to combine keyphrase informativeness with dissimilarity among selected keyphrases.

The original MMR from information retrieval and text summarization is based on the set of all initially retrieved documents, R, for a given input query Q, and on an initially empty set S representing documents that are selected as good answers for Q. S is iteratively populated by computing MMR as described in Equation 1, where D_i and D_j are retrieved documents, and Sim_1 and Sim_2 are similarity functions.

$$\text{MMR} := \underset{D_i \in R \setminus S}{\arg\max} \left[\lambda \cdot Sim_1(D_i, Q) - (1-\lambda) \max_{D_j \in S} Sim_2(D_i, D_j) \right], \tag{1}$$

When $\lambda = 1$ MMR computes a standard, relevance-ranked list, while when $\lambda = 0$ it computes a maximal diversity ranking of the documents in R.

To use MMR here, we adapt the original equation as:

$$\text{MMR} := \underset{C_i \in C \setminus K}{\arg\max} \left[\lambda \cdot \widetilde{cos_{sim}}(C_i, \text{doc}) - (1-\lambda) \max_{C_j \in K} \widetilde{cos_{sim}}(C_i, C_j) \right], \tag{2}$$

where C is the set of candidate keyphrases, K is the set of extracted keyphrases, doc is the full document embedding, C_i and C_j are the embeddings of candidate phrases i and j, respectively. Finally, $\widetilde{cos_{sim}}$ is a normalized cosine similarity [Mori and Sasaki, 2003], described by the following equations. This ensures that, when $\lambda = 0.5$, the relevance and diversity parts of the equation have equal importance.

$$\widetilde{cos_{sim}}(C_i, \text{doc}) := 0.5 + \frac{ncos_{sim}(C_i, \text{doc}) - \overline{ncos_{sim}(C, \text{doc})}}{\sigma(ncos_{sim}(C, \text{doc}))}. \tag{3}$$

$$ncos_{sim}(C_i, \text{doc}) := \frac{cos_{sim}(C_i, \text{doc}) - \min_{C_j \in C} cos_{sim}(C_j, \text{doc})}{\max_{C_j \in C} cos_{sim}(C_j, \text{doc})} \tag{4}$$

We apply an analogous transformation for the similarities among the candidate phrases themselves.

Summarizing, the method presented in the previous section is equivalent to using our newly defined MMR for keyphrase extraction from Equation (2) with $\lambda = 1$. The generalized version of the algorithm, EmbedRank++, remains the same, except for the last step, where we instead use Equation (2) to perform the final selection of the N candidates, therefore returning simultaneously relevant and diverse keyphrases, tuned by the trade-off parameter λ.

5 Experiments and results

In this section we show that EmbedRank outperforms the graph-based state-of-the-art schemes on the most common datasets, when using traditional F-score evaluation. In addition, we report on the results of a sizable user study showing that, although EmbedRank++ achieves slightly lower F-scores than EmbedRank, users prefer the semantically diverse keyphrases it returns to those computed by the other method.

We first present the datasets on which we evaluate our method. Then we compare EmbedRank's performance to the state of the art. Finally, we describe the user study.

5.1 Datasets

We evaluate our methods on three common datasets for keyphrase extraction, summarized in Table 1.

The **Inspec** dataset [Hulth, 2003] consists of 2 000 short documents selected from scientific journal abstracts. For the

N	Method	Inspec			DUC			NUS		
		P	R	F_1	P	R	F_1	P	R	F_1
5	TextRank	24.87	10.46	14.72	19.83	12.28	15.17	5.00	2.36	3.21
	SingleRank	38.18	23.26	28.91	30.31	19.50	23.73	4.06	1.90	2.58
	TopicRank	33.25	19.94	24.93	27.80	18.28	22.05	**16.94**	**8.99**	**11.75**
	WordAttractionRank	38.55	23.55	29.24	30.83	19.79	24.11	4.09	1.96	2.65
	EmbedRank d2v	**41.49**	**25.40**	**31.51**	30.87	19.66	24.02	3.88	1.68	2.35
	EmbedRank s2v	39.63	23.98	29.88	**34.84**	**22.26**	**27.16**	5.53	2.44	3.39
	EmbedRank++ s2v ($\lambda = 0.5$)	37.44	22.28	27.94	24.75	16.20	19.58	2.78	1.24	1.72
	EmbedRank$_{positional}$ s2v	38.84	23.77	29.49	**39.53**	**25.23**	**30.80**	15.07	7.80	10.28
10	TextRank	22.99	11.44	15.28	13.93	16.83	15.24	6.54	6.59	6.56
	SingleRank	34.29	39.04	36.51	24.74	30.97	27.51	5.22	5.04	5.13
	TopicRank	27.43	30.8	29.02	21.49	27.26	24.04	**13.68**	**13.94**	**13.81**
	WordAttractionRank	34.10	38.94	36.36	25.06	31.41	27.88	5.15	5.12	5.14
	EmbedRank d2v	**35.75**	**40.40**	**37.94**	25.38	31.53	28.12	3.95	3.28	3.58
	EmbedRank s2v	34.97	39.49	37.09	**28.82**	**35.58**	**31.85**	5.69	5.18	5.42
	EmbedRank++ s2v ($\lambda = 0.5$)	30.31	34.29	32.18	18.27	23.34	20.50	1.91	1.69	1.79
	EmbedRank$_{positional}$ s2v	32.46	36.61	34.41	**32.23**	**39.95**	**35.68**	13.50	13.36	13.43
15	TextRank	22.80	11.50	15.29	11.25	19.21	14.19	6.14	9.16	7.35
	SingleRank	30.91	48.92	37.88	21.20	38.77	27.41	5.42	8.24	6.54
	TopicRank	24.51	37.45	29.62	17.78	32.92	23.09	**11.04**	**16.47**	**13.22**
	WordAttractionRank	30.74	48.62	37.66	21.82	40.05	28.25	5.11	7.41	6.05
	EmbedRank d2v	31.06	48.80	37.96	22.37	40.48	28.82	4.33	5.89	4.99
	EmbedRank s2v	**31.48**	**49.23**	**38.40**	**24.49**	**44.20**	**31.52**	5.34	7.06	6.08
	EmbedRank++ s2v ($\lambda = 0.5$)	27.24	43.25	33.43	14.86	27.64	19.33	1.59	2.06	1.80
	EmbedRank$_{positional}$ s2v	29.44	46.25	35.98	**27.38**	**49.73**	**35.31**	**12.27**	**17.63**	**14.47**

Table 2: Comparison of our method with state of the art on the three datasets. Precision (P), Recall (R), and F-score (F_1) at 5, 10, 15 are reported. Two variations of EmbedRank with $\lambda = 1$ are presented: s2v uses Sent2Vec embeddings, while d2v uses Doc2Vec.

sake of consistency with previous work on keyphrase extraction [Mihalcea and Tarau, 2004; Hasan and Ng, 2010; Bougouin *et al.*, 2013; Wan and Xiao, 2008], we evaluated our methods on the test part of dataset (500 documents).

DUC 2001 consists of 308 medium length newspaper articles from TREC-9. The documents originate from several newspapers and are organized in 30 topics. Wan and Xiao [Wan and Xiao, 2008] created the dataset, including manual annotations. For keyphrase extraction, we used exclusively the text contained in the first <TEXT> tags of the original documents (we do not use titles and other metadata).

Nguyen [Nguyen and Kan, 2007] consists of 211 long documents (full scientific conference papers), of between 4 and 12 pages. Each document has several sets of keyphrases: one created by the authors and, potentially, several others created by annotators. Following Hasan and Ng [Hasan and Ng, 2010], we evaluate on the union of all sets of assigned keyphrases (author and annotator(s)). We discarded two documents with no assigned keyphrases. This dataset is similar to another one widely used for keyphrase extraction: SemEval. Since our results on SemEval are very similar to Nguyen, we leave them out due to space constraints.

As shown in Table 1, not all assigned keyphrases are present in the documents (missing kp in doc). It is thus impossible to achieve a recall of 100%. We show in the next subsection that our method beats the state of the art on short scientific documents and clearly outperforms it on medium length news articles. On long scientific documents, TopicRank [Bougouin *et al.*, 2013] performs better. We hypothesize that this is due to the use of positional bias: i.e. giving higher relevance to phrases appearing earlier in the document. We confirm this hypothesis by showing that EmbedRank with positional bias matches TopicRank's performance.

5.2 Performance Comparison and Discussion

We compare EmbedRank s2v and d2v (no diversity) to four state-of-the-art, corpus-independent methods[5]: TextRank [Mihalcea and Tarau, 2004], SingleRank [Wan and Xiao, 2008], WordAttractionRank [Rui Wang, Wei Liu, 2015], and TopicRank[6] [Bougouin *et al.*, 2013].

For TextRank and SingleRank, we set the window size to 2 and to 10 respectively, i.e. the values used in the respective papers. We used the same PoS tagged text for all methods. For both underlying d2v and s2v document embedding methods, we use their standard settings as described in Section 3.

We followed the common practice to stem - with the Porter Stemmer [Porter, 1980] - the extracted and assigned keyphrases when computing the number of true positives. As shown in Table 2, EmbedRank performs significantly better than the state of the art on two of the three datasets in terms of precision, recall, and Macro F_1 score. In the context of typical Web-oriented use cases, most data comes as either very

[5]TextRank, SingleRank, WordAttractionRank were implemented using the graph-tool library `https://graph-tool.skewed.de`. We reset the co-occurence window on new sentence.

[6]`https://github.com/boudinfl/pke`

short documents (e.g. tweets) or medium ones (e.g. news articles). The expected performance for Web applications is thus closer to the one observed on the Inspec and DUC2001 datasets, rather than on Nguyen.

However, on long documents, TopicRank outperforms all other methods. The most plausible explanation is that, when selecting an exemplar from the top selected clusters, TopicRank takes the one that appears first in the document. As the original paper points out, using this feature leads to an important gain on long documents. Not using it can lead to a 90% relative drop in F-score. We add this feature to EmbedRank by multiplying the distance of a candidate to the document by the normalized offset position of the candidate. We thus confirm the "positional bias" hypothesis, with EmbedRank$_{positional}$ matching the TopicRank scores.

The results also show that the choice of document embeddings has a high impact on the keyphrase quality. Compared to EmbedRank d2v, EmbedRank s2v is significantly better for DUC2001 and Nguyen, regardless of how many phrases are extracted. On Inspec however, changing the embeddings from doc2vec to sent2vec made almost no difference. A possible explanation is that, given the small size of the original text, the extracted keyphrases have a high likelihood of being single words, thus removing the advantage of having better embeddings for word groups. In all other cases, the results show a clear superiority of Sent2Vec over Doc2Vec in terms of accuracy, adding to the already existing practical advantage of improved inference speed for very large datasets.

5.3 Keyphrase Diversity and Human Preference

In this section, we add EmbedRank++ to the evaluation using the same three datasets. We fixed λ to 0.5 in the adapted MMR equation (2), to ensure equal importance to informativeness and diversity. As shown in Figure 1b, EmbedRank++ reduces the redundancy we faced with EmbedRank. However, EmbedRank++ surprisingly results in a decrease of the F-score, as shown in Table 2.

To investigate this more in depth, we conducted a user study where we asked people to choose between two sets of extracted keyphrases: one generated with EmbedRank ($\lambda = 1$) and another with EmbedRank++ ($\lambda = 0.5$). We set N to the number of assigned keyphrases for each document. During the study, we provided the annotators with the original text, and ask them to choose between the two sets.

For this user study, we randomly selected 20 documents from the Inspec and 20 documents from the DUC2001 dataset, collected 214 binary user preference votes. The long scientific papers (Nguyen) were included in the study, as the full papers were considered too long and too difficult for non-experts to comprehend and summarize.

As shown in Figure 2, users largely prefer the keyphrase extracted with EmbedRank++ ($\lambda = 0.5$). This is a major finding, as it is in contradiction with the F-scores given in Table 2. If the result is confirmed by future tests, it casts a shadow on using solely F-score as an evaluation measure for keyphrase quality. A similar issue was shown to be present in Information Retrieval test collections [Tonon *et al.*, 2015], and calls for research on new evaluation methodologies. We acknowledge that the presented study is a preliminary one and does not support a strong claim about the usefulness of the F-score for the given problem. It does however show that

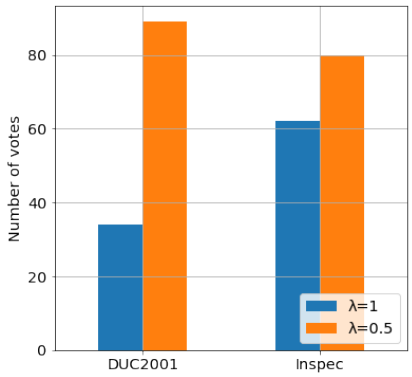


Figure 2: User study among 20 documents from Inspec and 20 documents from DUC2001. Users were asked to choose their preferred set of keyphrases between the one extracted with EmbedRank++ ($\lambda = 0.5$) and the one extracted with EmbedRank ($\lambda = 1$).

people dislike redundancy in summaries and that the $\lambda < 1$ parameter in EmbedRank is a promising way of reducing it.

Our intuition behind this novel result is that the EmbedRank method ($\lambda = 1$), as well as WordAttractionRank, SingleRank and TextRank can suffer from an accumulation of a lot of redundant keyphrases in which a true positive is present. By restricting the redundancy with EmbedRank++, we might select a keyphrase that is not present in the gold keyphrases, but expresses exactly the same idea. By doing so, with the current F-score evaluation, we are penalized as if we had chosen a totally unrelated keyphrase.

6 Conclusion

In this paper we presented EmbedRank and EmbedRank++, two simple and scalable methods for keyphrase extraction from a single document, that leverage sentence embeddings. Both methods are entirely unsupervised, corpus-independent, and they only require the current document itself, rather than the entire corpus to which it belongs (that might not exist at all). They both depart from traditional methods for keyphrase extraction based on graph representations of the input text, and fully embrace sentence embeddings and their ability to model the concepts of informativeness and diversity.

EmbedRank can be implemented on top of any underlying document embeddings, provided that these embeddings can encode documents of arbitrary length. We compared the results obtained with Doc2Vec and Sent2Vec, the latter one being much faster at inference time, which is important in a Web-scale setting. We showed that on short and medium length documents, EmbedRank based on Sent2Vec consistently improves the state of the art, while match it on long documents, by employing an intuitive artifice.

Additionally, thanks to a fairly large user study that we run, we showed that users appreciate diversity of keyphrases, and we raised questions on the reliability of evaluations of keyphrase extraction systems based on F-score.

References

[Bougouin *et al.*, 2013] Adrien Bougouin, Florian Boudin, and Béatrice Daille. TopicRank : Graph-Based Topic Ranking for Keyphrase Extraction. *Proc. IJCNLP 2013*, (October):543–551, 2013.

[Drosou and Pitoura, 2010] Marina Drosou and Evaggelia Pitoura. Search result diversification. *SIGMOD Rec.*, 39(1):41–47, September 2010.

[Florescu and Caragea, 2017] Corina Florescu and Cornelia Caragea. A position-biased pagerank algorithm for keyphrase extraction. In *AAAI Student Abstracts*, pages 4923–4924, 2017.

[Goldstein, 1998] Jade Goldstein. The Use of MMR , Diversity-Based Reranking for Reordering Documents and Producing Summaries. pages 335–336, 1998.

[Hasan and Ng, 2010] Kazi Saidul Hasan and Vincent Ng. Conundrums in Unsupervised Keyphrase Extraction : Making Sense of the State-of-the-Art. 2010.

[Hasan and Ng, 2011] Kazi Saidul Hasan and Vincent Ng. Automatic Keyphrase Extraction: A Survey of the State of the Art. *Association for Computational Linguistics Conference (ACL)*, pages 1262–1273, 2011.

[Hulth, 2003] Anette Hulth. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, EMNLP '03, pages 216–223, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[Kiros *et al.*, 2015] Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. Skip-Thought Vectors. (786):1–11, 2015.

[Lau and Baldwin, 2016] Jey Han Lau and Timothy Baldwin. An empirical evaluation of doc2vec with practical insights into document embedding generation. *ACL 2016*, page 78, 2016.

[Le *et al.*, 2014] Quoc Le, Tomas Mikolov, and Tmikolov Google Com. Distributed Representations of Sentences and Documents. *ICML*, 32, 2014.

[Liu *et al.*, 2009] Zhiyuan Liu, Peng Li, Yabin Zheng, and Maosong Sun. Clustering to Find Exemplar Terms for Keyphrase Extraction. *Language*, 1:257–266, 2009.

[Mihalcea and Tarau, 2004] Rada Mihalcea and Paul Tarau. TextRank: Bringing Order into Texts. *Proceedings of EMNLP*, 85:404–411, 2004.

[Mikolov *et al.*, 2013] Tomas Mikolov, Greg Corrado, Kai Chen, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. pages 1–12, 2013.

[Mori and Sasaki, 2003] Tatsunori Mori and Takuro Sasaki. Information Gain Ratio meets Maximal Marginal Relevance. 2003.

[Nguyen and Kan, 2007] Thuy Dung Nguyen and Min-yen Kan. Keyphrase Extraction in Scientific Publications. 2007.

[Page, 1998] L Page. The PageRank Citation Ranking: Bringing Order to the Web. 1998.

[Pagliardini *et al.*, 2017] Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. Unsupervised Learning of Sentence Embeddings using Compositional n-Gram Features. 2017.

[Pennington *et al.*, 2014] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543, 2014.

[Porter, 1980] Martin F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[Rui Wang, Wei Liu, 2015] Chris McDonald Rui Wang, Wei Liu. Corpus-independent Generic Keyphrase Extraction Using Word Embedding Vectors. 2015.

[Tonon *et al.*, 2015] Alberto Tonon, Gianluca Demartini, and Philippe Cudré-Mauroux. Pooling-based continuous evaluation of information retrieval systems. *Inf. Retr. Journal*, 18(5):445–472, 2015.

[Wan and Xiao, 2008] Xiaojun Wan and Jianguo Xiao. Single Document Keyphrase Extraction Using Neighborhood Knowledge. pages 855–860, 2008.

A Appendix

A.1 Implementation of the Graph-Based methods

Concerning TopicRank [Bougouin *et al.*, 2013] we used the publicly available package made by the author which is available on Github[7].

The other graph-based methods (TextRank [Mihalcea and Tarau, 2004], SingleRank [Wan and Xiao, 2008], WordAttractionRank [Rui Wang, Wei Liu, 2015]) were implemented using the graph-tool[8] library in order to build the graph and perform the PageRank [Page, 1998] computations. For these three methods the co-occurence window was reset when reaching a new sentence.

A.2 WordAttraction

Word Embeddings. We used pre-trained GloVe [Pennington *et al.*, 2014] embeddings trained on Wikipedia 2014 + Gigaword 5 (300-dimensional)[9].

Unknown Words. In the case of edges for which one or both words are not present in the pre-trained embeddings, we set the attraction score to 1. The remaining weighting scheme is composed only of the dice coefficient. Alternatively, discarding these edges resulted in a significantly worse performance on all three datasets.

[7] `https://github.com/boudinfl/pke`

[8] https://graph-tool.skewed.de

[9] https://nlp.stanford.edu/projects/glove/