Back to the Basics: A Quantitative Analysis of Statistical and Graph-Based Term Weighting Schemes for Keyword Extraction

Asahi Ushio and **Federico Liberatore** and **Jose Camacho-Collados**
School of Computer Science and Informatics
Cardiff University, United Kingdom
{UshioA,LiberatoreF,CamachoColladosJ}@cardiff.ac.uk

Abstract

Term weighting schemes are widely used in Natural Language Processing and Information Retrieval. In particular, term weighting is the basis for keyword extraction. However, there are relatively few evaluation studies that shed light about the strengths and shortcomings of each weighting scheme. In fact, in most cases researchers and practitioners resort to the well-known tf-idf as default, despite the existence of other suitable alternatives, including graph-based models. In this paper, we perform an exhaustive and large-scale empirical comparison of both statistical and graph-based term weighting methods in the context of keyword extraction. Our analysis reveals some interesting findings such as the advantages of the less-known lexical specificity with respect to tf-idf, or the qualitative differences between statistical and graph-based methods. Finally, based on our findings we discuss and devise some suggestions for practitioners.[1]

1 Introduction

Keyword extraction has been an essential task in many scientific fields as a first step to extract relevant terms from text corpora. Despite the simplicity of the task, it still poses practical problems, and often researchers resort to simple but reliable techniques such as tf-idf (Jones, 1972). In turn, term weighting schemes such as tf-idf paved the way for developing large-scale Information Retrieval (IR) systems (Ramos et al., 2003; Wu et al., 2008). Its simple formulation is still widely used nowadays, not only for keyword extraction but also as an important component in IR (Jabri et al., 2018; Marcos-Pablos and García-Peñalvo, 2020) and Natural Language Processing (NLP) tasks (Riedel et al., 2017; Arroyo-Fernández et al., 2019).

While there exist supervised and neural techniques (Lahiri et al., 2017; Xiong et al., 2019; Sun et al., 2020), as well as ensembles of unsupervised methods (Campos et al., 2020; Tang et al., 2020) that can provide competitive performance, in this paper we go back to the basics and analyze in detail the single components of unsupervised methods for keyword extraction. In fact, it is still common to rely on unsupervised methods for keyword extraction given their versatility and the lack of training sets in specialized domains.

In order to fill this gap, in this paper we perform an extensive analysis of single unsupervised keyword extraction techniques in a wide range of settings and datasets. To the best of our knowledge, this is the first large-scale empirical evaluation performed across base statistical and graphical keyword extraction methods. Our analysis sheds light on some properties of statistical methods largely unknown. For instance, our experiments show that a statistical weighting scheme based on the hypergeometric distribution such as lexical specificity (Lafon, 1980) can perform at least as well as or better than tf-idf (Jones, 1972), while having additional advantages with respect to flexibility and efficiency. As for the graph-based methods, they can be more reliable than statistical methods without being considerably slower in practice. In fact, graph-based methods initialized with tf-idf or lexical specificity performs best overall.

2 Keyword Extraction

Given a document with m words $[w_1 \cdots w_m]$, keyword extraction is a task to find n noun phrases, which can comprehensively represent the document. As each of such phrases consists of contiguous words in the document, the task can be seen as an ordinary ranking problem over all candidate phrases appeared in the document. A typical keyword extraction pipeline is thus implemented as, first, to construct a set of candidate phrases $\mathcal{P}_d$

[1] Source code to reproduce our experimental results, including a keyword extraction library, are available in the following repository: https://github.com/asahi417/kex

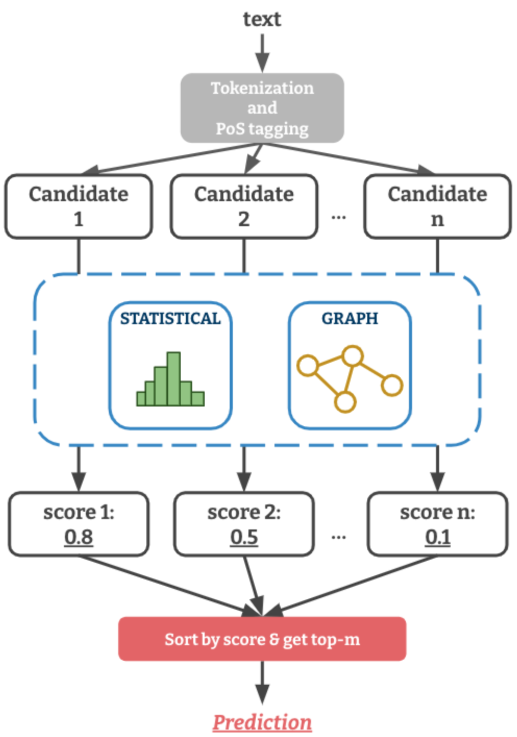


Figure 1: Overview of the keyword extraction pipeline.

for a target document d and, second, to compute importance scores for all of individual words in d.[2] Finally, the top-n phrases $\{y_j|j = 1 \ldots n\} \subset \mathcal{P}_d$ in terms of the aggregated score are selected as the prediction (Mihalcea and Tarau, 2004). Figure 1 shows an overview of the overarching methodology for unsupervised keyword extraction.

To compute word-level scores, there are mainly two types of approach: statistical and graph-based. There are also contributions that focus on training supervised models for keyword extraction (Witten et al., 2005; Liu et al., 2010). However, due to the absence of large labeled data and domain-specificity, most efforts are still unsupervised, which is the focus of this paper.

2.1 Statistical Models

A statistical model attains an importance score based on word-level statistics or surface features, such as the word frequency or the length of word.[3] A simple keyword extraction method could be to simply use term frequency (tf) as a scoring function for each word, which tend to work reasonably well. However, this simple measure may miss important information such as the relative importance of a given word in a corpus. For instance, prepositions such as *in* or articles such as *the* tend to be highly frequent in a text corpus. However, they barely represent a keyword in a given text document. To this end, different variants have been proposed, which we summarize in two main alternatives: tf-idf (Section 2.1.1) and *lexical specificity* (Section 2.1.2).

2.1.1 TF-IDF

As an extension of tf, term frequency–inverse document frequency (tf-idf) (Jones, 1972) is one of most popular and effective methods used for statistical keyword extraction (El-Beltagy and Rafea, 2009), as well as still being an important component in modern information retrieval applications (Marcos-Pablos and García-Peñalvo, 2020; Guu et al., 2020).

Given a set of documents $\mathcal{D}$ and a word w from a document $d \in \mathcal{D}$, tf-idf is defined as the proportion between its word frequency and its inverse document frequency[4], as

$$s_{\mathit{tfidf}}(w|d) = \mathit{tf}(w|d) \cdot \log_2 \frac{|\mathcal{D}|}{\mathit{df}(w|\mathcal{D})} \quad (1)$$

where we define $|\cdot|$ as the number of elements in a set, $\mathit{tf}(w|d)$ as a frequency of w in d, and $\mathit{df}(w|\mathcal{D})$ as a document frequency of w over a dataset $\mathcal{D}$. In practice, $\mathit{tf}(w|d)$ is often computed by counting the number of times that w occurs in d, while $\mathit{df}(w|\mathcal{D})$ by the number of documents in $\mathcal{D}$ that contain w.

To give a few examples of statistical models based on tf-idf and its derivatives in a keyword extraction context, KP-miner (El-Beltagy and Rafea, 2009) utilizes tf-idf, a word length, and the absolute position of a word in a document to determine the importance score, while RAKE (Rose et al., 2010) uses the term degree, the number of different word it co-occurs with, divided by tf. Recently, YAKE (Campos et al., 2020) established strong baselines on public datasets by combining various statistical features including casing, sentence position, term/sentence-frequency, and term-dispersion. In this paper, however, we focus on the vanilla implementation of term frequency and tf-idf.

[2]In the case of multi-token candidate phrases, this score is averaged among its tokens.

[3]The term *Statistical* may not be strictly accurate to refer to tf-idf or purely frequency-based models, but in this case we follow previous conventions by grouping all these methods based on word-level frequency statistics as *statistical* (Aizawa, 2003).

[4]While there are other formulations and normalization techniques for tf-idf (Paik, 2013), in this paper we focus on the traditional inverse-document frequency formulation.

2.1.2 Lexical specificity

Lexical specificity (Lafon, 1980) is a statistical metric to extract relevant words from a subcorpus using a larger corpus as reference. In short, lexical specificity extracts a set of most representative words for a given text based on the hypergeometric distribution. The hypergeometric distribution represents the discrete probability of k successes in n draws, without replacement. In the case of lexical specificity, k represents the word frequency and n the size of a corpus. While not as widely adoped as tf-idf, lexical specificity has been used in similar term extraction tasks (Drouin, 2003), but also in textual data analysis (Lebart et al., 1998), domain-based disambiguation (Billami et al., 2014), or as a weighting scheme for building vector representations for concepts and entities (Camacho-Collados et al., 2016) or sense embeddings (Scarlini et al., 2020) in NLP.

Formally, the lexical specificity for a word w in a document d is defined as

$$s_{spec}(w|d) = -\log_{10} \sum_{l=f}^{F} P_{hg}(x{=}l, m_d, M, f, F) \tag{2}$$

where m_d is the total number of words in d and $P_{hg}(x{=}l, m, M, f, F)$ represents the probability of a given word to appear l times exactly in d according to the hypergeometric distribution parameterised with m_d, M, f, and F, which are defined as below.

$$M = \sum_{d \in \mathcal{D}} m_d,\ f = tf(w|d),\ F = \sum_{d \in \mathcal{D}} tf(w|d) \tag{3}$$

Note also that, unlike in tf-idf, for lexical specificity a perfect partition of documents of $\mathcal{D}$ (reference corpus) is not required. This also opens up to other possibilities, such as using larger corpora as reference, for example.

2.2 Graph-based Methods

The basic idea behind graph-based methods is to identify the most relevant words from a graph constructed from a text document, where words are nodes and their connections are measured in different ways (Beliga et al., 2015). For this, PageRank (Page et al., 1999) and its derivatives have proved to be highly successful (Mihalcea and Tarau, 2004; Wan and Xiao, 2008a; Florescu and Caragea, 2017; Sterckx et al., 2015; Bougouin et al., 2013).

Formally, let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph where $\mathcal{V}$ and $\mathcal{E}$ are its associated set of vertices and edges. In a typical word graph construction on a document d (Mihalcea and Tarau, 2004), $\mathcal{V}$ is defined as the set of all unique words in d and each edge $e_{w_i,w_j} \in \mathcal{E}$ represents a strength of the connection between two words $w_i, w_j \in \mathcal{V}$. Then, a Markov chain from w_j to w_i on a word graph can be defined as

$$p(w_i|w_j) = (1-\lambda)\frac{e_{w_i,w_j}}{\sum_{w_k \in \mathcal{V}_i} e_{w_i,w_k}} + \lambda p_b(w_i) \tag{4}$$

where $\mathcal{V}_i \subset \mathcal{V}$ is a set of incoming nodes to w_i, $p_b(\cdot)$ is a prior probabilistic distribution over $\mathcal{V}$, and $0 \leq \lambda \leq 1$ is a parameter to control the effect of $p_b(\cdot)$. This probabilistic model (4) is commonly known as the random surfer model (Page et al., 1999). The prior term $p_b(\cdot)$, which is originally a uniform distribution, is introduced to enable any transitions even if there are no direct connections among them. Once a word graph is built, PageRank is applied to estimate a probability $\hat{p}(w)$ for every word $w \in \mathcal{V}$, which is used as an importance score.

TextRank (Mihalcea and Tarau, 2004) uses an undirected graph and defines the edge weight as $e_{w_i,w_j} = 1$ if w_i and w_j co-occurred within l contiguous sequence of words in d, otherwise $e_{w_i,w_j} = 0$. SingleRank (Wan and Xiao, 2008a) extends TextRank by modifying the edge weight as the number of co-occurrence of w_i and w_j within the l-length sliding window and ExpandRank (Wan and Xiao, 2008b) multiplies the weight by cosine similarity of tf-idf vector within neighbouring documents. To reflect a statistical prior knowledge to the estimation, recent works proposed to use non-uniform distributions for $p_b(\cdot)$. Florescu and Caragea (2017) observed that keywords are likely to occur very close to the first few sentences in a document in academic paper and proposed PositionRank in which $p_b(\cdot)$ is defined as the inverse of the absolute position of each word in a document. TopicalPageRank (TPR) (Jardine and Teufel, 2014; Sterckx et al., 2015) introduces a topic distribution inferred by Latent Dirichlet Allocation (LDA) as a $p_b(\cdot)$, so that the estimation contains more semantic diversity across topics. TopicRank (Bougouin et al., 2013) clusters the candidates before running PageRank to group similar words together, and MultipartiteRank (Boudin, 2018) extends it by employing a multipartite graph for a better candidate selection within a cluster.

Finally, there are a few other works that directly run graph clustering (Liu et al., 2009; Grineva et al., 2009), using edges to connect clusters instead of

Data	Size	Domain	Type	Divers.	# NPs		# tokens		Vocab size		# keyphrases			
											total		multi-words	
					avg	std	avg	std	avg	std	avg	std	avg	std
KPCrowd	500	-	news	0.44	77	62.0	447	476.7	197	140.6	16.5	12.0	3.7	3.8
Inspec	2000	CS	abstract	0.55	27	12.3	138	66.6	76	28.6	5.8	3.5	4.8	3.2
Krapivin2009	2304	CS	article	0.12	815	252.1	9131	2524.4	1081	256.2	3.8	2.1	2.9	1.9
Nguyen2007	209	-	article	0.15	618	113.9	5931	1023.1	909	142.4	7.2	4.3	4.8	3.2
PubMed	500	BM	article	0.18	566	196.5	4461	1626.4	800	223.7	5.7	2.7	1.5	1.3
Schutz2008	1231	BM	article	0.29	630	287.7	4201	2251.1	1217	468.4	28.5	10.3	10.1	4.9
SemEval2010	243	CS	article	0.13	898	207.7	9740	2443.4	1218	209.1	11.6	3.3	8.8	3.3
SemEval2017	493	-	paragraph	0.54	40	12.9	198	60.3	106	27.5	9.3	4.9	6.3	3.4
citeulike180	183	BI	article	0.21	822	173.0	5521	978.8	1171	202.9	7.8	3.4	1.1	1.0
fao30	30	AG	article	0.21	774	93.2	5438	927.5	1125	157.1	15.9	5.6	5.5	2.6
fao780	779	AG	article	0.19	776	147.2	5591	902.4	1087	210.3	4.2	2.3	1.6	1.3
theses100	100	-	article	0.21	728	131.3	5397	958.4	1134	192.3	2.4	1.5	0.8	0.8
kdd	755	CS	abstract	0.59	16	17.0	82	93.0	48	45.7	0.7	0.9	0.6	0.8
wiki20	20	CS	report	0.15	817	322.4	7146	3609.8	1088	295.4	12.8	3.2	6.7	2.7
www	1330	CS	abstract	0.58	18	16.5	91	89.1	53.0	43.3	0.9	1.0	0.5	0.7

Table 1: Dataset statistics, where size refers to the number of documents; diversity refers to a measure of variety of vocabulary computed as the number of unique words divided by the total number of words; number of noun phrases (NPs) refers to candidate phrases extracted by our pipeline; number of tokens is the size of the dataset; vocab size is the number of unique tokens, and number of keyphrase shows the statistics of gold keyphrases for which we report the total number keyphrases, as well as the number of keyphrases composed by more than one token (multi-tokens). In terms of statistics, we show the average (avg) and the standard deviation (std).

words, with semantic relatedness as a weight. Although these techniques can capture high-level semantics, the relatedness-based weights rely on external resources such as Wikipedia (Grineva et al., 2009), and thus add another layer of complexity in terms of generalization. For these reasons, they are excluded from this study.

3 Experimental Setting

In this section, we explain our keyword extraction experimental setting. All our experiments are run on a 16-core Ubuntu computer equipped with 3.8GHz i7 core and 64GiB memory.[5]

Datasets. To evaluate the keyword extraction methods, we consider 15 different public datasets in English.[6] Each entry in a dataset consists of a source document and a set of gold keyphrases, where the source document is processed through the pipeline described in Section 3 and the gold keyphrase set is filtered to include only phrases which appear in its candidate set. Table 1 provides high-level statistics of each dataset, including length and number of keyphrases[7] (both average and standard deviation).

Preprocessing. Before running keyword extraction on each dataset, we apply standard text preprocessing operations. The documents are first tokenized into words by segtok[8], a python library for tokenization and sentence splitting. Then, each word is stemmed to reduce it to its base form for comparison purpose by Porter Stemmer from NLTK (Bird et al., 2009), a widely used python library for text processing. Part-of-speech annotation is carried out using NLTK tagger. To select a candidate phrase set $\mathcal{P}_d$, following the literature (Wan and Xiao, 2008b), we consider contiguous nouns in the document d that form a noun phrase satisfying the regular expression (ADJECTIVE)*(NOUN)+.[9] We then filter the candidates with a stopword list taken from the official YAKE implementation[10] (Campos et al., 2020). Finally, for the statistical methods and the graph-based methods based on them (i.e., LexRank and TFIDFRank), we compute

[5] All the details to reproduce our experiments are available at https://github.com/asahi417/kex

[6] All the datasets were fetched from a public data repository for keyword extraction data: https://github.com/LIAAD/KeywordExtractor-Datasets: KPCrowd (Marujo et al., 2013), Inspec (Hulth, 2003), Krapivin2009 (Krapivin et al., 2009), SemEval2017 (Augenstein et al., 2017), kdd (Gollapalli and Caragea, 2014), www (Gollapalli and Caragea, 2014), wiki20 (Medelyan and Witten, 2008), PubMed (Schutz et al., 2008), Schutz2008 (Schutz et al., 2008), citeulike180 (Medelyan et al., 2009), fao30 and fao780 (Medelyan and Witten, 2008), guyen2007 (Nguyen and Kan, 2007), and SemEval2010 (Kim et al., 2010).

[7] We use keyword and keyphrase almost indistinctly, as some datasets contain keyphrases of more than a single token.

[8] https://pypi.org/project/segtok/

[9] While the vast majority of keywords in the considered datasets follow this structure, there are a few cases of different Part-of-Speech tags as keywords, or where this simple formulation can miss a correct candidate. Nonetheless, our experimental setting is focused on comparing keyword extraction measures, within the same preprocessing framework.

[10] https://github.com/LIAAD/yake

Metric	Dataset	Statistical				Graph-based						
		FirstN	TF	Lex Spec	TFIDF	Text Rank	Single Rank	Position Rank	Lex Rank	TFIDF Rank	Single TPR	Topic Rank
P@5	KPCrowd	35.8	25.3	**39.0**	**39.0**	30.6	30.5	31.8	32.0	32.1	26.9	37.0
	Inspec	31.0	18.9	31.0	31.5	33.2	**33.8**	32.7	32.9	33.3	30.4	31.3
	Krapivin2009	**16.7**	0.1	8.7	7.6	6.6	9.1	14.3	9.7	9.7	7.4	8.5
	Nguyen2007	17.8	0.2	17.2	15.9	13.1	17.3	**20.6**	18.6	18.6	14.0	13.3
	PubMed	9.8	3.6	7.5	6.7	10.1	**10.6**	10.1	8.9	8.8	9.3	7.8
	Schutz2008	16.9	1.6	39.0	38.9	34.0	36.5	18.3	38.9	39.4	14.5	**46.6**
	SemEval2010	15.1	1.5	14.7	12.9	13.4	17.4	**23.2**	16.8	16.6	12.8	16.5
	SemEval2017	30.1	17.0	45.7	**47.2**	41.5	43.0	40.5	46.0	46.4	34.3	36.5
	citeulike180	6.6	9.5	18.0	15.2	23.0	23.9	20.3	23.2	**24.4**	23.7	16.7
	fao30	17.3	16.0	24.0	20.7	26.0	30.0	24.0	29.3	29.3	**32.7**	24.7
	fao780	9.3	3.2	11.7	10.5	12.4	14.3	13.2	13.2	13.1	**14.5**	12.0
	kdd	11.7	7.0	11.2	11.6	10.6	11.5	11.9	11.6	**11.9**	9.4	10.7
	theses100	5.6	0.9	**10.7**	9.4	6.6	7.8	9.3	10.6	9.1	8.3	8.1
	wiki20	13.0	13.0	17.0	21.0	13.0	19.0	14.0	22.0	**23.0**	19.0	16.0
	www	12.2	8.1	11.9	12.2	10.6	11.2	**12.6**	11.6	11.7	10.2	11.2
	AVG	16.6	8.4	20.5	20.0	19.0	21.1	19.8	21.7	**21.8**	17.8	19.8
MRR	KPCrowd	60.1	45.5	**73.6**	72.4	62.4	61.6	64.0	65.8	65.2	50.2	60.7
	Inspec	57.3	33.0	52.4	52.8	51.4	52.4	57.1	53.3	53.7	50.5	**57.8**
	Krapivin2009	**36.1**	1.3	22.9	21.0	18.1	22.2	31.4	23.6	23.8	19.1	21.8
	Nguyen2007	43.0	2.8	38.1	41.2	30.8	34.6	**43.2**	36.4	37.9	29.8	33.7
	PubMed	23.1	13.3	23.5	21.4	**31.7**	30.5	30.6	26.9	26.3	26.0	19.8
	Schutz2008	24.6	8.6	76.6	**76.7**	68.9	70.9	38.5	75.5	76.3	33.7	67.3
	SemEval2010	**49.7**	4.5	35.8	34.6	32.9	35.5	47.8	35.3	36.4	28.7	35.9
	SemEval2017	52.0	32.7	68.6	**68.7**	61.4	63.5	62.4	67.3	67.2	54.3	63.7
	citeulike180	20.9	23.6	55.5	47.7	58.2	62.6	51.0	63.0	**65.7**	62.5	40.3
	fao30	31.1	38.3	61.8	49.1	60.2	70.0	48.6	66.1	67.0	**74.6**	50.6
	fao780	17.0	8.5	39.0	35.9	36.1	38.6	35.9	**39.5**	38.9	38.4	31.6
	kdd	26.1	13.0	27.0	27.8	24.5	26.5	28.1	27.9	**28.8**	18.3	26.2
	theses100	15.1	3.1	**32.5**	31.6	23.2	26.3	24.9	31.6	31.1	26.1	26.9
	wiki20	27.5	27.7	**52.7**	47.7	40.1	45.7	31.1	52.2	46.5	39.6	35.5
	www	29.7	17.1	30.5	**30.6**	26.5	27.6	30.4	29.2	30.1	21.7	27.9
	AVG	34.2	18.2	46.0	44.0	41.8	44.6	41.7	46.2	**46.3**	38.2	40.0

Table 2: Mean precision at top 5 (P@5) and mean reciprocal rank (MRR). The best score in each dataset is highlighted using a bold font.

prior statistics including term frequency (tf), tf-idf, and LDA by Gensim (Řehůřek and Sojka, 2010) within each dataset.

Comparison Models. As **statistical** models, we include keyword extraction methods based on tf, tf-idf, and lexical specificity referred as TF, TFIDF, and LexSpec[11] respectively.[12] Each model uses its statistics as a score for the individual words and then aggregates them to score the candidate phrases (see Section 2.1). We also add a heuristic baseline which takes the first n phrases as its prediction (FirstN). As **graph-based** models, we compare five distinct methods: TextRank (Mihalcea and Tarau, 2004), SingleRank (Wan and Xiao, 2008a), PositionRank (Florescu and Caragea, 2017), SingleTPR (Sterckx et al., 2015), and TopicRank (Bougouin et al., 2013). Additionally, we propose two extensions of SingleRank, which we call TFIDFRank and LexRank, where a word distribution computed by tf-idf or lexical specificity is used for $p_b(\cdot)$. As implementations of graph operations such as PageRank and word graph construction, we use NetworkX (Hagberg et al., 2008), a graph analyzer in Python.

4 Results

In this section, we report our main experimental results comparing unsupervised keyword extraction methods. Table 2 shows the results obtained by all comparison systems.[13] The algorithms in each metric that achieve the best accuracy across datasets are TFIDFRank for P@5, and LexSpec and TFIDF for MRR. In the averaged metrics over all datasets,

[11] For lexical specificity, we follow the implementation of Camacho-Collados et al. (2016).

[12] As mentioned in Section 2.1, we do not include YAKE (Campos et al., 2020) as our experiments are focused on analyzing single features on their own in a unified setting. YAKE utilizes a unified preprocessing and a combination of various textual features, which are out of scope in this paper.

[13] Results are reported according to standard metrics in keyword extraction and IR: precision at 5 (P@5) and mean reciprocal rank (MRR). The appendix includes details about these metrics and results for additional metrics.

Prior	Model	Time prior	Time total	Time per doc
tf	TF	10.2	11.5	0.0058
	LexSpec		12.1	0.0061
	LexRank		25.5	0.0128
tf-idf	TFIDF	10.3	22.4	0.0112
	TFIDFRank		26.5	0.0133
LDA	SingleTPR	16.2	29.4	0.0147
-	FirstN	-	11.5	0.0058
	TextRank		14.9	0.0075
	SingleRank		15.0	0.0075
	PositionRank		15.0	0.0075
	TopicRank		19.0	0.0095

Table 3: Average clock time (sec.) to process the Inspec dataset over 100 independent trials.

lexical specificity and tf-idf based models (TFIDF, LexSpec, TFIDFRank, and LexRank) are shown to perform high in general. In particular, the hybrid models LexRank and TFIDFRank achieve the best accuracy on all the metrics, with LexSpec and TFIDF being competitive in MRR. Overall, despite their simplicity, both lexical specificity and tf-idf appear to be able to exploit effective features for keyword extraction from a variety of datasets, and perform robustly to domain shifts including document size, format, as well as the source domain. Moreover, TF gives a remarkably low accuracy on every metric and the huge gap between TF and TFIDF can be interpreted as the improvement given by the normalization provided by the inverse document frequency. However, as we discuss in Section 6.1, this IDF normalization relies on a corpus partition, which may not be available in all cases. On the other hand, a measure such as lexical specificity only needs overall term frequencies, which may have advantages in practical settings. In the following sections we perform a more in-depth analysis on these results and the global performance of each type model.

Execution time. In terms of efficiency for each algorithm, we report the average process time over 100 independent trials on the Inspec dataset in Table 3, which also includes the time to compute each statistical prior over the dataset. In general, none of the models perform very slowly. Not surprisingly, statistical models are faster than graph-based models due to the overhead introduced by the PageRank algorithm, although as a drawback they need to perform prior statistical computations for each dataset beforehand.

5 Analysis

Following the main results presented in the previous section, we perform an analysis on different aspects of the evaluation. In particular, we focus on the agreement among methods, overall performance (Section 5.1), and the features related to each dataset leading to each method's performace (Section 5.2).

5.1 Mean Precision Analysis

The objective of the following statistical analysis is to compare the overall performance of the keyword extraction methods in terms of their mean performance (i.e., P@5 and MRR). For this analysis, all 117,447 documents are considered individually.

Table 4 illustrates the mean P@5 and MMR for each key extraction method. Across all the metrics, the best results are obtained by TFIDFRank. The differences between the models are tested for statistical significance using paired Wilcoxon rank sum tests.[14] A method is said to dominate another in terms of performance if it is non-worse in all the metrics and strictly better in at least one metric. Following this rule, it is possible to rank the methods according to their dominance order (i.e., the Pareto ranking): the top methods are those that are non-dominated, followed by those that are dominated only by methods of the first group, et cetera. The resulting ranking, which only considers statistically significant differences, is presented in the following: (1) TFIDFRank; (2) LexRank and LexSpec; (3) SingleRank and TFIDF; (4) PositionRank and TopicRank; (5) TextRank; (6) FirstN; (7) SingleTPR; (8) TF.

As can be observed in this ranking and in the results of Table 4, the best method is TFIDFRank, which dominates all the others. LexSpec slightly but consistently outperforms TFIDF, which is an interesting result on its own given the predominance in the use of TFIDF in the literature and in practical applications. We extend the discussion about the comparison of LexSpec and TFIDF in Section 6.1.

5.2 Regression Analysis

The objective of this analysis is to understand what are a dataset's characteristics that make one method better than another at extracting keywords. For this purpose, a regression model is built for every

[14] For the sake of space, the full statistical significance analyses are presented in the appendix. However, these are also commented in our discussion (Section 6).

	Method	P@5	MRR
Statistical	FirstN	18.8	37.1
	TF	7.9	16.1
	LexSpec	20.8	42.9
	TFIDF	20.5	42.2
Graph-based	TextRank	19.5	39.2
	SingleRank	21.0	41.2
	PositionRank	20.0	40.9
	LexRank	21.4	42.9
	TFIDFRank	21.6	43.3
	SingleTPR	16.4	33.2
	TopicRank	21.0	40.3

Table 4: Key extraction methods' mean P@5 and MRR. Each column is independently colour-coded according to a gradient that goes from green (best/highest value) to red (worst/lowest value).

performance metric (P@5 and MRR) and pair of key extraction methods (*m1* and *m2*). Formally, each observation is a pair in the Cartesian product $(dataset \times method)$ in the regression models. The following independent variables are considered: avg_word and sd_word (i.e., average and standard deviation of the number of tokens in the dataset, representing the length of the documents); avg_vocab and sd_vocab (i.e., average and standard deviation of the number of unique tokens in the dataset, representing the lexical richness of the documents); avg_phrase and sd_phrase (i.e., average and standard deviation of the number of noun phrases in the dataset, representing the number of candidate keywords in the documents); avg_keyword and sd_keyword (i.e., average and standard deviation of the number of gold keyphrases associated to the dataset).[15] The regression models estimate the dependent variable as $\Delta avg\_score = avg\_score_{m1} - avg\_score_{m2}$, where avg_score_{m1} and avg_score_{m2} are the average performance metrics obtained by the methods *m1* and *m2* on the dataset's documents, respectively. Feature selection is carried out by forward stepwise-selection using BIC penalization to remove non-significant variables. Each model considers 15 observations and, overall, 110 regression models are fitted.

Given a regression model, its adjusted coefficient of determination ($\mathrm{adj}R^2$) is used as a measure of its goodness of fit. In fact, an $\mathrm{adj}R^2 > 0.50$ indicates that the independent variables explain most of the differences in performance between the models. The distribution of the $\mathrm{adj}R^2$ obtained by the regression models shows overall good explanatory capabilities: the 0%, 25%, 50%, 75%, and 100% quantiles are 0, 0.6479, 0.7760, 0.8729, and 0.9776, respectively. Thus, ~75% of the models have an $\mathrm{adj}R^2 > 0.65$, and ~50% have $\mathrm{adj}R^2 > 0.78$, suggesting that, in general, the considered dataset's characteristics explain satisfactorily the differences in the results obtained by the key extraction methods. Therefore, the variables can be used to determine what method is more performant for a given dataset. In the rest of the paper, only the models having an $\mathrm{adj}R^2 > 0.50$ and their statistically significant variables (i.e., p-value < 0.05) are considered for interpretation.[16]

The coefficients of the regression models can be used to understand under what circumstances each model is preferable. In fact, a positive coefficient identifies a variable that positively correlates with a greater precision for $m1$, while a negative coefficient corresponds to a variable that positively correlates with a greater precision for $m2$. Table 5 illustrates the significant variables for a selection of regression models. These are used in the following section to draw insights on the methods' preferences in terms of dataset features.

6 Discussion

In this section, we provide a focused comparison among the different types of model, highlighting their main differences, advantages and disadvantages. First, we discuss the two main statistical methods analyzed in this paper, namely LexSpec and TFIDF (Section 6.1). Then, we analyze graphical methods, and in particular SingleRank and TopicRank (Section 6.2). Finally, we provide an overview of the main differences between statistical and graph-based methods (Section 6.3).

[15] It can be noticed that not all the variables from Table 1 have been included in the regression analysis. The reasons for that are detailed in the following. The variable *size* represents the number of documents in the dataset. As this is not a characteristic of the documents comprising the dataset, it has been disregarded (note that the size of each document is indeed included in the analysis, i.e., 'avg_word'). Variables *domain* and *type* are too sparse to be relevant. Finally, the variable *diversity* is computed as $\frac{\text{avg\_vocab}}{\text{avg\_word}}$. Since both terms are already included in the regression model, adding diversity would result in an interdependence among the variables, consequently decreasing the interpretability of the results.

[16] More details about individual regression analyses and the significance of their variables are available in the appendix.

$m1$	metric	statistically significant variables		metric	$m2$
LexSpec	P@5	avg_word (***), sd_vocab (*)	sd_word (***)	P@5	TFIDF
	MRR	avg_phrase (**) avg_keyword (*)	avg_word (**), avg_vocab (**)	MRR	
SingleRank	P@5	avg_phrase (**), sd_phrase (**), avg_keyword (**)	sd_word (**), sd_vocab (***), sd_keyword (*)	P@5	TopicRank
	MRR	avg_phrase (***), avg_keyword (*)	avg_word (***), avg_vocab (**)	MRR	
LexSpec	P@5	sd_word (*), avg_vocab (**), sd_keyword (**)	avg_phrase (**), avg_keyword (**)	P@5	SingleRank
SingleRank	P@5	avg_phrase (*), avg_keyword (*)	sd_word (*), avg_vocab (*), sd_keyword (*)	P@5	TFIDF
	MRR	avg_phrase (*), avg_keyword (*)	sd_word (*), avg_vocab (*), sd_keyword (*)	MRR	

Table 5: Significant variables in the regression models comparing key-extraction methods' performance. Columns $m1$ and $m2$ report the compared methods; columns'metric' shows the performance metric considered; the central columns illustrate the statistically significant variables that positively affect the performance of each model. The significance of the variables is indicated between parenthesis, according to the following scale: 0 '***' 0.001 '**' 0.01 '*' 0.05. Only models having $\text{adj}R^2 > 0.5$ are included, as those are the variables that explain most of the differences in performance between the incumbent models.

6.1 LexSpec vs. TFIDF

According to Table 4, LexSpec and TFIDF have similar average performance, although LexSpec obtains slightly better scores in both metrics. These differences are also statistically significant. As for the Pareto ranking, LexSpec ranks second, while TFIDF ranks third. Therefore, the former should be preferred over the latter performance-wise.

However, TFIDF still performs better than LexSpec in certain datasets (see Table 2). According to Table 5, the choice of the key-extraction method strongly depends on the metric used. For P@5, TFIDF performs better in datasets having a higher variability in the number of words (sd_word), while LexSpec prefers datasets with longer documents (avg_word) and more variability in terms of lexical richness (sd_vocab). For MRR, LexSpec exhibits a very different behaviour, performing significantly better in datasets with high average number of noun phrases (avg_phrase) and high variability in the number of gold keywords (sd_keyword). On the other hand, TFIDF prefers datasets with longer and lexically richer documents (avg_word and avg_vocab).

Broadly speaking, LexSpec and TFIDF have qualitative differences. Being based on the hypergeometric distribution, LexSpec has a statistical nature and probabilities can be directly inferred from it. While TFIDF can also be integrated within a probabilistic framework (Joachims, 1996) or interpreted from an information-theoretic perspective (Aizawa, 2003), it is essentially heuristics-based. In practical terms, LexSpec has the advantage of not requiring a partition into documents unlike the traditional formulation of TFIDF. Moreover, given its statistical nature, LexSpec has been shown to be more robust to different document sizes (Camacho-Collados et al., 2015), as we could also empirically corroborate in the variable correlation analysis in the appendix. On the flip side, TFIDF is generally found to be relatively simple to tune for specific settings (Cui et al., 2014).

6.2 SingleRank vs. TopicRank

In this analysis we compare two qualitatively different graph-based methods, namely Single Rank (a representative of vanilla graph-based methods) and TopicRank, which leverages topic models. The two methods have similar performances in term of P@5; however, SingleRank achieves a much better average MRR score, as illustrated in Table 4. The latter is also statistically significant. For this reason, SingleRank completely dominates TopicRank. This is also reflected in the Pareto ranking, where the former ranks third and the latter fourth. Therefore, in general, SingleRank should be preferred.

The insights drawn from the regression models are summarised in the following. Table 5 shows that the performance of TopicRank depends on the metric used. On the other hand, SingleRank has a more stable set of preferences. However, it is still possible to identify a pattern. In fact, TopicRank is positively influenced by the number of words and the lexical richness of the documents in a dataset (sd_word and sd_vocab for P@5, and avg_word and avg_vocab for MRR), while SingleRank is affected by the number of noun phrases and keyphrases associated to the documents (avg_phrase, sd_phrase, and avg_keyword).

6.3 Statistical vs. Graph-based

When comparing SingleRank versus TFIDF and LexSpec in terms of average performance (see Table 4), it can be seen that SingleRank performs better in terms of average P@5 (albeit only the difference with TFIDF is statistically significant); however, it performs worse than both the statistical methods in terms of average MRR (albeit no difference is statistically significant). Still, SingleRank ranks third (as TFIDF), because it is dominated by LexRank (all differences statistically significant). As LexSpec ranks second, it is recommendable to use this method instead of the other two. On the other hand, this is not a definitive argument in favour of using only statistical methods. In general, statistical methods tend to dominate in MRR over vanilla graph-based techniques. However, the method that achieves the highest scores across all the documents is TFIDFRank, which is graph-based in nature but initialized with TF-IDF. Results suggest that while statistical methods can be reliably used to extract relevant terms when precision is required (reminder that MRR rewards systems extracting the first correct candidate in top ranks), graphical methods can extract a more coherent set of keywords overall thanks to its graph-connectivity measures. This finding should be investigated more in detail in future research.

In terms of dataset features, Table 5 shows that the behaviour of SingleRank is very stable. In fact, across all metrics, SingleRank performs better for datasets with a high average of noun phrases and keyphrases (avg_phrase and avg_keyword). On the other hand, the statistical methods (i.e. TFIDF and LexSpec) achieve better performances on datasets with a high standard deviation for the number of words and keyphrases, and a high average number of unique tokens (sd_word, sd_keyword, and avg_vocab). In conclusion, SingleRank performs better in datasets having a high number of candidate and gold keyphrases, while its performance is hindered in datasets having more lexical richness.

Efficiency and running time. Statistical methods are shown to be faster overall in terms of computation time in our experiments (see Table 3). However, all methods are overall efficient in practical settings, and this factor should not be of especial relevant unless computations need to be done on the fly or on a very large scale. As an advantage of graphical models, these do not require a prior computation over the whole dataset. Therefore, graph-based models could potentially reduce the gap in overall execution time in online learning settings, where new documents are added after the initial computations.

7 Conclusion

In this paper, we have presented a large-scale empirical comparison of unsupervised keyword extraction techniques. Our study was focused on two types of keyword extraction methods, namely statistical relying on frequency-based features, and graph-based exploiting the inter-connectivity of words in a corpus. Our analysis on fifteen diverse keyword extraction datasets revealed various insights with respect to each type of method.

In addition to well-known term weighting schemes such as tf-idf, our comparison includes statistical methods such as lexical specificity, which shows better performance than tf-idf while being significantly less used in the literature. We have also explored various types of graph-based methods based on PageRank and on topic models, with varying conclusions with respect to performance and execution time. Our extensive evaluation and analysis can serve as a reference for future research to understand in detail the advantages and disadvantages of each approach in different settings, both qualitatively and quantitatively.

As future work, we plan to extend this analysis to fathom the extent and characteristics of the interactions of different methods and their complementarity. Moreover, we will extend this empirical comparison to other settings where the methods are used as weighting schemes for NLP and IR applications, and for languages other than English.

Acknowledgements

Jose Camacho-Collados acknowledges support from the UKRI Future Leaders Fellowship scheme.

References

Akiko Aizawa. 2003. An information-theoretic perspective of tf–idf measures. *Information Processing & Management*, 39(1):45–65.

Ignacio Arroyo-Fernández, Carlos-Francisco Méndez-Cruz, Gerardo Sierra, Juan-Manuel Torres-Moreno, and Grigori Sidorov. 2019. Unsupervised sentence representations as word information series: Revisiting tf–idf. *Computer Speech & Language*, 56:107–129.

Isabelle Augenstein, Mrinal Das, Sebastian Riedel, Lakshmi Vikraman, and Andrew McCallum. 2017. SemEval 2017 task 10: ScienceIE - extracting keyphrases and relations from scientific publications. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 546–555, Vancouver, Canada. Association for Computational Linguistics.

Slobodan Beliga, Ana Meštrović, and Sanda Martinčić-Ipšić. 2015. An overview of graph-based keyword extraction methods and approaches. *Journal of information and organizational sciences*, 39(1):1–20.

Mokhtar-Boumeyden Billami, José Camacho-Collados, Evelyne Jacquey, and Laurence Kister. 2014. Annotation sémantique et validation terminologique en texte intégral en SHS. In *Proceedings of TALN*, pages 363–376.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Florian Boudin. 2018. 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, Volume 2 (Short Papers)*, pages 667–672, New Orleans, Louisiana. Association for Computational Linguistics.

Adrien Bougouin, Florian Boudin, and Béatrice Daille. 2013. Topicrank: Graph-based topic ranking for keyphrase extraction. In *International joint conference on natural language processing (IJCNLP)*, pages 543–551.

Jose Camacho-Collados, Claudio Delli Bovi, Luis Espinosa-Anke, Sergio Oramas, Tommaso Pasini, Enrico Santus, Vered Shwartz, Roberto Navigli, and Horacio Saggion. 2018. SemEval-2018 task 9: Hypernym discovery. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 712–724, New Orleans, Louisiana. Association for Computational Linguistics.

José Camacho-Collados, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. NASARI: a novel approach to a semantically-aware representation of items. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 567–577, Denver, Colorado. Association for Computational Linguistics.

José Camacho-Collados, Mohammad Taher Pilehvar, and Roberto Navigli. 2016. Nasari: Integrating explicit knowledge and corpus statistics for a multilingual representation of concepts and entities. *Artificial Intelligence*, 240:36–64.

Ricardo Campos, Vítor Mangaravite, Arian Pasquali, Alipio Jorge, Célia Nunes, and Adam Jatowt. 2020. Yake! keyword extraction from single documents using multiple local features. *Information Sciences*, 509:257–289.

Jia Cui, Jonathan Mamou, Brian Kingsbury, and Bhuvana Ramabhadran. 2014. Automatic keyword selection for keyword search development and tuning. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7839–7843. IEEE.

Patrick Drouin. 2003. Term extraction using non-technical corpora as a point of leverage. *Terminology*, 9(1):99–115.

Samhaa R El-Beltagy and Ahmed Rafea. 2009. Kp-miner: A keyphrase extraction system for english and arabic documents. *Information systems*, 34(1):132–144.

Corina Florescu and Cornelia Caragea. 2017. Position-rank: An unsupervised approach to keyphrase extraction from scholarly documents. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1105–1115.

Sujatha Das Gollapalli and Cornelia Caragea. 2014. Extracting keyphrases from research papers using citation networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28.

Maria Grineva, Maxim Grinev, and Dmitry Lizorkin. 2009. Extracting key terms from noisy and multi-theme documents. In *Proceedings of the 18th international conference on World wide web*, pages 661–670.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *International Conference on Machine Learning*, pages 3929–3938. PMLR.

Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. 2008. Exploring network structure, dynamics, and function using networkx. In *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, Pasadena, CA USA.

Anette Hulth. 2003. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 216–223.

Siham Jabri, Azzeddine Dahbi, Taoufiq Gadi, and Abdelhak Bassir. 2018. Ranking of text documents using tf-idf weighting and association rules mining. In *2018 4th International Conference on Optimization and Applications (ICOA)*, pages 1–6. IEEE.

James Jardine and Simone Teufel. 2014. Topical pagerank: A model of scientific expertise for bibliographic search. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 501–510.

Thorsten Joachims. 1996. A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. Technical report, Carnegie-mellon univ pittsburgh pa dept of computer science.

Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*.

Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. 2010. SemEval-2010 task 5 : Automatic keyphrase extraction from scientific articles. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 21–26, Uppsala, Sweden. Association for Computational Linguistics.

Mikalai Krapivin, Aliaksandr Autaeu, and Maurizio Marchese. 2009. Large dataset for keyphrases extraction.

Pierre Lafon. 1980. Sur la variabilité de la fréquence des formes dans un corpus. *Mots. Les langages du politique*, 1(1):127–165.

Shibamouli Lahiri, Rada Mihalcea, and Po-Hsiang Lai. 2017. Keyword extraction from emails. *Nat. Lang. Eng.*, 23(2):295–317.

Ludovic Lebart, A Salem, and Lisette Berry. 1998. *Exploring textual data*. Kluwer Academic Publishers.

Fei Liu, Feifan Liu, and Yang Liu. 2010. A supervised framework for keyword extraction from meeting transcripts. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(3):538–548.

Zhiyuan Liu, Peng Li, Yabin Zheng, and Maosong Sun. 2009. Clustering to find exemplar terms for keyphrase extraction. In *Proceedings of the 2009 conference on empirical methods in natural language processing*, pages 257–266.

Samuel Marcos-Pablos and Francisco J García-Peñalvo. 2020. Information retrieval methodology for aiding scientific database search. *Soft Computing*, 24(8):5551–5560.

Luis Marujo, Márcio Viveiros, and João Paulo da Silva Neto. 2013. Keyphrase cloud generation of broadcast news. *arXiv preprint arXiv:1306.4606*.

Olena Medelyan, Eibe Frank, and Ian H. Witten. 2009. Human-competitive tagging using automatic keyphrase extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1318–1327, Singapore. Association for Computational Linguistics.

Olena Medelyan and Ian H Witten. 2008. Domain-independent automatic keyphrase indexing with small training sets. *Journal of the American Society for Information Science and Technology*, 59(7):1026–1040.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pages 404–411.

Thuy Dung Nguyen and Min-Yen Kan. 2007. Keyphrase extraction in scientific publications. In *International conference on Asian digital libraries*, pages 317–326. Springer.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab.

Jiaul H Paik. 2013. A novel tf-idf weighting scheme for effective ranking. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 343–352.

Juan Ramos et al. 2003. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 29–48. Citeseer.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA.

Benjamin Riedel, Isabelle Augenstein, Georgios P Spithourakis, and Sebastian Riedel. 2017. A simple but tough-to-beat baseline for the fake news challenge stance detection task. *arXiv preprint arXiv:1707.03264*.

Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. 2010. Automatic keyword extraction from individual documents. *Text Mining: Applications and Theory*, pages 1–20.

Bianca Scarlini, Tommaso Pasini, and Roberto Navigli. 2020. Sensembert: Context-enhanced sense embeddings for multilingual word sense disambiguation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8758–8765.

Alexander Thorsten Schutz et al. 2008. Keyphrase extraction from single documents in the open domain exploiting linguistic and statistical methods.

Lucas Sterckx, Thomas Demeester, Johannes Deleu, and Chris Develder. 2015. Topical word importance for fast keyphrase extraction. In *Proceedings of the 24th International Conference on World Wide Web*, pages 121–122.

Si Sun, Chenyan Xiong, Zhenghao Liu, Zhiyuan Liu, and Jie Bao. 2020. Joint keyphrase chunking and salience ranking with bert. *arXiv preprint arXiv:2004.13639*.

Zhong Tang, Wenqiang Li, and Yan Li. 2020. An improved term weighting scheme for text classification. *Concurrency and Computation: Practice and Experience*, 32(9):e5604.

Xiaojun Wan and Jianguo Xiao. 2008a. Collabrank: towards a collaborative approach to single-document keyphrase extraction. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 969–976.

Xiaojun Wan and Jianguo Xiao. 2008b. Single document keyphrase extraction using neighborhood knowledge. In *AAAI*, volume 8, pages 855–860.

Ian H Witten, Gordon W Paynter, Eibe Frank, Carl Gutwin, and Craig G Nevill-Manning. 2005. Kea: Practical automated keyphrase extraction. In *Design and Usability of Digital Libraries: Case Studies in the Asia Pacific*, pages 129–152. IGI global.

Ho Chung Wu, Robert Wing Pong Luk, Kam Fai Wong, and Kui Lam Kwok. 2008. Interpreting tf-idf term weights as making relevance decisions. *ACM Transactions on Information Systems (TOIS)*, 26(3):1–37.

Lee Xiong, Chuan Hu, Chenyan Xiong, Daniel Campos, and Arnold Overwijk. 2019. Open domain web keyphrase extraction beyond language modeling. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5175–5184, Hong Kong, China. Association for Computational Linguistics.

A Graph-based Models Formula

Supposing that we have computed tf-idf for a given dataset $\mathcal{D}$, the prior distribution for TFIDFRank is defined as

$$p_b(w) = \frac{s_{\mathit{tfidf}}(w|d)}{\sum_{\tilde{w}\in\mathcal{V}} s_{\mathit{tfidf}}(\tilde{w}|d)} \tag{5}$$

for $w \in \mathcal{V}$. Likewise, LexRank relies on the pre-computed *lexical specificity* prior (see Section 2.1.2 of the main paper), which defines the prior distribution for d as

$$p_b(w) = \frac{s_{\mathit{spec}}(w)}{\sum_{\tilde{w}\in\mathcal{V}} s_{\mathit{spec}}(\tilde{w})}. \tag{6}$$

All remaining specifications follow SingleRank's graph construction procedure.

B Evaluation Metrics

To evaluate the keyword extraction models, we employ standard metrics in the literature in keyword extraction and information retrieval: precision at k (P@k) and mean reciprocal rank (MRR). In general, precision at k is computed as

$$\text{P@k} = \sum_{d=1}^{|\mathcal{D}|} \frac{|y_d \cap \hat{y}_d^k|}{\min\{|y_d|, k\}} \tag{7}$$

where y_d is the set of gold keyphrases provided with a document d in the dataset $\mathcal{D}$ and $\hat{y}_d^k$ is a set of estimated top-k keyphrases from a model for the document. The minimum operation between the number of gold keyphrases and gold labels in the denominator of Eq. 7 is included as to provide a measure between 0 and 1, given the varying number of gold labels. This formulation follows previous retrieval tasks with similar settings such as SemEval 2018 (Camacho-Collados et al., 2018).

MRR measures the ranking quality given by a model as follows:

$$\text{MRR} = \frac{1}{|\mathcal{D}|} \sum_{d=1}^{|\mathcal{D}|} \frac{1}{\min\left\{k \mid |\hat{y}_d^k \cap y_d| \geq 1\right\}} \tag{8}$$

In this case, MRR takes into account the position of the first correct keyword from the ranked list of predictions.

C Additional Results (P@10)

In addition to the metrics used in the main paper (i.e., P@5 and MRR), in Table 6 we show the main results for precision at 10 (P@10).

D Agreement Analysis

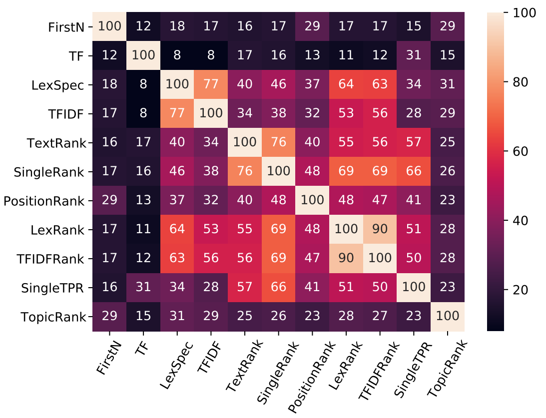
For a visualization purpose, we compute agreement scores over all possible pairs of models as the percentage of predicted keywords the two models have in common in the top-5 prediction, as displayed in Table 7. Interestingly, the most similar models in terms of the agreement score are TFIDFRank and LexRank. Not surprisingly, TFIDF and LexSpec also hold a very high similarity that implies those two statistical measures capture quite close features. However, they also have a few marked differences. Moreover, we can see that graph-based models provide fairly high agreement scores, except for TopicRank, which can be due to the difference in the word graph construction procedure. In fact, TopicRank unifies similar word before building a word graph and that results in such a distinct behaviour among graph-based models. In the discussion section, we investigate the relation among each model in more detail.

E Correlation Analysis

Tables 8 and 9 illustrate the correlation between algorithms and regression variables for P@5 and MMR, respectively. The results of this analysis can be interpreted in the following way: a model with

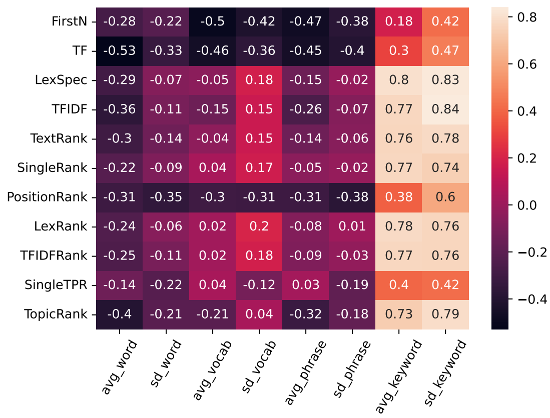
Metric	Dataset	Statistical				Graph-based						
		FirstN	TF	Lex Spec	TFIDF	Text Rank	Single Rank	Position Rank	Lex Rank	TFIDF Rank	Single TPR	Topic Rank
P@10	KPCrowd	**33.1**	23.7	31.8	32.0	24.6	25.0	26.9	26.8	27.0	23.5	32.6
	Inspec	27.4	21.0	29.7	30.3	32.4	**32.6**	31.6	31.9	32.3	29.1	26.1
	Krapivin2009	**16.0**	0.1	8.4	7.1	6.5	9.0	13.6	9.4	9.3	7.3	8.1
	Nguyen2007	15.0	0.5	13.4	12.3	11.5	13.8	**16.9**	15.2	15.0	11.9	12.4
	PubMed	**8.1**	3.5	5.2	4.9	7.2	7.4	7.6	6.2	6.0	6.6	6.9
	Schutz2008	14.4	4.2	28.3	28.0	25.5	27.3	17.2	28.4	28.8	13.2	**41.1**
	SemEval2010	12.0	0.9	11.0	10.5	11.0	13.8	**17.3**	13.0	13.5	10.9	14.4
	SemEval2017	28.3	20.2	40.7	**41.5**	39.1	39.6	38.2	41.0	41.3	34.0	29.9
	citeulike180	5.6	7.7	11.8	10.4	14.4	**15.9**	14.8	14.8	15.8	15.7	12.2
	fao30	15.3	12.7	15.3	13.7	20.3	22.3	19.0	20.3	21.0	**23.7**	18.0
	fao780	7.5	3.2	9.3	8.1	10.2	11.4	11.1	10.8	10.7	**11.8**	9.9
	kdd	11.2	6.7	10.9	11.4	10.2	11.0	**11.5**	11.2	11.5	9.0	10.4
	theses100	4.6	0.7	8.8	7.3	5.5	6.7	8.5	**9.3**	8.0	7.1	6.5
	wiki20	13.0	9.5	12.0	12.0	12.0	15.0	12.5	14.0	15.0	**16.5**	16.0
	www	11.3	7.5	11.1	11.4	9.9	10.5	**11.8**	10.9	10.9	9.6	10.3
	AVG	14.9	8.1	16.5	16.1	16.0	17.4	17.2	17.5	**17.7**	15.3	17.0

Table 6: Mean precision at top 10 (P@10). The best score in each dataset is highlighted using a bold font.



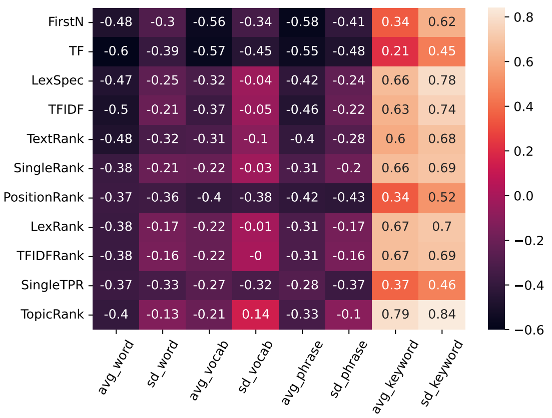
	FirstN	TF	LexSpec	TFIDF	TextRank	SingleRank	PositionRank	LexRank	TFIDFRank	SingleTPR	TopicRank
FirstN	100	12	18	17	16	17	29	17	17	15	29
TF	12	100	8	8	17	16	13	11	12	31	15
LexSpec	18	8	100	77	40	46	37	64	63	34	31
TFIDF	17	8	77	100	34	38	32	53	56	28	29
TextRank	16	17	40	34	100	76	40	55	56	57	25
SingleRank	17	16	46	38	76	100	48	69	69	66	26
PositionRank	29	13	37	32	40	48	100	48	47	41	23
LexRank	17	11	64	53	55	69	48	100	90	51	28
TFIDFRank	17	12	63	56	56	69	47	90	100	50	28
SingleTPR	16	31	34	28	57	66	41	51	50	100	23
TopicRank	29	15	31	29	25	26	23	28	27	23	100

Table 7: Overall pairwise agreement scores for the top 5 predictions.



	avg_word	sd_word	avg_vocab	sd_vocab	avg_phrase	sd_phrase	avg_keyword	sd_keyword
FirstN	-0.28	-0.22	-0.5	-0.42	-0.47	-0.38	0.18	0.42
TF	-0.53	-0.33	-0.46	-0.36	-0.45	-0.4	0.3	0.47
LexSpec	-0.29	-0.07	-0.05	0.18	-0.15	-0.02	0.8	0.83
TFIDF	-0.36	-0.11	-0.15	0.15	-0.26	-0.07	0.77	0.84
TextRank	-0.3	-0.14	-0.04	0.15	-0.14	-0.06	0.76	0.78
SingleRank	-0.22	-0.09	0.04	0.17	-0.05	-0.02	0.77	0.74
PositionRank	-0.31	-0.35	-0.3	-0.31	-0.31	-0.38	0.38	0.6
LexRank	-0.24	-0.06	0.02	0.2	-0.08	0.01	0.78	0.76
TFIDFRank	-0.25	-0.11	0.02	0.18	-0.09	-0.03	0.77	0.76
SingleTPR	-0.14	-0.22	0.04	-0.12	0.03	-0.19	0.4	0.42
TopicRank	-0.4	-0.21	-0.21	0.04	-0.32	-0.18	0.73	0.79

Table 9: Correlations between algorithms and regression variables for metric MRR.



	avg_word	sd_word	avg_vocab	sd_vocab	avg_phrase	sd_phrase	avg_keyword	sd_keyword
FirstN	-0.48	-0.3	-0.56	-0.34	-0.58	-0.41	0.34	0.62
TF	-0.6	-0.39	-0.57	-0.45	-0.55	-0.48	0.21	0.45
LexSpec	-0.47	-0.25	-0.32	-0.04	-0.42	-0.24	0.66	0.78
TFIDF	-0.5	-0.21	-0.37	-0.05	-0.46	-0.22	0.63	0.74
TextRank	-0.48	-0.32	-0.31	-0.1	-0.4	-0.28	0.6	0.68
SingleRank	-0.38	-0.21	-0.22	-0.03	-0.31	-0.2	0.66	0.69
PositionRank	-0.37	-0.36	-0.4	-0.38	-0.42	-0.43	0.34	0.52
LexRank	-0.38	-0.17	-0.22	-0.01	-0.31	-0.17	0.67	0.7
TFIDFRank	-0.38	-0.16	-0.22	-0	-0.31	-0.16	0.67	0.69
SingleTPR	-0.37	-0.33	-0.27	-0.32	-0.28	-0.37	0.37	0.46
TopicRank	-0.4	-0.13	-0.21	0.14	-0.33	-0.1	0.79	0.84

Table 8: Correlations between algorithms and regression variables for metric P@5.

a correlation close to zero can be said to be more robust towards the corresponding variable (i.e., less affected by changes in the variable's value) than another model with a higher absolute correlation. Overall, the scores of all the algorithms are higher for datasets with a high average number of gold keyphrases, while they are lower for datasets with a higher average number of tokens, unique tokens (thus, lexical richness), and candidate keywords. The standard deviations follow the same behavior as the averages.

F Statistical Significance

The performances of the algorithms on both P@5 and MRR have been tested to verify if they are statistically significant. As the data is not normally distributed ($p - values \sim 0$ in Anderson-Darling normality tests), paired Wilcoxon rank sum tests have been used. The results are illustrated in Table

10.

G Regression Models

This section provides additional information regarding the regression models. In the following Tables 11 - 17, the regression models for the comparisons considered in Section "Discussion" are presented. For each variable, the tables show the estimated coefficient value, the standard error, the t-value, and the p-value. The last column identifies the significance of the coefficient, according to the following scale: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1. The adjusted coefficient of determination ($\mathrm{adj}R^2$) is provided in the caption. Note that only the models having $\mathrm{adj}R^2 > 0.5$ are reported.

	FirstN	LexSpec	TF	TFIDF	TFIDFRank	TextRank	SingleRank	PositionRank	LexRank	SingleTPR	TopicRank
FirstN	-	0.00	0.00	0.00	0.00	0.07	0.00	0.00	0.00	0.00	0.00
LexSpec	0.00	-	0.00	0.00	0.00	0.00	0.24	0.03	0.00	0.00	0.61
TF	0.00	0.00	-	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
TFIDF	0.00	0.00	0.00	-	0.00	0.00	0.00	0.69	0.00	0.00	0.02
TFIDFRank	0.00	0.00	0.00	0.00	-	0.00	0.00	0.00	0.07	0.00	0.00
TextRank	0.00	0.00	0.00	0.00	0.00	-	0.00	0.00	0.00	0.00	0.00
SingleRank	0.00	0.27	0.00	0.40	0.00	0.00	-	0.00	0.00	0.00	0.37
PositionRank	0.00	0.10	0.00	0.98	0.00	0.00	0.10	-	0.00	0.00	0.06
LexRank	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-	0.00	0.01
SingleTPR	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-	0.00
TopicRank	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.08	0.00	0.00	-

Table 10: P-values of paired Wilcoxon rank sum tests on the P@5 score (upper triangular matrix) and MMR (lower triangular matrix) obtained by the algorithms on all the documents considered. A value below 0.05 indicates that the difference between the algorithms is statistically significant.

	Estimate	Std. Error	t value	Pr(>|t|)	
(Intercept)	-0.7718	0.4430	-1.7424	0.1093	
avg_word	0.0007	0.0001	6.2946	0.0001	***
sd_word	-0.0029	0.0005	-6.2922	0.0001	***
sd_vocab	0.0093	0.0032	2.8805	0.0150	*

Table 11: LexSpec VS TFIDF; metric: P@5; $\mathrm{adj}R^2 = 0.76$.

	Estimate	Std. Error	t value	Pr(>|t|)	
(Intercept)	-0.9170	1.2298	-0.7457	0.4730	
avg_word	-0.0027	0.0007	-3.9583	0.0027	**
avg_vocab	-0.0383	0.0106	-3.6158	0.0047	**
avg_phrase	0.0800	0.0179	4.4689	0.0012	**
avg_keyword	0.3064	0.1213	2.5251	0.0301	*

Table 12: LexSpec VS TFIDF; metric: MRR; $\mathrm{adj}R^2 = 0.64$.

	Estimate	Std. Error	t value	Pr(>|t|)	
(Intercept)	4.5423	1.0236	4.4377	0.0022	**
sd_word	-0.0096	0.0021	-4.4760	0.0021	**
sd_vocab	-0.1277	0.0187	-6.8144	0.0001	***
avg_phrase	0.0072	0.0021	3.4006	0.0094	**
sd_phrase	0.1810	0.0375	4.8281	0.0013	**
avg_keyword	0.6920	0.1726	4.0101	0.0039	**
sd_keyword	-0.9239	0.3274	-2.8218	0.0224	*

Table 13: SingleRank VS TopicRank; metric: P@5; $\mathrm{adj}R^2 = 0.88$.

	Estimate	Std. Error	t value	Pr(>|t|)	
(Intercept)	-1.8764	2.0988	-0.8940	0.3923	
avg_word	-0.0061	0.0012	-5.2350	0.0004	***
avg_vocab	-0.0638	0.0181	-3.5287	0.0055	**
avg_phrase	0.1503	0.0306	4.9186	0.0006	***
avg_keyword	0.4663	0.2071	2.2518	0.0480	*

Table 14: SingleRank VS TopicRank; metric: MRR; $\mathrm{adj}R^2 = 0.74$.

	Estimate	Std. Error	t value	Pr(>|t|)	
(Intercept)	-1.9469	1.1930	-1.6320	0.1413	
sd_word	0.0073	0.0023	3.1395	0.0138	*
avg_vocab	0.0397	0.0107	3.6991	0.0061	**
avg_phrase	-0.0575	0.0141	-4.0726	0.0036	**
sd_phrase	-0.0518	0.0253	-2.0479	0.0748	
avg_keyword	-1.0153	0.2214	-4.5867	0.0018	**
sd_keyword	1.9318	0.3985	4.8471	0.0013	**

Table 15: LexSpec VS SingleRank; metric: P@5; $\mathrm{adj}R^2 = 0.76$.

	Estimate	Std. Error	t value	Pr(>|t|)	
(Intercept)	0.9891	1.7134	0.5773	0.5796	
sd_word	-0.0088	0.0034	-2.6226	0.0305	*
avg_vocab	-0.0388	0.0154	-2.5141	0.0361	*
avg_phrase	0.0628	0.0203	3.0966	0.0147	*
sd_phrase	0.0476	0.0363	1.3103	0.2265	
avg_keyword	0.9859	0.3179	3.1009	0.0146	*
sd_keyword	-1.7774	0.5724	-3.1051	0.0146	*

Table 16: SingleRank VS TFIDF; metric: P@5; $\mathrm{adj}R^2 = 0.68$.

	Estimate	Std. Error	t value	Pr(>|t|)	
(Intercept)	0.1791	3.6510	0.0491	0.9621	
sd_word	-0.0164	0.0049	-3.3190	0.0106	*
avg_vocab	-0.1326	0.0462	-2.8690	0.0209	*
sd_vocab	0.1004	0.0544	1.8446	0.1023	
avg_phrase	0.1921	0.0600	3.2029	0.0126	*
avg_keyword	2.1376	0.6667	3.2062	0.0125	*
sd_keyword	-3.4630	1.2615	-2.7451	0.0253	*

Table 17: SingleRank VS TFIDF; metric: MRR; $\mathrm{adj}R^2 = 0.56$.