Information Sciences 509 (2020) 257–289

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

YAKE! Keyword extraction from single documents using multiple local features

Ricardo Campos [a,e,*], Vítor Mangaravite [e,g], Arian Pasquali [e], Alípio Jorge [b,e], Célia Nunes [c,f], Adam Jatowt [d]

[a] Ci2 - Smart Cities Research Center, Polytechnic Institute of Tomar, Tomar, Portugal
[b] FCUP, University of Porto, Porto, Portugal
[c] Department of Mathematics, University of Beira Interior, Covilhã, Portugal
[d] Graduate School of Informatics, Kyoto University, Kyoto, Japan
[e] LIAAD - INESC TEC - INESC Technology and Science, Porto, Portugal
[f] Center of Mathematics and Applications, University of Beira Interior, Covilhã, Portugal
[g] Federal University of Minas Gerais, Minas Gerais, Brazil

ARTICLE INFO

Article history:
Received 13 November 2018
Revised 3 September 2019
Accepted 9 September 2019
Available online 11 September 2019

Keywords:
Keyword extraction
Information extraction
Unsupervised Algorithm

ABSTRACT

As the amount of generated information grows, reading and summarizing texts of large collections turns into a challenging task. Many documents do not come with descriptive terms, thus requiring humans to generate keywords on-the-fly. The need to automate this kind of task demands the development of keyword extraction systems with the ability to automatically identify keywords within the text. One approach is to resort to machine-learning algorithms. These, however, depend on large annotated text corpora, which are not always available. An alternative solution is to consider an unsupervised approach. In this article, we describe YAKE!, a light-weight unsupervised automatic keyword extraction method which rests on statistical text features extracted from single documents to select the most relevant keywords of a text. Our system does not need to be trained on a particular set of documents, nor does it depend on dictionaries, external corpora, text size, language, or domain. To demonstrate the merits and significance of YAKE!, we compare it against ten state-of-the-art unsupervised approaches and one supervised method. Experimental results carried out on top of twenty datasets show that YAKE! significantly outperforms other unsupervised methods on texts of different sizes, languages, and domains.

1. Introduction

The number of documents published on the Internet is extremely large reaching approximately 1 billion websites (source: www.internetworldstats.com). Such a huge amount of information makes the process of indexing and finding documents a challenging and difficult task. The vast majority of documents are also not associated with any keywords (aka keyphrases), which forces users to read an entire document to get an overall idea of its contents. The process of manually creating descriptive terms for texts can also quickly turn into a distressful and time-consuming task for those who need to tag their documents. With so many documents available, manual keyword extraction is no longer feasible. To automate this process,

* Corresponding author.
E-mail addresses: ricardo.campos@ipt.pt (R. Campos), mangaravite@dcc.ufmg.br (V. Mangaravite), arianpasquali@gmail.com (A. Pasquali), amjorge@fc.up.pt (A. Jorge), celian@ubi.pt (C. Nunes), jatowt@gmail.com (A. Jatowt).

https://doi.org/10.1016/j.ins.2019.09.013
0020-0255/

Table 1
Running example: document #1001 of Inspec collection. Gold Keywords (ground-truth) are given at the bottom of the table.

A conflict between language and atomistic information Fred Dretske and Jerry Fodor are responsible for popularizing three well-known theses in contemporary philosophy of mind: the thesis of Information-Based Semantics (IBS), the thesis of Content Atomism (Atomism) and the thesis of the Language of Thought (LOT). LOT concerns the semantically relevant structure of representations involved in cognitive states such as beliefs and desires. It maintains that all such representations must have syntactic structures mirroring the structure of their contents. IBS is a thesis about the nature of the relations that connect cognitive representations and their parts to their contents (semantic relations). It holds that these relations supervene solely on relations of the kind that support information content, perhaps with some help from logical principles of combination. Atomism is a thesis about the nature of the content of simple symbols. It holds that each substantive simple symbol possesses its content independently of all other symbols in the representational system. I argue that Dretske's and Fodor's theories are false and that their falsehood results from a conflict IBS and Atomism, on the one hand, and LOT, on the other
Gold Keywords: philosophy of mind, information-based semantics, content atomism, ibs, language of thought, lot, cognitive states, beliefs, desires, artificial intelligence, cognitive systems, philosophical aspects

keyword extraction programs are frequently used to capture the fundamental idea of a document and create or select keywords. Typically, these make available a shortlist of five to ten keywords (with one or more terms[1]), thus instantaneously providing users and machines with a summary of the document. Keyword extraction may be of interest to a myriad of people who need to become acquainted with a given topic. For instance, a journalist looking for information on the *Donald Trump* election would likely be glad to be offered a summary consisting of keywords, a timeline, or an automatically created narrative [23] to explore and digest retrieved information. It would also be of benefit to librarians and readers who wish to have a quick glimpse of a given article, for Internet users looking for the most relevant keywords of a web page, or even for search engines seeking to improve their process of indexing, retrieval, or results presentation. Table 1 presents a running example of the keyword extraction task for document #1001 of the Inspec collection. The bottom row of the table shows the ground-truth keywords (aka gold keywords[2]). This document will guide us through the rest of this article, being used in further illustrative examples.

Although considerable research has been devoted to this topic over the years, the problem of extracting relevant keywords with high accuracy remains unsolved [20]. Several factors contribute to this. Among these, the difficulties of defining relevance and the language diversity itself (e.g., texts of different languages, sizes, and domains) are among the most important challenges. Other issues involve the problem posed by absent keywords,[3] the exact match restriction,[4] and the high number of candidate keywords that can be generated from a single text.[5] All these issues highlight the difficulties of developing a global solution that performs equally in all scenarios, and motivate the need for further research.

Among existing research, supervised approaches have established themselves as the de-facto standard methodology for extracting keywords from documents, making use of discriminant features and of machine learning algorithms to learn a model that distinguishes between relevant and non-relevant keywords. While machine learning algorithms offer some advantages, they require a rather long training process and demand large manually annotated collections of documents to capture the nature of a language. This contrasts with general unsupervised algorithms, which may be quickly applied to a document across different languages or domains at reduced effort due to their plug and play nature. In this research article, we follow an unsupervised approach by proposing YAKE!, a lightweight alternative method to supervised machine learning techniques. To identify important keywords from individual unstructured documents, we rely on local text features and statistical information, such as term co-occurrence and frequencies. The system we propose is largely robust to language or domain diversity and can easily scale to vast collections of documents and contexts. The fact that it relies on individual documents makes it possible to operate independent of the existence of a corpus. In an era of vast but often unlabeled collections, this can be a great advantage over other approaches in cases where access to a training corpus is either limited or restricted. This offers an attractive solution for a plethora of situations, including companies which have just started their businesses and therefore lack an available corpus (known as the cold-start problem), or for those operating in a specialized domain (thus with limited access to a training collection). In such cases, a keyword extraction algorithm that functions without a corpus will always be advantageous.

Extracting keywords without using a training set or an external corpus, such as a controlled vocabulary, poses some challenges, however. The fundamental difficulty lies in determining a concise list of descriptive terms solely based on the document itself. Several factors contribute to this difficulty, including the intricacies of the natural language and document length and structure. To tackle this problem, we adopt a twofold approach: (1) first, we employ local statistic features that extract informative content within the text to calculate the importance of single terms; (2) second, we apply a special *n*-

[1] We use the term keyword to refer interchangeably to both single keywords (e.g., "info") and multi-word keywords (e.g., "information retrieval").
[2] The term gold keywords is used interchangeably with ground-truth keywords to signify the set of expected keywords (manually assigned by human annotators) to be extracted from a text.
[3] I.e., gold keywords manually assigned by the editors that cannot be found in the text.
[4] I.e., gold keywords that need to be the same as the extracted keywords, which end up impeding automated systems to obtain 100% recall.
[5] Thus making it difficult to position the most important ones at the top.

gram construction model to form multi-word terms and employ a heuristic measure to determine their relevance. While the features may be heuristic, they are not arbitrarily chosen and are founded on language principles, keeping the method rather simple yet still able to achieve better results than state-of-the-art methods.

Our contributions can be summarized as follows:

1. *Unsupervised approach*: we propose a light-weight unsupervised automatic keyword extraction algorithm which builds upon local text statistical features extracted from single documents; i.e., it does not require any training corpus.
2. *Corpus-Independent*: we offer a solution which can retrieve keywords from a single document only, without the need to rely on external document collection statistics as IDF does; i.e., it can be applied to any text.
3. *Domain and Language-Independent*: YAKE! works with domains and languages for which there are no ready keyword extraction systems, as it neither requires a training corpus nor depends on sophisticated external sources (such as WordNet or Wikipedia) or linguistic tools (such as NER or PoS taggers) other than a static list of stopwords.
4. *Interior Stopwords*: YAKE! can retrieve keywords containing *interior* stopwords (e.g., "game *of* Thrones") with higher precision than the state-of-the-art methods.
5. *Scale*: YAKE! scales to any document length linearly in the number of candidate terms identified.
6. *Term Frequency-free*: meaning that no conditions are set with respect to the minimum frequency or sentence frequency that a candidate keyword must have. Therefore, based on the features used, a keyword may be considered significant or insignificant with either one occurrence or with multiple occurrences.
7. *Open-Source*: finally, we make available a demo [yake.inesctec.pt] [9] and an app on Google Play, as well as an API [yake.inesctec.pt/api] and a python package [github.com/LIAAD/yake], so that the scientific community can test our approach and evaluate it in the future in subsequent studies.

To understand and illustrate the differences between our system and the state-of-the-art methods, we compare it against ten unsupervised approaches and one supervised method. The unsupervised approaches include three statistical methods (TF.IDF [22], KP-Miner [13], and RAKE [42]) and seven graph-based methods (TextRank [37], SingleRank [48], ExpandRank [48], TopicRank [4], TopicalPageRank [46], PositionRank [15], and MultipartiteRank [6]). The supervised method is KEA [50]. We carry out our tests using twenty different public document collections [github.com/LIAAD/KeywordExtractor-Datasets]: *110-PT-BN-KP* [32], *500N-KPCrowd-v1.1* [31], *Inspec* [21], *Krapivin2009* [28], *Nguyen2007* [38], *PubMed, Schutz2008* [44], *SemEval2010* [27], *SemEval2017* [1], *cacic* [2], *citeulike180* [34], *fao30* [35], *fao780* [35], *pak2018, theses100, wicc* [2], *wiki20* [33], *WWW* [16], *KDD* [16], and WikiNews. The collections are in 5 different languages: English, French, Spanish, Portuguese, and Polish). To the best of our knowledge this is the most comprehensive experimental evaluation ever performed in terms of state-of-the-art methods and datasets used.

This article is an extended version of Campos et al. [10] (Best Short Paper Award at ECIR'18 conference), which briefly introduced our approach and preliminary evaluation results. In comparison with that paper, this article additionally does the following:

- Gives a high-level overview of the related research on keyword extraction;
- Extends, details, and explains the reasoning for each of the steps of our methodology;
- Makes use of two additional distance similarity measures (Jaro Winkler and Sequence Matcher) to discard potential similar candidate keywords;
- Evaluates the effectiveness of YAKE! under various settings (through a fine-tuning process that enables determination of the best deduplication and window parameters);
- Makes use of sixteen additional text collections - *110-PT-BN-KP, Inspec, Krapivin2009, Nguyen2007, PubMed, SemEval2017, cacic, citeulike180, fao30, fao780, pak2018, theses100, wiki20, WWW, KDD*, and WikiNews - to further corroborate that the effectiveness of our approach is not limited to a single dataset, domain, or language;
- Makes available a new dataset for the Polish language;
- Uses a random oversampling technique to evaluate YAKE! and state-of-the-art methods from a single unified view;
- Extends the experimental part of our project with an entirely new set of comparative experiments under seven additional state-of-the-art baseline approaches (Expand Rank, TopicRank, TopicalPageRank, PositionRank, MultipartiteRank, and KEA);
- Makes use of further metrics, including MAP@X, F1@X, P@X, R@X, R-Precision, Mean Reciprocal Rank, and Precision Recall curves, to better interpret the results;
- Provides an evaluation of the effectiveness of YAKE! and baseline methods when dealing with interior stopwords (which has never been done before); and
- Evaluates feature importance in order to understand the functioning of our approach at a deeper level.

The work here presented is part of a comprehensive research project [7,8] whose aim is to enable users to effectively search for results in documents containing dates. Our system rests on an approach that automatically extracts the most representative keywords (obtained with the method explained below) and dates of a document. Documents are then presented through clustering and a ranking structure. Further to this, we have applied YAKE! in "Tell me Stories"[6] [39], an interac-

[6] 1st prize (among 27 participants) of the Arquivo.pt 2018 contest.

tive system [archive.tellmestories.pt; demo.tellmestories.pt] that automatically generates timeline summarizations of news articles maintained by the Portuguese Web Archive [arquivo.pt] [18].

The remainder of this article is structured as follows. Section 2 offers a comprehensive overview of related research. Section 3 defines the architecture of YAKE!. Section 4 describes the experimental setup. Section 5 discusses the obtained results. Section 6 provides a detailed analysis on feature importance. Finally, Section 7 summarizes the article and concludes with some final remarks.

2. Keyword extraction

The problem of extracting relevant keywords from documents is longstanding [43] and solutions have proven to be of immense value to a myriad of tasks, scenarios, and players, including text summarization, clustering, thesaurus building, opinion mining, categorization, query expansion, recommendation, information visualization, retrieval, indexing, and digital libraries. Typically, keyword solutions fall into one of two broad approaches: *keyword assignment* and *keyword extraction*. Keyword assignment is a multi-label text classification task which assigns a set of keywords selected from a controlled vocabulary (dictionary or thesaurus relevant to the domain being discussed) to an instance of data (documents). Keyword extraction, in contrast, involves the extraction of keywords from the documents themselves, following either an unsupervised (see Section 2.1) or a supervised approach (see Section 2.2).

2.1. Unsupervised approaches

In an era where the volume of documents is increasing to a large extent, human labeling of a training set from time to time is a laborious task [25]. To overcome this bottleneck, unsupervised approaches, ranging from statistical methods to graph-based approaches, have been gaining importance over the last few years.

The baseline method in unsupervised approaches is TF.IDF [22], which compares the frequency of a term in a document to its frequency in a large collection. Although quite simple to implement, TF.IDF requires access to a large corpus, which may not always be available. In order to overcome this limitation, a few alternative approaches have been proposed. One of the first was introduced by El-Beltagy and Rafea [13], who proposed KP-Miner, an unsupervised keyword extraction system which relies on TF.IDF measure and two boosting factors (word length and position in the document) to determine the importance of a keyword. More recently, Florescu and Caragea [14] proposed computing the score of a keyword by combining the mean of the scores of the individual terms weighted by the frequency (TF) of the keyword within a document, as a means to reduce the impact of summing up the scores of long keywords. However, the most well-known approaches of this kind, and the closest prior research to our proposal, are the work of Rose et al. [42], who presented RAKE, a domain-independent and language-independent (no PoS) unsupervised single-document approach, and SBKE (Selectivity-Based Keyword Extraction), proposed by Beliga et al. [3]. In particular, RAKE considers sequences of terms (delimited by stopword or phrase delimiters) as candidate keywords, and based on this builds a matrix of term co-occurrences. Term scores are then calculated for each candidate keyword as the sum of its member word scores, based on the degree and frequency of the terms in the matrix: (1) term frequency; (2) term degree (the number of different terms that it co-occurs with), and (3) ratio of degree to frequency. In contrast, SBKE presents a single-document and domain-independent novel network measure, where the term-node selectivity is defined as the average strength of the node based on their co-occurrences.

Other methods, while remaining unsupervised, offer an alternative solution to purely statistical methods by drawing on graphs to represent a document's text. The most well-known approach of this kind is probably the TextRank [37] algorithm. Mihalcea and Tarau build an undirected graph where nodes are terms that respect a certain PoS, and edges (connections) are set between nodes that co-occur within a window of n terms. A ranking algorithm is then run on top of this graph, and keywords are sorted by decreasing order. Variations of TextRank include SingleRank [48] and ExpandRank [48], the latter being enriched by terms of the k-nearest neighboring documents. An extended version of this neighborhood methodology was proposed by Gollapalli and Caragea [16], through their CiteTextRank algorithm, a graph-based methodology for research articles that incorporates evidence from both a document's content and a citation network (short text descriptions surrounding the mention of the paper). Several other approaches have extracted relevant keywords for the major topics of documents. For instance, Bougouin et al. [4] presented TopicRank, an improvement of the TextRank and TextRank-based methods; TopicRank represents documents as a graph, where nodes instead of terms are topic clusters of single and composed expressions. Liu et al. [25] built a TopicalPageRank on a word graph to measure term importance concerning different topics by using Latent Dirichlet Allocation (LDA) learned from a large-scale document collection. In their work, Wikipedia is used as a source to determine term occurrence information. Once the topic distribution of the document is performed, the graph is built. In subsequent research, Grineva et al. [19] extracted candidate keywords using Wikipedia to weigh term importance and created a semantic graph to estimate semantic relatedness between keywords. A graph clustering technique was then used to detect communities in the graph on the assumption that densely interconnected groups of terms typically correspond to the main topics of the document. More recently Florescu and Caragea [15] proposed PositionRank, an unsupervised graph-based model that extracts keywords from scholarly documents (the titles and abstracts of research papers) by incorporating information from all positions of a term's occurrence and its frequency in a document to compute a biased PageRank score for each candidate term. Finally, Boudin [6] proposed an unsupervised keyword extraction model

that encodes topical information within a multipartite graph structure, in which nodes are keyword candidates that are only connected if they belong to different topics.

Stimulated by recent advances in deep learning techniques, a few approaches related to embedding word models have emerged as an alternative methodology for the extraction of keywords. One of the first in this regard was that of Wang et al. [49]. In their work, word scores are defined and computed using the values provided by the word embeddings (pre-trained over Wikipedia) and local statistical information. The computed scores are then assigned to each edge as weights, and a weighted PageRank algorithm is used to compute the final scores of words. Other similar works worth citing have been proposed in this context [30,40].

2.2. Supervised approaches

One of the first approaches to automatically extract keywords from texts as a supervised learning binary task was proposed by Turney [47], who developed a custom-designed algorithm named GenEx that outperforms general-purpose machine learning algorithms, such as decision trees. Arguably the most widespread implementation of one such algorithm is KEA [50], which uses only two features (TF.IDF and the term's first occurrence) and the Naïve Bayes machine learning algorithm to determine what is a keyword and what is not. Another system, proposed by Hulth [21], uses linguistic features, such as PoS tag(s) of a term, as a means to enlarge the knowledge conveyed by statistical properties of the text: term frequency, collection frequency, and position of the first occurrence of a term. All these features are then used as input to a rule induction with a bagging supervised machine learning algorithm. Several other works, such as those of Medelyan and Witten [34] and Nguyen and Kan [38], are based on improved versions of KEA. Caragea et al. [11] try to incorporate information from neighborhoods; in their work, a supervised algorithm named CeKE [16] is proposed which combines information available in citation networks of research papers in which traditional features can be extracted from the paper itself. Alternative approaches include the supervised feature-based models proposed by Meng et al. [36], who use a neural network deep learning model as a way to predict keywords from scientific texts, and by Gollapalli and Li [17], who studied keyword extraction from research papers as a sequence tagging task. A more thorough discussion and in-depth review of research on keyword extraction systems can be found in the studies of Hasan and Ng [20] and Papagiannopoulou and Tsoumakas [41].

2.3. Motivation

Our approach differs from previous research on keyword extraction in several aspects. First, we propose an unsupervised approach which does not require any annotated text corpora or a training stage (as supervised models do). Second, instead of operating on top of a document collection, we rely on single documents, thus making it possible to quickly apply our solution to a myriad of scenarios. Moreover, we do not resort to linguistic tools or external sources (as unsupervised graph-based models do). Instead, in our approach, we detect relevant keywords based on statistics extracted from the documents. This means that we can apply our solution to various different languages or domains in a plug and play nature. Finally, apart from retrieving a list of potentially relevant keywords, we also estimate their degree of relevance. This contrasts with generic binary classifiers. Compared to our previous publication [10], this paper makes several significant new contributions (listed above at the end of Section 1). In the following section, we describe the methodology behind YAKE!.

3. YAKE! algorithm

YAKE!, the algorithm proposed in this paper, has five main steps: (1) *text pre-processing and candidate term identification*; (2) *feature extraction*; (3) *computing term score*; (4) *n-gram generation and computing candidate keyword score*; and (5) *data deduplication* and *ranking*. The first step pre-processes the document into a machine-readable format in order to identify potential candidate terms. This is an important and crucial step to identify better candidate terms and thus to improve the effectiveness of the algorithm. The second phase takes as input a list of individual terms and represents them by a set of statistical features. In the third step, these features are heuristically combined into a single score likely to reflect the importance of the term. The fourth step then generates the candidate keywords (through an *n*-gram[7] construction methodology) and assigns them scores, based on their importance. Finally, the fifth step compares likely similar keywords through the application of a deduplication distance similarity measure. The list of final keywords is then sorted by their relevance scores. The overall flow of our model is shown in Algorithm 1.

The algorithm receives a *text* and the following parameters as inputs: a window size w (to be used by one of the statistical features), the number of n-grams, the deduplication threshold θ, and the text *language* (note that the text language parameter is only needed for identification of the specific list of stopwords). Given a *text*, the algorithm begins by dividing it into sentences (line 2). Each *sentence* is then pre-processed giving rise to a number of pre-processed *terms* (lines 3–5), thus concluding the first step of the algorithm. In the next stage (which entails steps 2 and 3 of our algorithm), each *term* is characterized by a number of statistical features (line 8) and given a score (line 9). *Candidate keywords* are then formed according to a sliding window of size n (lines 11–17) and given a score (lines 18–21). This corresponds to the fourth step of

[7] In our work a n-gram is defined as a contiguous sequence of n-words occurring in the text.

Algorithm 1 Keyword extraction procedure.

```
Input: text, w, n, θ, language
1:  # (Step 1) Text pre-processing and candidate term identification
2:  sentences = split text into sentences
3:  for each sentence in sentences do
4:      Pre-process (sentence, language)
5:  end for
6:  # (Step 2) Feature extraction & (Step 3) Term score
7:  for each term in sentences do
8:      Feature extraction
9:      Compute term weight
10: end for
11: # (Step 4) n-gram generation
12: for each sentence in sentences do
13:     chunks = split sentence into chunks
14:     for each chunk in chunks do
15:         Build n-gram candidateKeywords list
16:     end for
17: end for
18: # (Step 4) Candidate keyword score
19: for each candidate in candidateKeywords do
20:     Compute candidateKeywords weight
21: end for
22: # (Step 5) Data deduplication
23: for each candidate1, candidate2 in candidateKeywords do
24:     if DistanceSimilarity(candidate1, candidate2) > θ:
25:         Remove candidate from candidateKeywords
26: end for
27: # (Step 5) Ranking
28: Keywords = sort (candidateKeywords) by ascending score
Output: (Keywords, score)
```

our algorithm. Finally, *candidate keywords* with a similarity score above a given θ are removed (lines 22–26). The final list of keywords and corresponding scores are sorted according to their relevance (lines 27–28) and returned as output, thus concluding the algorithm. In the following, we describe each of these steps in greater detail.

3.1. Text pre-processing and candidate term identification

The pre-processing stage of the text is the first step before text representation and text analytics take place. Here, the text is cleaned and transformed into a machine-readable format: important chunks are identified, and uninformative and noisy ones are removed. A typical pre-processing procedure involves cleaning the text, sentence splitting, text annotation, tokenization, and stopwords identification. Other procedures may include natural language processing (NLP) techniques, such as part-of-speech tagging (PoS), named-entity recognition (NER), normalization, linguistic parsing, or stemming. These, however, require specialized tools for any different language considered. The flow of our pre-processing model is shown in Algorithm 2 (see annex).

Algorithm 2 Text pre-processing and candidate term identification procedure.

```
Input: text, language

1:  sentences = split text into sentences
2:  for each sentence in sentences do
3:      chunks = split sentence into chunks
4:      for each chunk in chunks do
5:          tokens = split chunk into tokens
6:          for each token in tokens do
7:              tag (token), to_lowercase (token), is_stopword (token, language)
8:          end for
9:      end for
10: end for
Output: List of annotated sentences, chunks
```

Given a *text*, the algorithm begins by dividing it into sentences (line 1). Here we employ the *segtok* rule-based sentence segmenter [pypi.python.org/pypi/segtok], which splits Indo-European texts into sentences following a given predefined pattern. For instance, “Python is awesome! But C# is also very good” would be divided into two sentences (“Python is awesome!”; “But C# is also very good”), while “Mr. Smith” would be considered a single sentence. Each sentence is then divided

Table 2
List of tag delimiters used in the pre-processing stage.

Tag	Name	Description
d	Digit or number	A term formed by digits, including those separated by "," and "."
u	unparsable Content	Any of the following: - A term that is formed by at least two punctuation marks (e.g., www.google.com, email@account.com, <head>); - A term that includes a character that is neither a digit nor a letter (e.g., #:) //(!)); - A term that is formed by a combination of numbers and letters (e.g., 3. #ffd700, Word2Vec); - A term formed by more than one digit and alpha char (e.g., 1e10);
a	Acronyms	A term only formed by uppercase chars
U	Uppercase	A term that begins with an uppercase char, and does not mark the beginning of a sentence
p	Parsable Content	All the remaining content

Table 3
Text pre-processing.

One of the most iconic 3 software's on this matter is the Practical Automatic Keyphrase Extraction (KEA), which was â‰¥ first known on 1999–06–07
Chunk1: <*p*>One <*p*>of <*p*>the <*p*>most <*p*>iconic <*d*>3 <*p*>software <*p*>'s <*p*>on <*p*>this <*p*>matter <*p*>is <*p*>the <*U*>Practical <*U*>Automatic <*U*>Keyphrase <*U*>Extraction
Chunk2: <*a*>KEA
Chunk3: <*p*>which <*p*>was <*p*>â <*u*>‰¥ <*p*>first <*p*>known <*p*>on <*u*>1999–06–07

into *chunks* (whenever punctuation is found) and split into *tokens* (lines 3–5). In order to accomplish this process, we use the *web_tokenizer* module of the *segtok segmenter*. This is an important step as it allows us not only to identify words but also to isolate noisy information such as punctuation, emails, or URLs, thus aiding the identification of messy data.[8] Each *token* (line 6) is then converted into its lowercase form and annotated with tag delimiters (line 7). In addition, we also use a static list of stopwords[9] based on the language given as input, to mark potentially meaningless *tokens* (although in this case without a tag in line 7). Note that *tokens* with fewer than three characters are also considered a stopword in our approach. Table 2 presents a brief description of each tag considered in our methodology.

In order to better illustrate this tagging process, we present in Table 3 an example for a given *text* consisting of just a single *sentence* (first row of the table). The last three rows show the three *chunks* identified within the *sentence* and the corresponding tagged *tokens*. Note that from here onwards, each linguistic unit of the *text* will be considered a candidate 1-gram *term*, rather than a *token* (aka term occurrence).

The result of this pre-processing stage is a list of *sentences* (in the case of the above *text*, a single sentence), where each sentence is divided into *chunks* formed by annotated *terms*. In the next section, we will describe each of the statistical features considered in our work.

3.2. Feature extraction

After the text pre-processing and candidate term identification stage, we employ a statistical analysis which pays particular attention to structure, term frequencies, and co-occurrence. The overall process is shown in Algorithm 3.1.

First, we begin by creating an empty structure, named *terms* (line 1), which will be used to keep all the 1-gram terms found within the text and additional information such as their statistics and scores (to be computed later). We then iterate within the list of *sentences* (line 2) and *chunks* (line 3). Next, we split the *chunks* into annotated *tokens* (line 4) and compute their *TF* (*term* frequency, line 6), *offsets_sentences* (index of sentences where the *terms* occur – line 7), *TF_a* (*term* frequency of acronyms: lines 8–9), and *TF_U* (term frequency of uppercase *terms*: lines 10–11). All these statistics are saved or updated in the *terms* array. In addition to this, a co-occurrence matrix (named *co-occur*) is built to save the co-occurrences between a given *term* and its predecessor (line 13) or subsequent *terms* (line 14) found within a window of a given *w*.

Once the statistics of each term are computed, the feature extraction process can now be conducted. A summary of the features' computation is given in Algorithm 3.2. We devise a set of five features likely to capture the nature of each candidate term, where a candidate term is a *1*-gram token (e.g., "information").[10] These are *TCase* (line 2), *TPos* (line 3), *TFNorm* (lines 4–7), *TRel* (lines 9–13), and *TSent* (line 14). Detailed explanations of each of these features are given below.

3.2.1. Casing (T_{Case})

The casing aspect of a term is an important feature when considering the extraction of keywords. The underlying rationale is that uppercase terms tend to be more relevant than lowercase ones. In our approach, we give extra attention to any term beginning with a capital letter (excluding the beginning of sentences). Furthermore, we also consider the occurrence

[8] E.g., the word "www" would be highly frequent in texts having several URLs, yet that generally doesn't mean it is an important word.
[9] https://github.com/LIAAD/yake/tree/master/yake/StopwordsList.
[10] For the generation of the definitive candidate keywords (terms with 1 or more than 1-gram), see Section 3.4.

Algorithm 3.1 Compute term statistics.

```
Input: sentences, chunks, w
terms = []
for each sentence in sentences do
    for each chunk in chunks do
        tokens = split chunk into tokens
        for i = 0..|tokens| do
            terms[index (tokens[i])].TF++
            terms[index (tokens[i])].offsets_sentences += index(sentence)
            if tokens.tag == 'a':
                terms[index (tokens[i])].TF_a++
            if tokens.tag == 'U':
                terms[index (tokens[i])].TF_U++
            for j = 1..w do
                cooccur [terms[index (tokens[i-j])], terms[index (tokens[i]]]++
                cooccur [terms[index (tokens[i]], terms[index(tokens[i+j])]]++
            end for
        end for
    end for
end for
Output: List of terms and corresponding statistics, cooccur matrix
```

Algorithm 3.2 Features computation.

```
Input: terms, cooccur
for i = 0.. |terms| do
    terms[i].TCase = max (terms [i]).TF_a, terms[i]).TF_U) / (1 + ln (terms[i]).TF))
    terms[i].TPos = ln ((3 + median (terms[i].offsets_sentences)))
    terms[i].TFNorm
        validTFs = [term.TF for term in terms if not term.stopword]
        avgTF = mean (validTFs)
        stdTF = stf (validTFs)
    terms[i].TFNorm = terms[i].TF / (avgTF + stdTF)
    terms[i].TRel
        maxTF = max ([term.TF for term in terms])
        DL = |coocur [[*][terms[i]]| / sum (coocur [[*][terms[i]])
        DR = |coocur [[terms[i] [*]]| / sum (coocur [[terms[i] [*]])
    terms[i].TRel = 1 + (DL + DR) * (terms[i]).TF / maxTF)
    terms[i].TSent = | terms[i].offsets_sentences | / |sentences|
end for
Output: List of terms with corresponding statistics and calculated features
```

of acronyms through a heuristic, where all the letters of the word are capitals. However, instead of counting this as a double weight, we will only consider the maximum occurrence within the two of them. Eq. (1) reflects the casing aspect of a candidate term:

$$T_{Case} = \frac{\max(TF(U(t)),\ TF(A(t)))}{\ln(TF(t))} \tag{1}$$

where TF(U(t)) is the number of occurrences of the candidate term t starting with an uppercase letter, TF(A(t)) is the number of times the candidate term t is marked as an acronym and TF(t) is the frequency of t. Thus, the more often the candidate term occurs with a capital letter, the more important it is considered. This means that a candidate term that occurs with a capital letter ten times within ten occurrences will be given a higher value (4.34) than a candidate term that occurs with a capital letter five times within five occurrences (3.10).

3.2.2. Term position ($T_{Position}$)

Another indicator of the importance of a candidate term is its position [6,21]. The rationale is that relevant keywords tend to appear at the very beginning of a document, whereas words occurring in the middle or at the end of a document tend to be less important. This is particularly evident for both news articles and scientific texts, two kinds of publications which tend to concentrate a high degree of important keywords at the top of the text (e.g., in the introduction or abstract). Like Florescu and Caragea [15], who posit that models that take into account the positions of terms perform better than those that only use the first position or no position at all, we also consider a term's position to be an important feature. However, unlike their model, we do not consider the positions of the terms, but of the sentences in which the terms occur; this, to the best of our knowledge, has never been done before. Our assumption is that terms that occur in the early sentences of a text should be more highly valued than terms that appear later. Thus, instead of considering a uniform distribution of terms, our model assigns higher scores to terms found in early sentences. We calculate this weight using the following equation:

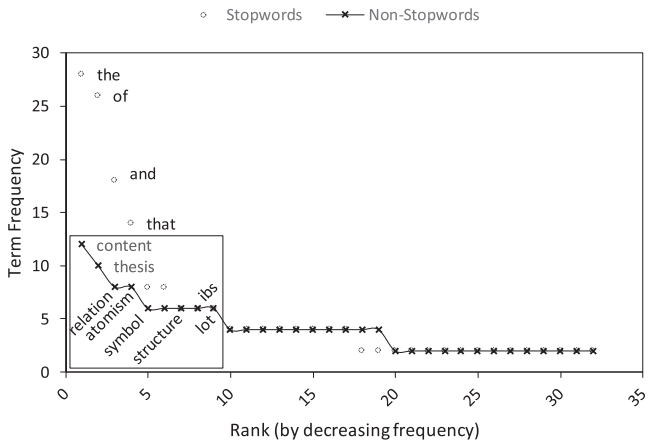


Fig. 1. Rank vs term frequency for stopword (circle marks) and non-stopword (straight line) terms of doc 1001 of Inspec collection.

$$\mathrm{T_{Position}} = ln(\ln(3 + Median(Sen_t))) \tag{2}$$

where Sen_t is the set of positions of the sentences where the candidate term t occurs, and the Median function is the median of Sen_t. In addition, knowing that $Median(Sen_t)$ can return a value of 0 (when the candidate term only appears in the first sentence), a constant $C > e$, in our case 3, is added to the equation (as the first integer to occur after the Euler's number e ($\sim$2.71)) to guarantee that $\mathrm{T_{Position}} > 0$. Also, note that a double log is applied in order to smooth the difference between terms that occur with a large median difference.

Based on this equation, a candidate term t_1 that occurs at the 2nd, 35th, 70th, 74th, and 4000th sentences in a document would have a $\mathrm{T_{Position}}$ value of 1.45 as a result of having $\ln(\ln(3 + Median[2, 35, 70, 74, 4000]))$, while a candidate term t_2 that occurs in the 1st, 4th, and 7th sentences would return a $\mathrm{T_{Position}}$ of 0.66, which stems from having $\ln(\ln(3 + Median[1, 4, 7]))$. The result is an increasing function, whose values tend to increase smoothly as candidate terms are positioned toward the end of the document, meaning that the more a candidate term appears at the beginning of a document, the lower its $\mathrm{T_{Position}}$ value. Conversely, candidate terms more positioned toward the end of a document (likely less relevant) will be given a higher $\mathrm{T_{Position}}$ value.

3.2.3. Term frequency normalization (TF_{Norm})

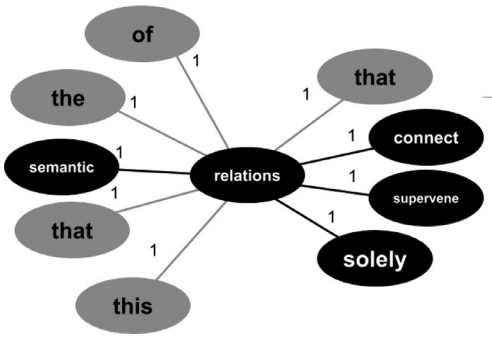
This feature indicates the frequency of the candidate term t within the document based on the work of Luhn [26], who states that "the frequency of word occurrence in an article furnishes a useful measurement of word significance". This reflects the belief that the higher the frequency of a candidate term, the greater its importance. Yet, this does not mean that importance is proportional to the number of times a term occurs. Thus, to prevent a bias towards high frequency in long documents the TF value of a candidate term t is divided by the mean of the frequencies (MeanTF) plus 1 times their standard deviation (σ), as in Eq. (3):

$$\mathrm{TF_{Norm}} = \frac{\mathrm{TF(t)}}{\mathrm{MeanTF} + 1{*}\sigma} \tag{3}$$

Our purpose is to value all candidate terms whose frequencies are above the mean, balanced by a certain degree of dispersion given by the standard deviation. To compute this, we opt to only consider the MeanTF and the standard deviation of non-stopwords, thus guaranteeing that the calculation of these two components is not influenced by the high frequencies usually registered by stopword terms. To illustrate this, we resort to document #1001 of the Inspec collection (initially introduced in Table 1 of Section 1) and compute the mean and standard deviation ($\mathrm{MeanTF} + 1{*}\sigma$) of stopwords and non-stopword terms to show the difference in the values obtained by computing the denominator of the equation using only stopword terms versus computing it using only non-stopword terms. To make matters more concrete, we plot in Fig. 1 the distribution of frequency for a few selected terms (stopwords and non-stopword terms). The black box represents the set of terms whose frequency is above the value 5.32 given by $\mathrm{MeanTF} + 1{*}\sigma$ when non-stopword terms are considered. Dividing the term frequency by this value gives more weight to all those terms that are within the box. On the other hand, if a score of 8.46 is applied (should stopwords be considered), none of these words will obtain a weight higher than 1 (when dividing their TF by the $\mathrm{MeanTF} + 1{*}\sigma$). Although anecdotal, it is interesting to note that of the nine terms that are inside the box, four ("*content*", "*atomism*", "*ibs*" and "*lot*") are part of the ground-truth of this particular document.

3.2.4. Term relatedness to context (T_{Rel})

Although stopwords offer an unquestionably useful source of knowledge of what is clearly not relevant, merely relying on a static list of information may not be sufficient to discard non-relevant words. In this section, we describe a statistical



w=2	w=1	t	w=1	w=2
of	the	**relations**	that	connect
	semantic	**relations**		
that	this	**relations**	supervene	solely

Fig. 2. Co-occurrences of relation within a w size of 2.

feature which aims to determine the dispersion (D) of a candidate term t with regards to its specific context. We rely on the assumption of Machado et al. (2009) [29] who state that the higher the number of different terms that co-occur with the candidate term t on both sides, the less significant term t will be. DL [DR] is formalized in Eq. (4):

$$DL[DR] = \frac{|A_{t,\ w}\ |}{\sum_{k\ \in A_{t,\ w}} CoOccur_{t,\ k}} \tag{4}$$

where $|A_{t,w}|$ represents the number of different terms, where a term is a parsable content, acronym, or uppercase word that occurs on the left [right] side of a candidate term t within a given predefined window of w size (to be defined later in the experimental section), with regard to the number of k terms that it co-occurs with.

DL as well as DR are then multiplied by the term frequency of the candidate term divided by the maximum term frequency (MaxTF) among all candidate terms that occur in the document, in order to penalize candidate terms that occur with high frequency (as stopwords do) and with many different terms on both the left- and right-hand side. The final equation of T_{Rel} is given by Eq. (5):

$$T_{Rel} = 1 + \ (DL + DR.....) * \frac{TF(t)}{MaxTF} \tag{5}$$

where the leftmost part of the equation measures the significance of the candidate term with regard to its left side, and the rightmost part measures its significance with regard to its right side. In practical terms, the less relevant the candidate term t is, the higher the score of this feature will be. Thus, stopwords and, similarly, non-discriminant terms, tend to obtain a higher score. Fig. 2 shows the different terms that co-occur with the candidate term "relations" within a w of 2, where "relations" is an unimportant term (not part of ground-truth, i.e., not a gold keyword) in the context of our running example. The right side of the figure shows the excerpts of the text where the term "relations" appears, while the left side shows a graphical representation. The number on the edges is the frequency that the candidate term "relations" co-occurs with the term in the corresponding oval. Gray color indicates stopwords.

Looking at the figure we can observe that all 5 occurrences with terms on the left are with different terms, thus resulting in a high DL dispersion of 5/5. Similarly, all 4 occurrences of "relation" with terms on the right are with different terms, again producing a high DR dispersion of 4/4. These values are then weighted according to how closely the candidate term frequency approaches the frequency of the max term frequency (which is usually given by stopwords). Next, we discuss the feature Term Different Sentence.

3.2.5. Term different sentence ($T_{Sentence}$)

This feature quantifies how often a candidate term appears within different sentences. It reflects the assumption that candidates which appear in many different sentences have a higher probability of being important. This score is computed using the following equation:

$$T_{Sentence} = \frac{SF(t)}{\#Sentences} \tag{6}$$

where SF(t) is the sentence frequency of the candidate term t, i.e., the number of sentences in which t appears, and #Sentences is the total number of sentences in the text. This results in the range of [0, 1].

All the statistical features discussed so far are then combined into a single term weight that reflects the candidate term's importance. We discuss this in the next section.

3.3. Computing term score

Once the features score is computed, we are now ready to determine the term score. To compute this score, we gather all the feature weights into a unique S(t) score. The smaller the value, the more significant the 1-gram term (t) is. Eq. (7)

formalizes this process:

$$S(t) = \frac{T_{Rel} * T_{Position}}{T_{Case} + \frac{TF_{Norm}}{T_{Rel}} + \frac{T_{Sentence}}{T_{Rel}}} \quad (7)$$

Looking at this equation, we can observe that both TF_{Norm} and $T_{Sentence}$ are divided by T_{Rel}. The idea underlying this mitigation is to assign a high value to 1-gram terms that appear frequently and in many sentences (likely indicative of their importance) as long as they are relevant (have a low T_{Rel}). Indeed, some terms may occur numerous times and in many sentences and yet be useless. These should be penalized. Thus, achieving a high score on TF_{Norm} and $T_{Sentence}$ is an important indication of a term's importance, as long as T_{Rel} has a low value. Likewise, the position of a term in sentences occurring at the top of a document is an important feature that is taken into account by multiplying $T_{Rel}{}^{*}T_{Position}$. Algorithm 4 depicts this process for each single term.

Algorithm 4 Term score procedure.

Input: *terms*
1: **for** $i = 0..$ $|terms|$ **do**
2: $terms[i].S_t = (TPos * TRel) / (TCase + ((TFNorm + TSent) / TRel))$
3: **end for**
Output: List of *terms* with corresponding statistics, calculated features, and term score

To demonstrate the appropriateness of $S(t)$ when weighting single terms, we again resort to document #1001 of the Inspec collection and show 60 single terms that are part of it. Tables 4 and 5 show the top and bottom 30 single terms, respectively, with the corresponding $S(t)$ scores. 1-gram terms that are a keyword (according to the ground-truth) or part of a keyword are printed in boldface. Stopwords are excluded from this analysis as they would dominate the lower part of the table, making it difficult to demonstrate the usefulness of $S(t)$ for devaluing less important terms. Looking at both tables, one can confirm the appropriateness of $S(t)$ when weighting 1-gram terms. Careful examination of Table 4 shows that most of the terms are either marked as relevant or offer some relevance. Although relevance is difficult to define, it is not difficult to agree that a considerable number of terms present in the table are relevant, as is the case of "*jerry*", "*atomistic*", and "*fodor*", to name but a few. Another piece of evidence is that most of the non-relevant terms can be found in Table 5, where only two terms (according to the ground-truth) are marked as relevant. Indeed, by looking at this table, one can find terms such as "*part*", "*kind*", and "*false*" which do not convey any relevant information. The evidence in these tables supports the claim that $S(t)$ is able to differentiate between relevant and non-relevant terms. This is an important step of our process, as 1-gram term scores are the basis for computing the scores of definitive candidate keywords, as we will see in the next section.

Table 4
Top-30 single terms and $S(t)$ scores for document #1001 of Inspec collection.

1	**language**	0.215	11	atomism	0,509	21	responsible	0.874
2	**thought**	0.240	12	**content**	0.626	22	popularizing	0.874
3	atomistic	0.252	13	structure	0.726	23	**information**	0.903
4	jerry	0.262	14	fred	0.768	24	nature	0.979
5	**lot**	0.277	15	**mind**	0.768	25	**cognitive**	0.988
6	**ibs**	0.387	16	representation	0.813	26	relation	1.000
7	dretske	0.429	17	**philosophy**	0.874	27	conflict	1.006
8	fodor	0.464	18	**information-based**	0.874	28	hold	1.053
9	thesi	0.496	19	well-known	0.874	29	symbol	1.105
10	**semantic**	0.508	20	contemporary	0.874	30	**desire**	1.119

Table 5
Bottom-30 single terms and $S(t)$ scores for document #1001 of Inspec collection.

31	simple	1.226	41	combination	1.722	51	hand	2.057
32	concern	1.273	42	connect	1.781	52	substantive	2.232
33	semantically	1.273	43	part	1.781	53	possesse	2.232
34	relevant	1.273	44	supervene	1.959	54	independently	2.232
35	involved	1.273	45	solely	1.959	55	representational	2.232
36	**state**	1.273	46	kind	1.959	56	argue	2.340
37	belief	1.273	47	support	1.959	57	theorie	2.340
38	maintain	1.561	48	logical	1.959	58	false	2.340
39	syntactic	1.561	49	principle	1.959	59	falsehood	2.340
40	mirroring	1.561	50	**system**	1.961	60	result	2.340

3.4. n-gram generation and computing candidate keyword score

To form the list of candidate keywords, we consider a sliding window of size *n*, generating a contiguous sequence of terms ranging from *1*-gram to *n*-grams (where *n* will be experimentally evaluated; more on this later). Candidate keywords can only be formed by sequences of terms if, in addition to belonging to the same sentence, they also belong to the same chunk. This allows us to prevent the formation of unrelated sequences in *n*-grams. For instance, the *n*-gram "*(KEA) which*", extracted from the sentence "*Keyphrase Extraction (KEA), which was*", would not be considered a valid *n*-gram, as, although it is part of the same sentence, its constituent terms belong to different chunks ("*Keyphrase Extraction (KEA)*" and "which was"[11]). Algorithm 5.1 depicts the process of *n*-gram generation.

Algorithm 5.1 *n*-gram generation procedure.

```
Input: sentences, chunks, terms, n
1:    candidateKeywords = []
2:    for each sentence in sentences do
3:        for each chunk in chunks do
4:            tokens = split chunk into tokens
5:                for i = 0..|tokens| do
6:                    cand = ''
7:                    if tokens[i].tag in ['a', 'U', 'p']:
8:                        for j = 0..n-1 do
9:                            cand += (tokens[i+j]) + ' '
10:                           if (!(cand.startswith(stopword)) & !(cand.endswith(stopword))):
11:                               candidateKeywords[index(cand)] = cand
12:                               candidateKeywords[index(cand)].KF++
13:                       end for
14:           end for
15:       end for
16:   end for
Output: List of candidate keywords and corresponding KF weight
```

First, we begin by creating an empty structure (line 1) which will be used to keep all the *n*-gram *candidate keywords* and additional information such as their statistics. We then iterate within the list of *sentences* (line 2) and *chunks* (line 3). Next, we split each *chunk* into annotated *tokens* (line 4) and begin to generate *n*-grams. During this process we only select as candidate keywords those whose individual terms are tagged (line 7) as acronyms (<*a*>), uppercase (<*U*>), or parsable content (<*p*>). Further, and similarly to other research [17,30,42], we assume that relevant keywords rarely begin or end with a stopword. To confirm this assumption, we carried out a study which revealed that only 2.3% of the keywords found in the ground-truth of our experimental datasets begin or end with a stopword. Based on these results, we opt not to consider candidate keywords that fulfill this condition (line 10). This may be understood as a compromise between precision and recall. However, candidate keywords with an *interior* stopword (e.g., "philosophy *of* mind" or "language *of* thought") are considered (more on this below). Digits (<*d*>) are also not considered, as they are rarely part of a keyword. In addition to this, we also add or update information that quantifies the frequency of keywords in the text (*KF*(*kw*) - line 12).

In what follows, we provide an example in order to better illustrate the process of finding keywords. To this purpose, we rely on an excerpt extracted from our running example[12] and consider the generation of *3*-grams. The obtained results are as follows: "*philosophy*", "*philosophy of mind*", "*thesis*", "*information-based*", "*thesis of information-based*", "*information-based semantics*", "*ibs*", "*thesis of content*", "*content*", "*atomism*", "*content atomism*", "*language*", "*language of thought*", and "*thought*". Note that candidate keywords such as "*philosophy of*", "*of mind*", "*the thesis*", "*and the thesis*", and similar keywords with unparsable content, with a number, or that begin or end with a stopword have been discarded. It is also important to highlight that, apart from this static list of stopwords, our approach is language-independent, as we do not follow a linguistic analysis (neither stemming nor part-of-speech is considered).

After concluding the process of producing keywords, each *candidate* is given a score. Algorithm 5.2 depicts this process.

To compute this score, we begin by splitting each candidate into *tokens* (line 2). Next, we evaluate whether the *token* is a stopword (line 6). For the non-stopword *tokens*, we consider their $S(t)$ term scores when multiplying (line 7) or summing (line 8) by the $S(t)$ term scores of the remaining candidate keyword *tokens*. Stopword *tokens* (which may stem from candidate keywords containing interior stopwords, e.g., "language *of* thought") are given special treatment (lines 9–14). Our aim is to mitigate the impact of applying the $S(t)$ score of the stopword term (e.g., "of") – likely a high value – when put in context with the left (e.g., "language") and right (e.g., "thought") non-stopword $S(t)$ scores. In order to achieve this goal, we propose a bigram probability approach which rests on the following expression: $(1 + (1 - \textit{Bigram Probability}))$, where the *Bigram Probability* of the stopword token t is given by $P(t \mid t_{i-1}) * P(t_{i+1} \mid t)$. This means that, in contrast to a high likely $S(t)$ value, a *BigramProbability* score of 1 will be obtained whenever the stopword *token* (e.g., "of") occurs with the very

[11] Recall that chunks stem from the existence of punctuation in a sentence.

[12] "…philosophy of mind: the thesis of Information-Based Semantics (IBS), the thesis of Content Atomism (Atomism) and the thesis of the Language of Thought (LOT)".

Algorithm 5.2 Candidate keyword score procedure.

Input: *candidateKeywords, terms*
1: **for** each *candidate* in *candidateKeywords* **do**
2: *tokens* = split *candidate* into tokens
3: **for** $i = 0..$ |*tokens*| **do**
4: *prod_S* = 1
5: *sum_S* = 0
6: **if** tokens[i] is not a stopword:
7: *prod_S* *= *terms*[index (tokens[*i*])].S_t
8: *sum_S* += *terms*[index (tokens[*i*])].S_t
9: **else**:
10: *probBefore* = *P*(tokens[i] | tokens[*i*-1])
11: *probAfter* = *P*(tokens[*i* + 1] | tokens[*i*])
12: *BigramProbability* = *probBefore* * *probAfter*
13: *prod_S* *= 1 + (1 - *BigramProbability*)
14: *sum_S* -= (1 - *BigramProbability*)
15: **end for**
16: *candidateKeywords*[*candidate*].S_{kw} = *prod_S* / (*candidateKeywords*[*candidate*].KF * (*sum_S* + 1))
17: sort (*candidateKeywords*) by ascending S_{kw} score
18: **end for**
Output: List of *candidateKeywords* sorted by ascending S_{kw} *score*

Table 6
3-gram candidate keywords, *S*(*kw*) scores, and *S*(*kw*) ranking position for document #1001 of Inspec collection.

1	thesis	0.066	20	**information-based semantics**	0.186
2	dretske and jerry	0.066	22	Thought	0.194
3	jerry fodor	0.070	34	**philosophy of mind**	0.254
4	**lot**	0.072	45	thesis of information-based	0.308
5	**language of thought**	0.081	56	**cognitive states**	0.386
6	language and atomistic	0.084	72	Philosophy	0.466
7	atomism	0.084	73	information-based	0.466
8	language	0.088	75	thesis of content	0.483
9	**ibs**	0.093	78	Desires	0.528
10	content	0.096	86	Beliefs	0.560
14	content atomism	0.149			

same *tokens* on the left (e.g., "language") and right (e.g., "thought") side. The impact of this proposal will be evaluated in the experimental part of our paper, in Section 5.3.

The final score of the candidate keyword is given by Eq. (8) (line 16):

$$S(kw) = \frac{\prod_{t \in kw} S(t)}{KF(kw) * (1 + \sum_{t \in kw} S(t))} \tag{8}$$

In this equation, *kw* represents a candidate keyword of one (e.g., "content") or more (e.g., "content atomism") terms, and *S*(*kw*) represents the final score – the lower, the more relevant. The score of a candidate keyword is determined by multiplying (in the numerator) the *S*(*t*) scores or *BigramProbability* scores (in the case of a stopword *token*) of the several terms that constitute the *candidate keyword*. One potential problem of this product, however, is that the length of the *candidate keyword* may highly impact its score, favoring lengthy relevant *candidate keywords* as opposed to short relevant ones. To reduce this impact and extract keywords irrespective of their length, we compute the final score by dividing the numerator by the sum of the *S*(*t*) scores or *BigramProbability* scores (in the case of a stopword *token*), weighted by the candidate keyword frequency *KF*(*kw*). This enables us to not only mitigate this problem but also to differentiate between *candidate keywords* which are constituted by the same terms ("keyword"; "extractor") but have a different order ("keyword extractor"; "extractor keyword"). The rationale is that a *candidate keyword* that appears more often ("keyword extractor") should be given a higher score than a *candidate keyword* ("extractor keyword") that, despite being formed by the same terms, does not occur that often. This also enables us to handle *1*-gram *candidate keywords* (e.g., "content", "atomism") more appropriately, by giving higher scores to relevant terms that appear multiple times than to relevant terms which do not appear that often. The final list of *candidate keywords* is then ordered by ascending S_{kw} score such that the lower the score, the more relevant the candidate keyword will be.

Table 6 lists the *3*-gram candidate keywords of our running example, their ranking position, and the corresponding *S*(*kw*) scores (the lower the better). Of the 126 candidate keywords, we list 23 candidate keywords. (We mostly focus – for reasons of space – on the candidate keywords of the excerpt above considered + gold keywords[13] (in boldface).) However, knowing

[13] Note that three of the gold keywords ("artificial intelligence"; "cognitive systems"; and "philosophical aspects") are absent; that is, they do not appear in the text. Therefore, they will not be listed in the table.

that some of the candidate keywords may be very similar, we propose a data deduplication phase, where potential similar candidate keywords may be eliminated. This will be discussed in the next section.

3.5. Data deduplication and ranking

In the final step of our proposal, we aim to learn whether or not removing similar potential candidates improves the ranking results. Algorithm 6 illustrates this process.

Algorithm 6 Data deduplication procedure.

```
Input: candidateKeywords, θ
1:    keywords = [candidateKeywords[0]]
2:    for each candidate in candidateKeywords do
3:        skip = false
4:        for each key in keywords do
5:            if DistanceSimilarity(candidate, key) > θ:
6:                skip = true
7:                Break
8:        end for
9:        If not skip:
10:           keywords.add(candidate)
11:   end for
Output: List of keywords sorted by ascending S_kw score
```

To compare similar candidate keywords, we begin by creating a structure, named *keywords*, which will be used to keep the final list of *keywords*. Our first step (line 1) is to initialize the structure with the *candidate keyword* having the lowest score (in our running example, "*thesis*"). Then each subsequent *candidate keyword* (e.g., "*dretske and jerry*" – line 2) is compared to the *keywords* already existing in the *keywords* structure (lines 4–5) and discarded whenever it is considered similar (lines 6–7). In our case, two strings are considered similar if their distance similarity score is above a given threshold θ (to be determined in the experiments section). Between two strings considered similar, we keep the most relevant one, that is, the one that has the lowest $S(kw)$ score (i.e., the keyword that is already in the structure). Conversely, a *candidate keyword* is inserted into the list (lines 9–10) if no similarity between it and keywords already in the structure is found to be relevant. The final list of *keywords* is then given by ascending order of the S_{kw} scores.

To study the effect of this deduplication phase and its suitability in discarding similar potential candidate keywords, we make use of three different distance similarity measures: the Levenshtein similarity [24], the Jaro-Winkler similarity [12], and the sequence matcher. We apply a randomly chosen Levenshtein's threshold of 0.7 to our running example and compare the results to those obtained in Table 6 (which have no deduplication; that is, they have a Levenshtein's threshold equal to 1). Looking at Table 7, we can observe that although the top 10 keywords (left-hand side of the table) remain the same, some minor changes can be observed in the remaining positions (right-hand side of the table). For instance, the gold keyword "*information-based semantics*" will climb one position from 20th to 19th, meaning that one more gold keyword can be found within the top-20 results if a deduplication stage is applied. This means that while precision at 10 remains the same, precision at 20 is improved with one more gold keyword at the top of the list.

If a more flexible threshold is applied, several candidate keywords will be more easily considered similar (as is the case of "language" and "language of thought", and of "language" and "language and atomistic"). But several others will be discarded. Table 8 shows the final list for a Levenshtein's threshold of 0.2. Indeed, while four gold keywords can still be found within the retrieved results, only six results are retrieved in total, thus affecting the retrieval of a number of gold keywords and consequently degrading recall. A more thorough analysis of this should, however, be conducted for different thresholds and different n to reach a conclusion. This is carried out in the experimental part of this research, which is described in the following section.

Table 7
3-gram candidate keywords, $S(kw)$ scores and $S(kw)$ ranking position for document #1001 of the Inspec collection. Levenshtein's = 0.7.

1	thesis	0.066	20 → 19	**information-based semantics**	0.186
2	dretske and jerry	0.066	22 → 20	Thought	0.194
3	jerry fodor	0.070	34 → 30	**philosophy of mind**	0.254
4	**Lot**	0.072	45 → 41	thesis of information-based	0.308
5	**language of thought**	0.081	56 → 50	**cognitive states**	0.386
6	language and atomistic	0.084	72 → 64	Philosophy	0.466
7	atomism	0.084	73 → 65	information-based	0.466
8	language	0.088	75 → 67	thesis of content	0.483
9	**lbs**	0.093	78 → 69	Desires	0.528
10	content	0.096	86 → 77	Beliefs	0.560
14	content atomism	0.149			

Table 8
3-gram candidate keywords, $S(kw)$ scores, and $S(kw)$ ranking position for document #1001 of Inspec collection. Levenshtein's = 0.2.

$1 \rightarrow 1$	Thesis	0.066
$2 \rightarrow 2$	dretske and jerry	0.066
$4 \rightarrow 3$	**Lot**	0.072
$5 \rightarrow 4$	**language of thought**	0.081
$9 \rightarrow 5$	**ibs**	0.093
$34 \rightarrow 6$	**philosophy of mind**	0.254

4. Experimental setup

In this section, we describe the experimental setup of our approach. In particular, we introduce the ground truth datasets, the baseline methods, and the evaluation metrics used in our experiments. This anticipates the discussion of the results in Section 5.

4.1. Dataset description

To evaluate the effectiveness of our system, we tested it on twenty different document collections. Our aim was to understand how YAKE! behaves under a large number of scenarios. The collections included five different languages (English, French, Spanish, Portuguese, and Polish), several different domains (including, but not limited to, computer science, agriculture, and physics), and six different types of documents (new reports, full papers, abstracts, research reports, theses, and paragraphs) ranging in length from very short (75 tokens per document) to large (8 K tokens per document). Besides classical collections such as *Inspec* [21], *SemEval2010* [27], *SemEval2017* [1], *citeulike180* [34], *Nguyen2007* [38], *PubMed, Schutz2008* [44], *Krapivin2009* [28], *WWW* [16], and *KDD* [16], we also made use of *110-PT-BN-KP* [32], *500N-KPCrowd-v1.1* [31], *theses100, wiki20* [33], *fao30* [35], *fao780* [35], *wicc* [2], *cacic* [2], and WikiNews, plus a newly developed collection, pak2018, which gathers together Polish documents. Table 9 provides an overview of the statistics of all the datasets [github.com/LIAAD/KeywordExtractor-Datasets], which together total 13,355 docs. In the following, we describe each dataset in detail.

Inspec [21] consists of 2,000 abstracts of scientific journal papers in computer science collected between the years 1998 and 2002. Each document has two sets of keywords assigned: the controlled keywords, which are manually controlled assigned keywords that appear in the Inspec thesaurus but may not appear in the document, and the uncontrolled keywords, which are freely assigned by the editors (that is, they are not restricted to the thesaurus or to the document). In our experiments, we consider a union of both sets as the ground-truth.

Similarly to the above datasets, the ***WWW*** [16] and ***KDD*** [16] collections are based on the abstracts of papers collected from the World Wide Web Conference (WWW) and the ACM Conference on Knowledge Discovery and Data Mining (KDD) published during the period 2004–2014, containing 1,330 and 755 documents, respectively. The gold keywords of these papers are the author-labeled terms. In terms of the number of tokens per document, these are the smallest datasets used in our experiments (84 and 75 tokens per document on average).

SemEval2010 [27] consists of 244 full scientific papers extracted from the ACM Digital Library (one of the most popular datasets previously used for keyword extraction evaluation). The papers range from 6 to 8 pages in length and belong to four different computer science research areas: distributed systems; information search and retrieval; distributed artificial intelligence (multiagent systems); and social and behavioral sciences (economics). Each paper has an author-assigned set of keywords (which are part of the original pdf file) and a set of keywords assigned by professional editors, both of which may or may not appear explicitly in the text. This dataset is the largest one used in our experiments, averaging 8,332 tokens per document.

Recently, Augenstein et al. [1] participated in SemEval2017 Task 10 with the purpose of extracting keywords and relations from scientific publications. This gave rise to a new dataset, ***SemEval2017***, which consists of 500 paragraphs selected from 500 ScienceDirect journal articles, evenly distributed among the domains of computer science, material sciences, and physics. Each text has a number of keywords selected by one undergraduate student and an expert annotator. The expert's annotation is prioritized whenever there is disagreement between the two annotators.

The ***Krapivin2009*** [28] is the largest dataset in terms of documents, with 2,304 full papers from the computer science domain, which were published by ACM in the period 2003–2005. The papers were downloaded from CiteSeerX Autonomous Digital Library and each one has keywords assigned by the authors and verified by the reviewers, resulting in 6.34 gold keywords per document and 8,040.74 tokens per document, on average.

The ***citeulike180*** [34] dataset is based on the CiteULike.org platform, which organizes academic citations. The CiteULike.org corpus is a freely available full-text paper collection that contains information about the users who tagged the documents. The dataset is based on a subset of CiteULike.org containing documents that have been indexed with at least three keywords on which at least two users have agreed. In addition to filtering the document set, the dataset only

Table 9
Dataset summary statistics.

Dataset	Language	Type of Doc	Domain	#Docs per Dataset	#Gold Keys (per doc)	#Tokens per Doc	Absent GoldKe	#Stopword in Gold Key
110-PT-BN-KP	PT	News	Misc.	110	2610 (23.73)	304.00	2.5%	15.7%
500N-KP-Crowd-v1.1	EN	News	Misc.	500	24459 (48.92)	408.33	13.5%	10.5%
Inspec	EN	Abstract	Comp. Science	2000	29230 (14.62)	128.20	37.7%	5.6%
Krapivin2009	EN	Paper	Comp. Science	2304	14599 (6.34)	8040.74	15.3%	5.3%
Nguyen2007	EN	Paper	Comp. Science	209	2369 (11.33)	5201.09	17.8%	8.5%
PubMed	EN	Paper	Comp. Science	500	7620 (15.24)	3992.78	60.2%	3.9%
Schutz2008	EN	Paper	Comp. Science	1231	55013 (44.69)	3901.31	13.6%	3.5%
WWW	EN	Abstract	Comp. Science	1330	7711 (5.80)	84.08	55.0%	8.0%
KDD	EN	Abstract	Comp. Science	755	3831 (5.07)	75.97	53.2%	5.7%
SemEval2010	EN	Paper	Comp. Science	243	4002 (16.47)	8332.34	11.3%	9.3%
SemEval2017	EN	Paragraph	Misc.	493	8969 (18.19)	178.22	0.0%	25.4%
cacic	ES	Paper	Comp. Science	888	4282 (4.82)	3985.84	2.2%	24.5%
citeulike180	EN	Paper	Misc.	183	3370 (18.42)	4796.08	32.2%	2.4%
fao30	EN	Paper	Agriculture	30	997 (33.23)	4777.70	41.7%	1.9%
fao780	EN	Paper	Agriculture	779	6990 (8.97)	4971.79	36.1%	3.5%
pak2018	PL	Abstract	Misc.	50	232 (4.64)	97.36	64.7%	22.0%
theses100	EN	Msc/Phd Thesis	Misc.	100	767 (7.67)	4728.86	47.6%	18.3%
wicc	ES	Paper	Comp. Science	1640	7498 (4.57)	1955.56	2.7%	28.5%
wiki20	EN	Research Reports	Comp. Science	20	730 (36.50)	6177.65	51.8%	3.7%
WikiNews	FR	News	Misc.	100	1177 (11.77)	293.52	5.0%	14.9%

considers annotators who have at least two additional co-annotators tagging the same common document. The result is a set of 180 documents indexed by 332 taggers, where most documents are related to the area of bioinformatics.

The ***Nguyen2007*** [38] is a dataset composed of 211 scientific conference papers. The gold keywords were manually assigned by student volunteers who were each given three papers to read. The keywords assigned by the authors of the papers were hidden to avoid bias. The result is a dataset with 5,201 tokens per document and 11.33 gold keywords per document, on average.

Both ***Schutz2008*** [44] and ***PubMed*** are datasets based on full-text papers collected from PubMed Central, which comprises over 26 million citations for biomedical literature from MEDLINE, life science journals, and online books. The former consists of 1,231 papers selected from PubMed Central, while the latter consists of 500 papers selected from the same source. *Schutz2008* documents are distributed across 254 different journals, ranging from Abdominal Imaging to World Journal of Urology. The keywords assigned by the authors are hidden in the article and used as gold keywords. *PubMed* uses the Medical Subject Headings (MeSH – [www.ncbi.nlm.nih.gov/mesh]), a controlled vocabulary thesaurus used for indexing articles for PubMed, as the gold keywords for the documents, resulting in 14.24 gold keywords per document.

The ***theses100*** dataset consists of 100 full master and Ph.D. theses from the University of Waikato, New Zealand. The domains of the theses are quite diverse, ranging from chemistry, computer science, and economics to psychology, philosophy, history, and others.

We also evaluated the performance of our system on agricultural documents obtained from two datasets based on Food and Agriculture Organization (FAO) of the United Nations: ***fao780*** [35], with 780 documents, and ***fao30*** [35], with 30 documents. Both datasets are full-text documents randomly selected from the FAO's repository, where *fao780* keywords were manually tagged by professional FAO staff with terms from the Agrovoc thesaurus and *fao30* keywords were manually assigned by six professional annotators at FAO.

wiki20 [33] consists of 20 English technical research reports covering different aspects of computer science. Fifteen teams, each consisting of two senior computer science undergraduates, assigned keywords to each report using Wikipedia article titles as the candidate vocabulary. The teams were instructed to assign around 5 keywords to each document. Each team assigned 5.7 keywords on average and each document has 35.5 gold keywords on average. An overall inter-indexer consistency agreement between the teams of 30.5% was reported by authors of the dataset. **WikiNews** is a French corpus created from the French version of WikiNews that contains 100 news articles published between May 2012 and December 2012 and manually annotated by at least three students.

The ***110-PT-BN-KP*** [32] is a TV Broadcast News (BN) dataset that contains 110 transcription text documents from 8 broadcast news programs from the European Portuguese ALERT BN database covering politics, sports, finance, and other broadcast news. After the speech-to-text transcription, each news report was manually reexamined to fix any segmentation errors and the gold keywords were created by asking one tagger to extract all keywords that summarize the document content. Similarly, the ***500N-KPCrowd-v1.1*** [31] is another broadcast news transcription dataset. This dataset consists of 500 English broadcast news stories in 10 different categories (art and culture; business; crime; fashion; health; politics us; politics world; science; sports; technology) with 50 docs per category. The ground truth is built using Amazon's Mechanical Turk service to recruit and manage taggers. Multiple annotators were required to look at the same news story and assign a set of keywords from the text itself. The final ground truth consists of keywords selected at least by 90% of the taggers, resulting in 48.92 gold keywords per documents.

The two Spanish datasets [2] consist of scientific papers published in the Argentine Congress of Computer Science (CACIC – [redunci.info.unlp.edu.ar/cacic.html]) and in the Workshop of Researchers in Computer Science (WICC – [redunci.info.unlp.edu.ar/wicc.html]). The ***cacic*** dataset is formed by a set of scientific articles published between 2005 and 2013 and consist of 888 documents. The ***wicc*** dataset is composed of 1,640 scientific articles published between 1999 and 2012, with the smallest number of gold keywords per document, 4.57 on average.

Finally, ***pak2018*** is a dataset in Polish created by a member of our team so that we could test our system and baselines over an additional language. The corpus consists of 50 abstracts of journal publications on technical topics collected from Measurement Automation and Monitoring (in Polish "Pomiary, Automatyka, Kontrola" – [pak.info.pl/]), a well-known monthly journal devoted to electronics and measurement sciences. The journal has been appearing for over 50 years. The keywords are author-assigned for each publication, ranging from 2 to 6 keywords per document (4.64 keywords per document on average). When creating the dataset, we randomly selected 50 articles published in recent editions of the journal and collected their manually assigned keywords. Note that the *pak2018* collection has 64.7% absent keys (as stated in the table); for this reason, we did not expect to obtain high scores when testing on this particular dataset. Similar results were expected for *PubMed, WWW* [16], *KDD* [16], *theses100, wiki20* [33], *fao30* [35], and *fao780* [35], which also present a substantial number of absent keywords.

A conspicuous feature of our overall dataset is that a considerable number of collections include a large proportion of keywords having at least one stopword. This is particularly evident for the Spanish and Portuguese collections, but also for SemEval2017, and presents an important challenge for keyword retrieval. This is given further attention in Section 6.3. Another interesting thing to note is that most of the annotators prefer keywords with either 1, 2, or 3 terms (see Table 10); such keywords represent more than half of the gold keywords in every collection. In contrast, yet not surprisingly, only a tiny portion of the gold keywords have four and five terms, which shows that people rarely use more than three terms to describe a given subject. This result in is line with the research of Spink et al. [45], who showed that the average length of a search query was 2.4 terms.

Table 10
Distribution of gold keywords per *n*-gram by dataset.

Datasets	1-gram		2- grams		3- grams		4- grams		≥ 5- grams	
110-PT-BN-KP	**1686**	**64.60%**	586	22.45%	284	10.88%	41	1.57%	13	0.50%
500N-KPCrowd-v1.1	**17851**	**72.98%**	4598	18.80%	1260	5.15%	439	1.79%	311	1.27%
Inspec	6147	21.03%	**13268**	**45.39%**	7010	23.98%	1976	6.76%	829	2.84%
Krapivin2009	4354	29.82%	**7000**	**47.95%**	2472	16.93%	604	4.14%	169	1.16%
Nguyen2007	638	26.93%	**1110**	**46.86%**	408	17.22%	144	6.08%	69	2.91%
PubMed	**2969**	**38.96%**	2957	38.81%	1223	16.05%	398	5.22%	73	0.96%
Schutz2008	**32312**	**58.74%**	16225	29.49%	5058	9.19%	1129	2.05%	289	0.53%
WWW	**3318**	**43.03%**	3033	39.33%	886	11.49%	422	5.47%	52	0.67%
KDD	1479	38.61%	**1654**	**43.17%**	514	13.42%	155	4.05%	29	0.76%
SemEval2010	918	22.94%	**1767**	**44.15%**	944	23.59%	266	6.65%	107	2.67%
SemEval2017	2362	26.34%	**2565**	**28.60%**	1604	17.88%	892	9.95%	1546	17.24%
cacic	**1843**	**43.04%**	1119	26.13%	832	19.43%	238	5.56%	250	5.84%
citeulike180	**2641**	**78.37%**	637	18.90%	89	2.64%	3	0.09%	0	0.00%
fao30	446	44.73%	**510**	**51.15%**	33	3.31%	7	0.70%	1	0.10%
fao780	**3405**	**48.71%**	3338	47.75%	226	3.23%	19	0.27%	2	0.03%
pak2018	95	40.95%	**105**	**45.26%**	24	10.34%	5	2.16%	3	1.29%
theses100	307	40.03%	**325**	**42.37%**	86	11.21%	40	5.22%	9	1.17%
wicc	**3437**	**45.84%**	1722	22.97%	1467	19.57%	510	6.80%	362	4.83%
wiki20	196	26.85%	**353**	**48.36%**	141	19.32%	37	5.07%	3	0.41%
WikiNews	**764**	**64.91%**	220	18.69%	97	8.24%	59	5.01%	37	3.14%
All	**87168**	**46.75%**	63092	33.84%	24658	13.22%	7384	3.96%	4154	2.23%

To understand the best parameters of YAKE! and the effectiveness of all the approaches from a single unified view, we also consider, in addition to each dataset, the union of all the dataset documents into a single aggregate collection. However, instead of merely merging all the documents indiscriminately, which would result in having a collection dominated by the biggest individual datasets, we apply a re-sampling technique called random oversampling, thus guaranteeing a collection free of bias. To this purpose, we randomly duplicate the documents of each dataset, until all the datasets have the same number of documents as the largest collection, which in this case is the Krapivin2009 [28] dataset with 2,304 documents. This means that each of the 20 datasets will see their documents randomly duplicated until they have 2,304 documents, creating an aggregate collection of 46,080 documents. By doing this, we guarantee a fair representation of each dataset in the aggregate collection. We will delve into this more deeply when discussing the use of this collection when applying cross-validation in the experiments.

4.2. Baseline methods

Although a considerable number of keyword extracting methodologies have been proposed over the last few years, evaluating the effectiveness of a keyword algorithm is still a challenging and difficult task. In this work, we compare our approach against state-of-the-art algorithms that have an available implementation. We make use of PKE [github.com/boudinfl/pke], a state-of-the-art open-source Python-based keyword extraction toolkit, made available by Boudin [5], which makes it possible to transparently compare our approach against the TF.IDF [22], KP-Miner [13], SingleRank [48], TopicRank [4], TopicalPageRank [46], PositionRank [15], and MultipartiteRank [6] unsupervised systems. In our experiments, IDF is calculated separately for each dataset being tested, both for TF.IDF and KP-Miner. For the graph-based methods we employed the default parameters as set in PKE and in the corresponding papers, namely, a co-occurrence window size set to 10 for SingleRank, PositionRank, and TopicalPageRank (an LDA model for each dataset), and a minimum similarity for clustering set to 0.74 with the linkage method for both TopicRank and MultipartiteRank. Appropriate stemmers (for each of the different languages considered in the evaluation stage) were also employed whenever needed. For a full list of the parameters, readers are recommended to read the full version of each of the papers or to refer to the PKE package [boudinfl.github.io/pke/build/html/index.html].

In addition to this, we also compare YAKE! against further available state-of-the-art algorithms: the TextRank [www.hlt.utdallas.edu/~saidul/code.html] [37], ExpandRank [www.hlt.utdallas.edu/~saidul/code.html] [48], and RAKE [github.com/zelandiya/RAKE-tutorial] [42] unsupervised systems. Following the work of Mihalcea and Tarau [37] and Wan and Xiao [48], we set the co-occurrence window size for TextRank to 2 and ExpandRank to 10, as these values have yielded the best results for their evaluation datasets. In addition, for ExpandRank, we define the five nearest document neighbors for each processed document. A full description of the parameters can be found in the corresponding papers. For RAKE, no parameters were needed or defined.

Supervised methods are also taken into account by testing against KEA [www.nzdl.org/Kea/] [50], a supervised approach which is trained through 5-fold cross-validation using the documents of each corresponding dataset. The maximum keyword length was set to 3. No external corpus statistics other than the documents themselves were used. We set the minimum keyword length to 1, the minimum number of occurrences of a keyword to 2, and used the SremovalStemmer as a stemmer. While this comparison is inherently biased towards methods that can use vast amounts of statistical data, it should allow us

Table 11
Summary features of YAKE!, and baseline methods.

Methodology		Name	Language Dependence			External Corpus
			Stopword List	POS	Stemming	
Unsupervised	Statistical	YAKE!	√			
		TF.IDF	√			√
		KP-Miner	√			
		RAKE	√			
	Graph-based methods	TextRank		√		
		SingleRank	√	√		
		TopicRank	√	√		
		TopicalPageRank	√	√	√	√
		PositionRank	√	√		
		MultipartiteRank	√	√	√	
		ExpandRank		√		√
supervised	binary	KEA			√	√

to quantify the performance drop resulitng from the lack of external corpora. Table 11 summarizes some of the specificities of the baseline methods.

4.3. Evaluation metrics

For evaluation of the results, we follow an exact match, where keywords (automatically extracted and manually labeled) are reduced to their stems as a means to reduce mismatch. We used Porter NLTK [www.nltk.org/_modules/nltk/stem/porter.html] for the English language, PolishStemmer [github.com/eugeniashurko/polish-stem] for the Polish, RSLP Stemmer. In our experiments, missing keywords were not removed. To empirically evaluate the effectiveness of our approach against different parameters and baselines methods, we propose classical evaluation metrics in IR based on a confusion matrix, with TP being the number of keywords correctly identified as relevant, TN the number of keywords correctly identified as non-relevant, FP the number of keywords wrongly identified as relevant, and FN the number of keywords wrongly identified as non-relevant. Traditionally, keyword extraction, by nature, is a ranking problem. Based on this, we opted to calculate Precision at k (P0040k), Recall at k (R@k), F1-Measure at k (F1@k), Mean Average Precision at k (MAP@k), r-precision, and mean reciprocal rank to determine the effectiveness of our method.

More specifically, *P@k* measures how many relevant keywords are in the top-k results. Similarly, *R@k* measures the fraction of relevant keywords that are successfully retrieved in the top-k positions. F1@k is the harmonic mean of recall and precision and has the advantage of summarizing effectiveness in a single number. MAP@k, in turn, distinguishes between differences in the rankings at positions 1 to k. In addition, we also use R-precision to measure the fraction of relevant keywords that are successfully retrieved at the Rth position in the ranking, where R is the total number of relevant keywords, as a different number of relevant keywords may be found in the different collections. The mean reciprocal rank is defined as the reciprocal (inverse) of the rank at which the first relevant document is retrieved. Finally, we also calculate precision at fixed recall levels from 0.0 to 1.0 as a way to summarize the ranking of all the relevant keywords.

For the computation of the IR metrics, we applied a micro-average approach where TP, FP, FN, and TN are first summed up before being computed. In order to avoid over-fitting and to understand the generalizability of the results, we followed a 5-fold cross-validation approach, which operates by randomly partitioning the set of documents into five folds. Paired tests were used to assess the effectiveness of YAKE! against the state-of-the-art approaches. The null hypothesis is that YAKE! has identical results as each of the other methods. The statistical significance level used is 99% (p-value < 0.01). The significance level of 95% (p-value < 0.05) is also used to give the reader more information when there is not enough evidence at 99%.

5. Results

In this section, we present the results of our experiments comparing YAKE! against state-of-the-art methods. We divided our experiments into two different parts: (1) parameter tuning; and (2) overall effectiveness versus baseline methods. In the first set of experiments, we aimed to determine the best parameter values. We did this considering not only each of the twenty datasets individually but also the aggregate oversampling view which gathers the documents of all the datesets into a single collection. This is an important step as it enables us to define the best possible configuration for YAKE! in the specific case of each collection and in the aggregate mode. Next, we evaluated the overall effectiveness of YAKE! against baseline methods over twenty different datasets so that we could reach conclusions with regard to its effectiveness with different document sizes, document types, and languages.

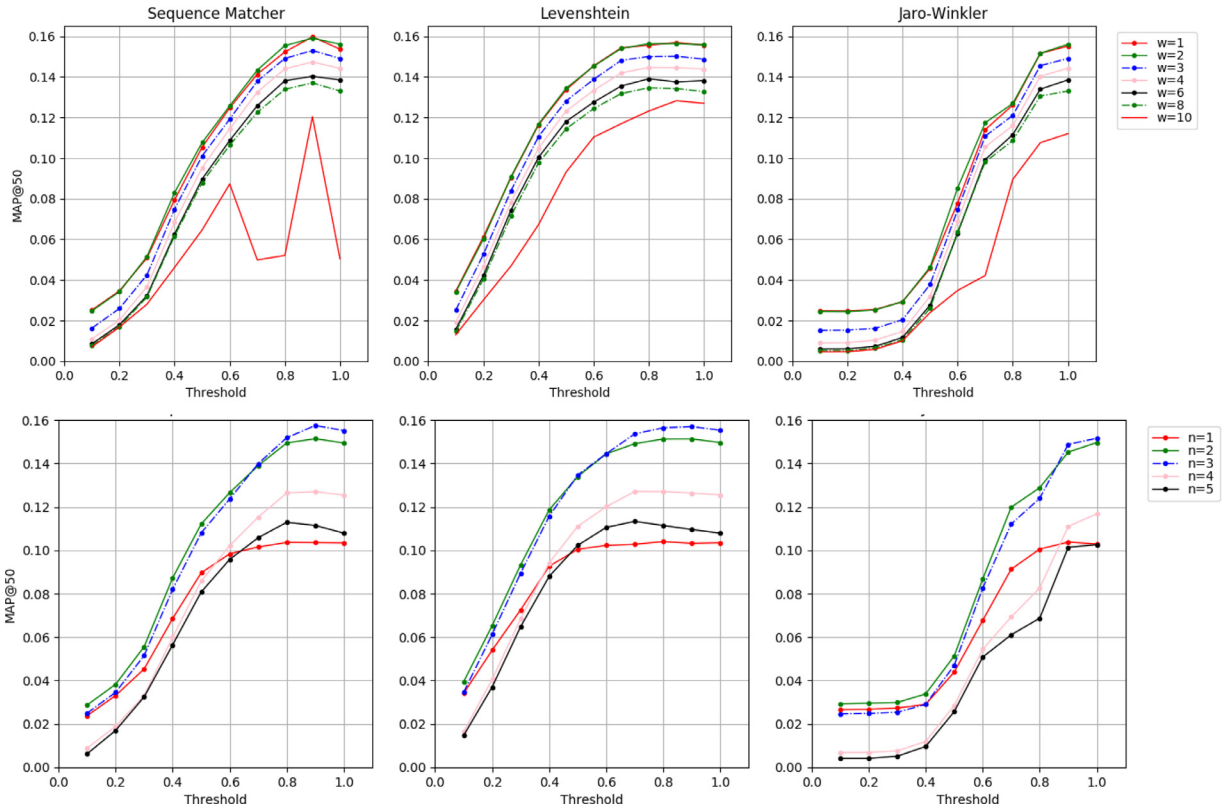


Fig. 3. Sequence Matcher vs Levenshtein vs Jaro-Winkler MAP@50 effectiveness when $1 \leq \boldsymbol{w} \leq 10$ (first row) and $1 \leq \boldsymbol{n} \leq 5$ (second row) on top of the oversampled collection.

5.1. YAKE! parameter tuning

In this section, we analyze the effectiveness of YAKE! under various settings. For this purpose, we began by experimentally evaluating the Sequence Matcher, the Levenshtein distance similarity measure, and the Jaro-Winkler on top of our oversampled collection for several different values of w and n, more specifically $1 \leq w \leq 10$ and $1 \leq n \leq 5$. Both experiments were carried out following a 5-fold cross-validation approach. However, instead of merely looking at the top-10 results (the de facto standard), we opted to look at the first 50 retrieved keywords as a means to better understand the effect of applying a deduplication stage. Three reasons for this can be advanced. First, several documents have more than 10 gold keywords assigned, thus making it important to understand their distribution along the final list of results. Second, looking at only the top-10 results severely limits the number of keywords in comparison to the high number of suggested candidate keywords, making it difficult for a keyword to place within the top 10. Therefore, it is again important to understand retrieved keywords' distribution along the list of the results. Finally, understanding whether to apply a more restrictive threshold, a more flexible threshold, or no threshold at all is an important step to understand whether there is any significant improvement that stems from pushing closer to the top relevant keywords that may (with a different threshold) be positioned at the bottom of the list. On these grounds, we resorted to MAP@50, which particularly suits this task.

In our first analysis, we plotted the effect of applying a deduplication step (for thresholds between 0.1 and 1) with regard to the w parameter, where $1 \leq w \leq 10$. An overall analysis of Fig. 3 (first row) shows that the best results are obtained by Sequence Matcher with a threshold of 0.9 for a MAP@50 score of approximately 0.16 (and a w of 1), followed by Levenshtein with a threshold of 0.8, and Jaro-Winkler with a threshold of 1 (thus meaning that no deduplication stage should be applied). The lines in the figure also suggest that, unlike Jaro-Winkler, both Sequence Matcher and Levenshtein are more stable for thresholds between 0.8 and 1 (meaning that any of these could be applied with just a tiny difference). Further observation enables us to conclude that the best results are obtained when w is equal to 1, though differences with $w = 2$ are negligible. In contrast, the behavior of w seems to be very unstable when equal to 10. Based on these results, we opted to define w to 1 for the rest of the experiments.

Next, we tried to evaluate the very same effect, this time on top of the n parameter, where $1 \leq n \leq 5$ (see the second row of Fig. 3). Once again, we were able to confirm that the best results were obtained by Sequence Matcher with a threshold of 0.9 for a MAP@50 score of approximately 0.16, followed by Levenshtein and Jaro-Winkler. A substantial difference, however, can be observed this time for Sequence Matcher, whose results are substantially superior when approaching a threshold of

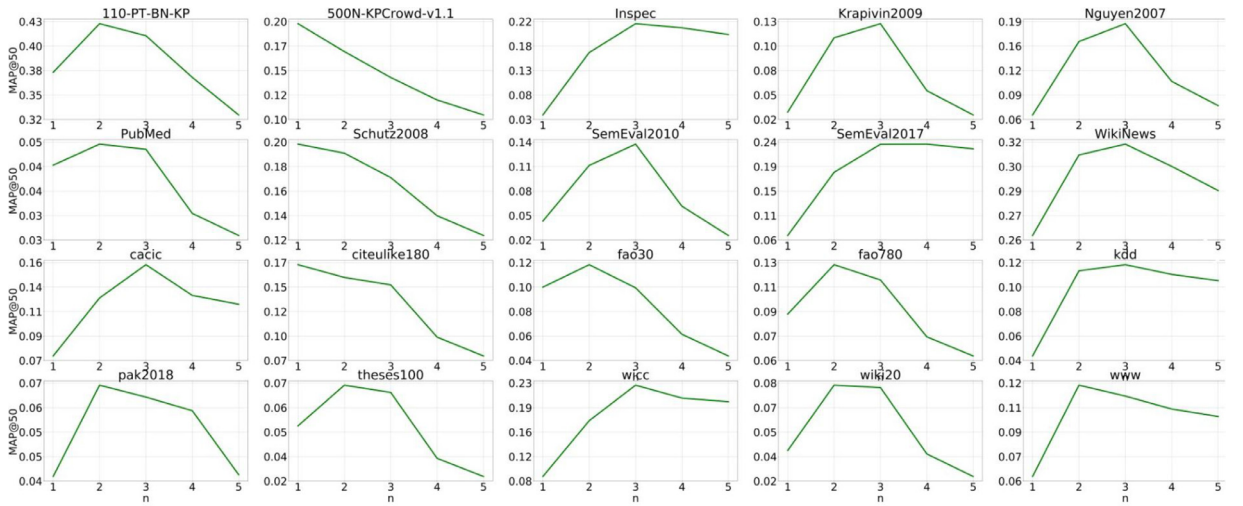


Fig. 4. Effects of changing the n-gram size from 1 to 5 on top of each collection.

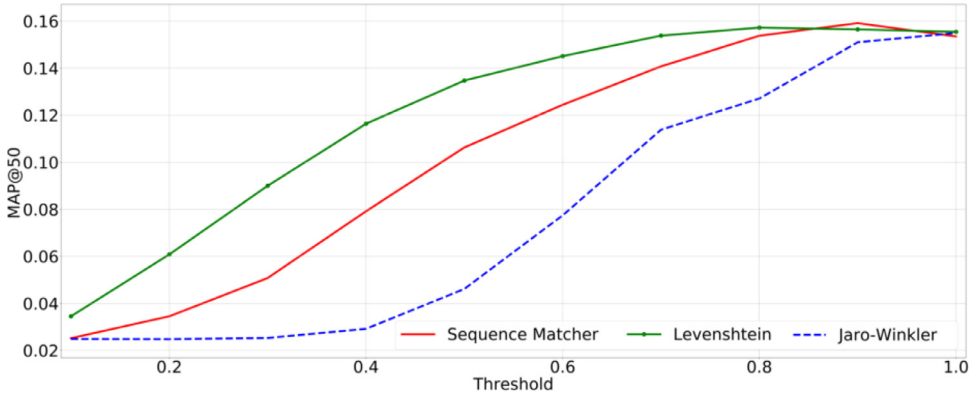


Fig. 5. Sequence Matcher vs Levenshtein vs Jaro-Winkler with $\boldsymbol{w} = 1$ and $\boldsymbol{n} = 3$ on top of the oversampled collection.

0.9, before they quickly decay when a threshold of 1 (no deduplication stage) is applied. Not surprisingly, the best results were obtained for $n = 3$ and $n = 2$, which according to Table 10 seems to concentrate the large majority of the gold keywords. Recall that 3-grams corresponds to the formation of a number of candidate keywords within a window of 3 terms that may be composed of 1 to 3 grams. Somehow these results were anticipated, as outputting only 1-gram term candidates seems too short,[14] while outputting 4- to 5-gram terms seems too complex, due to the increasing complexity of selecting, comparing, and eventually obtaining the best keywords from among so many candidates. An experiment conducted on top of the twenty datasets shows that the number of candidate terms produced increases linearly as n grows where n varies from $1 \leq n \leq 5$. For instance, for SemEval2010 (similarly to other datasets), we can observe a slope of 1 M candidate keywords, which means that for each increase in n, 1 M of candidate keywords are added to the pool. This means that while for 3 grams there are approximately 2 M cumulative candidate keywords, for 5 grams there are roughly 5 M, significantly increasing the complexity in finding relevant keywords.

In the following, we analyze the variation of n on each of the 20 datasets, to understand whether there is any substantial difference between them. Not surprisingly, the results plotted in Fig. 4 show that the values vary according to the dataset studied, with a significant majority obtaining the best results when $n = 3$. Based on all these analyses, we opted to fix n at 3.

Before moving into the next section, we show in Fig. 5a summary plot of the effects of applying the Sequence Matcher, Levenshtein, and Jaro-Winkler deduplication step for the parameters set above ($w = 1$ and $n = 3$). This coarse-grained figure clearly shows that, while values are similar between thresholds 0.8 and 1 (for which no deduplication is applied), the best results are still those of SequenceMatcher with a threshold of 0.9 (and a MAP@50 score of 0.159), though closely followed by Levenshtein with a threshold of 0.8 (and MAP@50 score of 0.157) and Jaro-Winkler with a threshold of 1 (and MAP@50 score

[14] This is because some 2-grams and 3-grams (which, together with 1-grams, constitute the large majority of the gold keywords) would be simply disregarded.

of 0.154). On these grounds, we opted to make use of SequenceMatcher (with a threshold equal to 0.9) and to set $w = 1$ and $n = 3$. In the following section, we discuss the importance of each of the six statistical features, on both the oversampling aggregate collection and each of the 20 datasets.

5.2. Overall effectiveness

In order to evaluate the effectiveness of YAKE! we carried out experiments over 20 different datasets and compared YAKE!'s performance against that of eleven state-of-the-art baselines, 10 unsupervised (TF.IDF, KP-Miner, Rake, TextRank, SingleRank, TopicRank, TopicalPageRank, PositionRank, MultipartiteRank, and ExpandRank) and one supervised (KEA). Table 12 lists the results of F1@10 (which is commonly used for this kind of evaluation) for $n = 3$ and $w = 1$, as these parameters achieved the best results in the tuning process. Again, we followed a *5*-fold cross-validation approach, which operates by randomly partitioning the set of documents into five folds. A paired *t*-test was used to assess the validity of the proposed solutions with statistical significance (p-value < 0.01 (▲, ▼) or p-value < 0.05 ($\triangledown$, $\triangle$)). Based on the results obtained, we confirm that keyword extraction suffers from poor effectiveness when compared to many other Natural Language Processing tasks, including Information Retrieval, for reasons already pointed out. From Table 12, we can observe that the best results of YAKE! occur with the Portuguese dataset with an F1@10 score close to 0.50. The worst results occur with the pak2018, PubMed, and Theses100 datasets, which, not coincidentally, are among the datasets with the most absent keywords. The results of our empirical evaluation show that our method generally outperforms not only statistical methods (tf.idf, kp-miner, and rake), but also the state-of-the-art graph-based methods (TextRank, SingleRank, TopicRank, TopicalPageRank, PositionRank, MultipartiteRank, and ExpandRank) across different datasets, languages, and domains. To put these results in perspective, one should keep in mind that graph-based methods use PoS while YAKE! does not; this enables YAKE! to be language-independent. Another notable point is that YAKE! outperforms KEA in most of the cases, which is particularly relevant in that YAKE!, unlike KEA, does not need a trained model.

A more fine-grained analysis allows us to conclude that while YAKE! generally outperforms state-of-the-art methods when dealing with large texts (as is the case of the 3 collections with the largest texts: Krapivin2009, SemEval2010, and Wiki20; see first row of Fig. 6), it still performs well when tackling short text lengths (as is the case of the 3 collections with the smallest texts on average: WWW, KDD, and pak2018; see second row of Fig. 6). In addition, we can also observe that YAKE! also performs generally well for different domains and type of documents (as is the case of Fao780 – agricultural papers; 500N-KP-Crowd-v1.1 – news; and Nguyen2007 – computer science papers; see third row of Fig. 6) and languages other than English (as is the case of WICC – Spanish; WikiNews – French; 110-PT-BN-KP – Portuguese; or pak2018 – Polish; see fourth and second row of Fig. 6).

A summary of the results is shown in Tables 13 and 14. Table 13 presents a direct comparison between YAKE!'s effectiveness and that of the baselines in percentage terms. It should be noted that the sum of the percentages of each row in the table may not add up to 100% as there are cases where the methods are statistically equivalent. From these results, we can conclude that YAKE! is better than KEA in 70% of the datasets with statistical significance of p-value < 0.01 (▲), and inferior in 10% of the datasets with the same statistical significance. The two methods perform the same in 20% of the cases. As noted previously, YAKE! also performs better than any of the unsupervised approaches, achieving the best results against Rake, TextRank, and ExpandRank, for which YAKE! is better in 100% of the datasets. Even in its weakest performance versus an unsupervised approach, against KP-Miner, YAKE! is still better in more than half of the cases. These results clearly demonstrate the superiority of YAKE when compared to the baselines, including KEA.

To extend our understanding of the effectiveness of YAKE! we also compared it directly against the baseline methods on top of each dataset. Not surprisingly, YAKE! achieves better effectiveness in a large majority of the datasets, achieving 100% success in 110-PT-BN-KP, WWW, KDD, cacic, Fao780, pak2018, and WikiNews. The weakest results are found for the recently launched SemEval2017 dataset, for which YAKE! is still better in 55% of the cases. Of particular importance, YAKE! is very effective not only with English but also when working with Portuguese, French, Spanish, and Polish texts. Overall, these results confirm that YAKE! performs better than almost all the baseline methods across different datasets, text sizes, domains, and languages. This should be viewed as an important contribution to the research community, which lacks any other approach with these characteristics.

To further test these assumptions, we plotted F1@10 (gray bar), P@10 (+ signal), and R@10 (x signal) on top of the oversampling re-sampling technique (see Fig. 7). This provided evidence supporting the claim that YAKE! performs better than all the baselines considered, including KEA. Additional important evidence is that the second-best result was obtained with KP-Miner, which is also a statistical-based approach, while RAKE, one of the best-known state-of-the-art methods of this kind, produced the poorest results of all. This further confirms that none of the graph-based methods (despite being language-dependent) obtain better results than YAKE!.

The very same pattern may be observed in the R-Precision plot (left side of Fig. 8) and Mean Reciprocal Rank (right side of Fig. 8). The former measures the fraction of relevant keywords that are successfully retrieved at the Rth position in the ranking, where R is the total number of relevant keywords per document. The latter measures the reciprocal (inverse) of the rank at which the first relevant keyword is retrieved, which may be a critical point in some information retrieval tasks. Unsurprisingly, the results are significantly better for this second metric, with YAKE! achieving a MRR score of almost 50%. KEA performed slightly better, however, with a score of slightly above 50%.

Table 12
YAKE! effectiveness vs. Baselines for $n = 3$ and $w = 1$. F1@10. Bold face indicates the best results.

Dataset	YAKE!	Unsupervised										Supervised
		Statistical Methods			Graph-based Methods							Binary
		TF.IDF	KP-MINER	RAKE	Text Rank	Single Rank	Topic Rank	Topical PageRank	Position Rank	MultiPartite Rank	Expand Rank	KEA
110-PT-BN-KP	**0.500**	0.231 ▼	0.072 ▼	0.010 ▼	0.207 ▼	0.275 ▼	0.256 ▼	0.186 ▼	0.072 ▼	0.179 ▼	0.221 ▼	0.215 ▼
500N-KPCrowd-v1.1	**0.173**	0.162 ▼	0.093 ▼	0.023 ▼	0.111 ▼	0.157 ▼	0.172	0.158 ▼	0.155 ▼	0.172	0.085 ▼	0.159 ▼
Inspec	0.316	0.155 ▼	0.047 ▼	0.052 ▼	0.098 ▼	**0.378** ▲	0.289 ▼	0.361 ▲	0.350 ▲	0.307 ▼	0.209 ▼	0.150 ▼
Krapivin2009	0.170	0.150 ▼	**0.227** ▲	0.002 ▼	0.121 ▼	0.097 ▼	0.138 ▼	0.104 ▼	0.104 ▼	0.146 ▼	0.027 ▼	0.171
Nguyen2007	0.256	0.225 ▼	**0.314** ▲	0.002 ▼	0.167 ▼	0.158 ▼	0.173 ▼	0.148 ▼	0.143 ▼	0.190 ▼	0.051 ▼	0.221 ▼
PubMed	0.106	0.081 ▼	0.114 △	0.002 ▼	0.071 ▼	0.039 ▼	0.085 ▼	0.052 ▼	0.045 ▼	0.089 ▼	0.012 ▼	**0.216** ▲
Schutz2008	0.196	0.190 ▼	0.230 ▲	0.011 ▼	0.118 ▼	0.086 ▼	**0.258** ▲	0.123 ▼	0.083 ▼	0.238 ▲	0.039 ▼	0.182 ▼
WWW	**0.172**	0.130 ▼	0.037 ▼	0.011 ▼	0.059▼	0.097 ▼	0.067 ▼	0.101 ▼	0.100 ▼	0.077▼	0.065▼	0.072 ▼
KDD	**0.156**	0.115 ▼	0.036 ▼	0.006 ▼	0.050 ▼	0.085 ▼	0.055 ▼	0.089 ▼	0.088 ▼	0.065 ▼	0.059 ▼	0.063 ▼
SemEval2010	0.211	0.177 ▼	**0.261** ▲	0.003 ▼	0.149 ▼	0.129 ▼	0.195 ▽	0.125 ▼	0.133▼	0.199	0.030 ▼	0.215
SemEval2017	0.329	0.181 ▼	0.071 ▼	0.065 ▼	0.125 ▼	**0.449** ▲	0.332	0.443 ▲	0.419 ▲	0.335	0.215 ▼	0.201 ▼
cacic	**0.196**	0.116▼	0.127 ▼	0.044 ▼	0.067▼	0.087 ▼	0.149 ▼	0.009 ▼	0.006 ▼	0.137 ▼	0.030 ▼	0.155 ▼
citeulike180	0.256	0.183 ▼	0.240	0.001 ▼	0.112 ▼	0.066 ▼	0.156 ▼	0.072 ▼	0.061 ▼	0.178 ▼	0.019 ▼	**0.317** ▲
fao30	**0.184**	0.130 ▼	0.183	0.002 ▼	0.077 ▼	0.066 ▼	0.154 ▽	0.107 ▼	0.070 ▼	0.149 ▼	0.021 ▼	0.139 ▼
fao780	**0.187**	0.119 ▼	0.174 ▼	0.000 ▼	0.083 ▼	0.085 ▼	0.137 ▼	0.108 ▼	0.087 ▼	0.145 ▼	0.031 ▼	0.114 ▼
pak2018	**0.086**	0.059 ▼	0.005 ▼	0.016 ▼	0.041 ▼	0.022 ▼	0.022 ▼	0.043 ▼	0.041 ▼	0.019 ▼	0.022 ▼	0.043 ▼
theses100	0.111	0.103	**0.158** ▲	0.002 ▼	0.058 ▼	0.060 ▼	0.114	0.083 ▼	0.069 ▼	0.117	0.025 ▼	0.104
wicc	**0.256**	0.136 ▼	0.145 ▼	0.043 ▼	0.082 ▼	0.133 ▼	0.146 ▼	0.027 ▼	0.017 ▼	0.136 ▼	0.048 ▼	0.167 ▼
wiki20	**0.162**	0.126 ▽	0.156	0.009 ▼	0.074 ▼	0.038 ▼	0.106 ▼	0.059 ▼	0.059▼	0.091 ▼	0.013 ▼	0.134
WikiNews	**0.450**	0.337 ▼	0.227 ▼	0.105 ▼	0.175 ▼	0.248 ▼	0.218 ▼	0.193 ▼	0.113 ▼	0.170 ▼	0.117 ▼	0.248 ▼

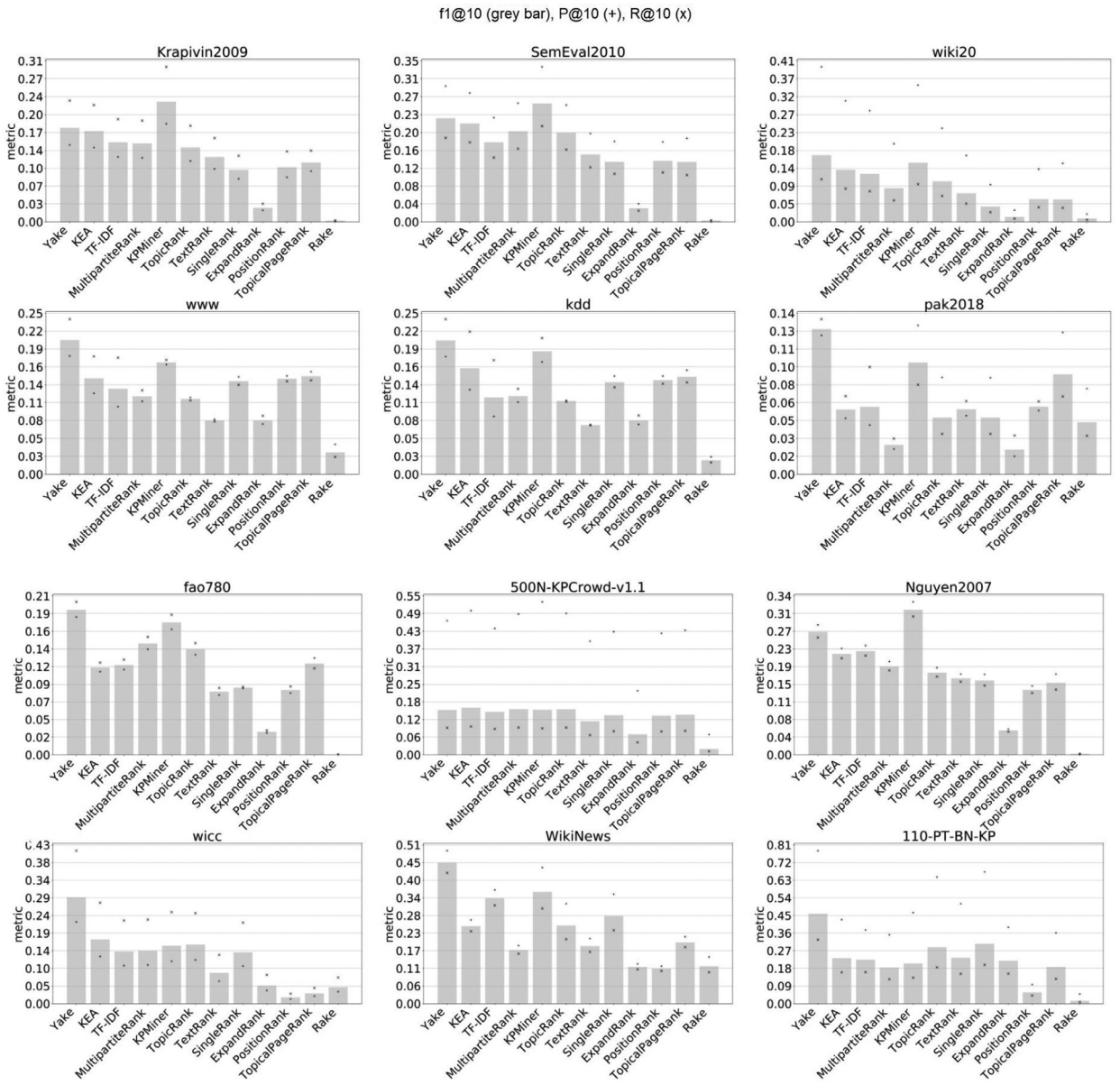


Fig. 6. Effectiveness of YAKE! on top of large (first row) and of short text lengths (second row), different domains (third row) and languages (fourth row).

Table 13
YAKE! effectiveness vs. Baseline methods for $n = 3$ and $w = 1$. Summary table.

Method	YAKE! is better		YAKE! is worse	
	▲	△	▼	▽
TF.IDF	90%	5%	0%	0%
KP-Miner	55%	0%	25%	5%
Rake	100%	0%	0%	0%
TextRank	100%	0%	0%	0%
Single Rank	90%	0%	10%	0%
TopicRank	70%	10%	5%	10%
TopicalPageRank	90%	0%	10%	0%
PositionRank	90%	0%	10%	0%
MultipartiteRank	75%	0%	5%	0%
ExpandRank	100%	0%	0%	0%
KEA	70%	0%	10%	0%

Table 14
YAKE! effectiveness vs Baseline methods under the 20 Datasets for $n = 3$ and $w = 1$. Summary table.

Dataset	YAKE! is better		YAKE! is worst	
	▲	△	▼	▽
110-PT-BN-KP	100%	0%	0%	0%
500N-KPCrowd-v1.1	82%	0%	10%	0%
Inspec	73%	0%	27%	0%
Krapivin2009	82%	0%	9%	0%
Nguyen2007	91%	0%	9%	0%
PubMed	82%	0%	9%	9%
Schutz2008	73%	0%	27%	8%
WWW	100%	0%	0%	0%
KDD	100%	0%	0%	0%
SemEval2010	64%	9%	9%	0%
SemEval2017	55%	0%	27%	0%
cacic	100%	0%	0%	0%
citeulike180	82%	0%	9%	0%
fao30	82%	9%	0%	0%
fao780	100%	0%	0%	0%
pak2018	100%	0%	0%	0%
theses100	55%	0%	9%	0%
wicc	100%	0%	0%	0%
wiki20	73%	9%	0%	0%
WikiNews	100%	0%	0%	0%

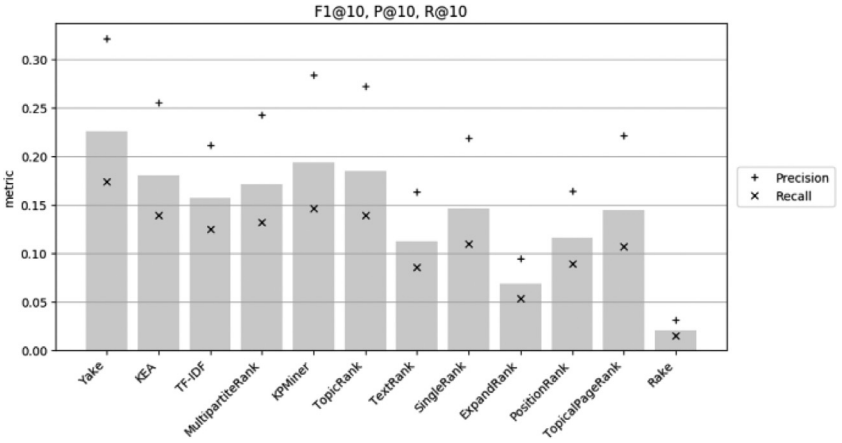


Fig. 7. Effectiveness of YAKE! on top of the oversampled collection. F1@10, P@10 and R@10.

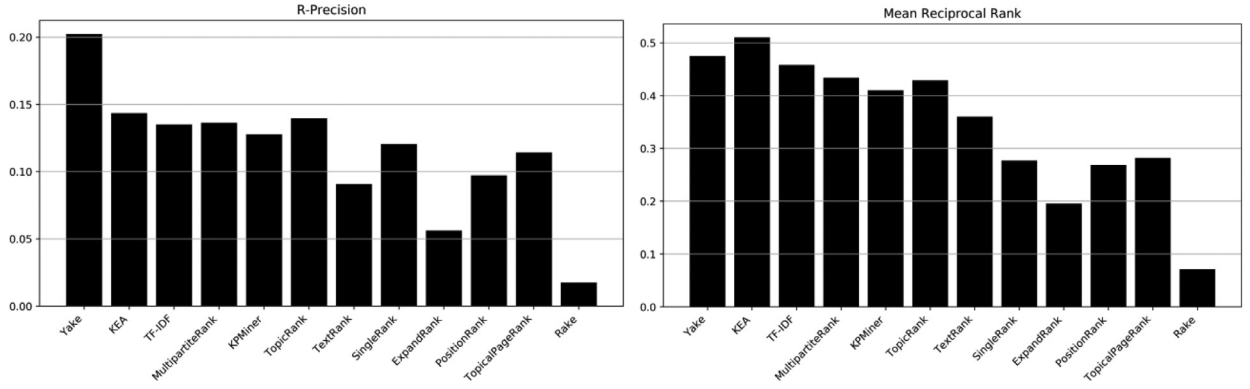


Fig. 8. Effectiveness of YAKE! on top of the oversampled collection. R-Precision (left side) and MRR (right side).

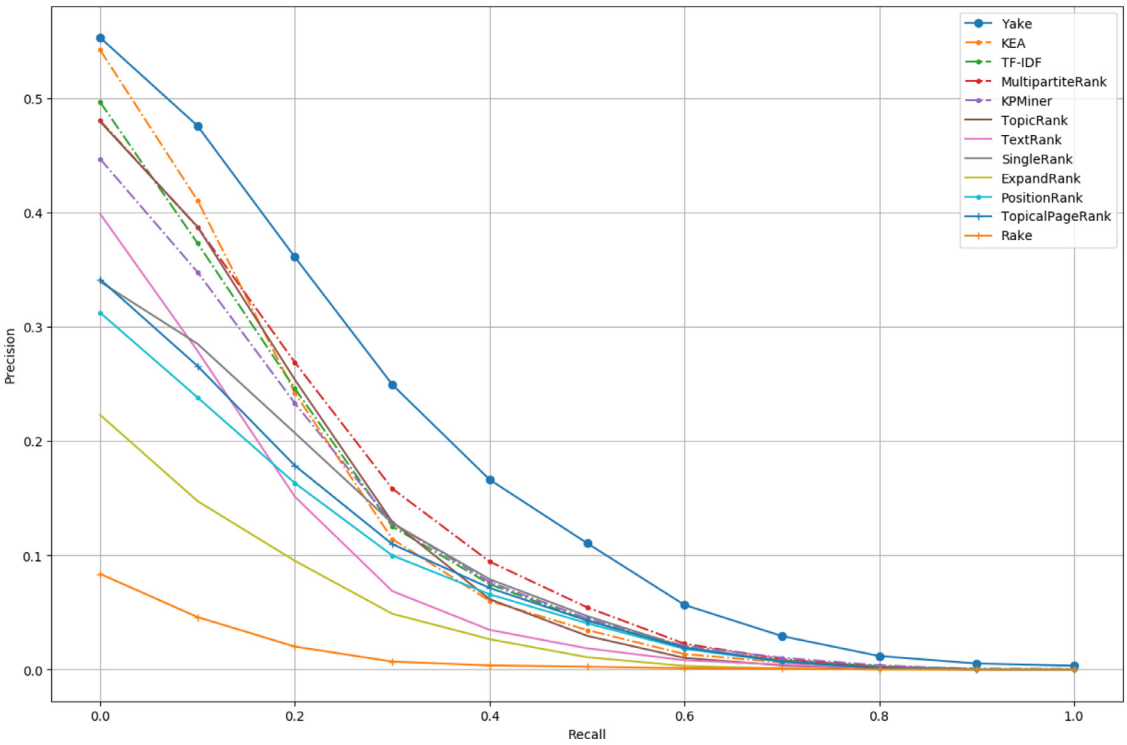


Fig. 9. Effectiveness of YAKE! on top of the oversampled collection. Precision vs. Recall curves.

We conclude this analysis by plotting the precision vs. recall curve for YAKE! and the baselines methods. A careful analysis of Fig. 9 enables us to conclude that YAKE! is able to obtain higher precision in any of the recall cases. The figure also shows that overall, the results deteriorate considerably as recall increases. This is particularly the case for recall scores above 0.5, for which the precision decreases to values below 0.1. We can also observe that while KEA seems to obtain the second-best result, its performance deteriorates sharply for recall values above 0.2. Based on these results we can conclude that YAKE! is an effective alternative method to state-of-the-art algorithms in extracting keywords regardless of the domain, size, or language of the text. Next, we aim to understand the usefulness and importance of each statistical feature of YAKE! and to evaluate the effectiveness of YAKE! versus state-of-the-art methods in the retrieval of keywords containing *interior* stopwords (e.g., "game *of* Thrones"), as this has been a challenging issue within this research area.

6. Feature importance

In this section, we analyze the importance of each statistical feature. For this purpose, we conduct three different analyses. First, we aim to understand the characteristics of the features T_{Case}, $T_{Positional}$, TF_{Norm}, T_{Rel}, and $T_{Sentence}$, and how they behave when they are combined in the single term weight S(t). This will be discussed in Section 6.1. While understanding the distribution of the different features and their contribution to the single term weight might be one important clue to the features' individual merits, measuring their contribution to the final keywords weight assignment S(kw) score also tells us something about their importance. This will be conducted in Section 6.2 through an ablation study. Then, in Section 6.3, we will evaluate the features' importance in terms of the keywords weight assignment over different text lengths, to understand whether different features behave differently depending on the size of the document. Finally, in Section 6.4, we will evaluate the effectiveness of YAKE! and of baseline methods when dealing with interior stopwords.

6.1. Individual feature analysis

In this first set of experiments, our aim was to simply understand the characteristics of each individual feature. Our purpose was to understand if relevant individual terms (i.e., single terms that are part of a gold keyword) tend to differ significantly from non-relevant individual terms (i.e., single terms that are not part of a gold keyword) with respect to their feature scores (T_{Case}, $T_{Positional}$, TF_{Norm}, T_{Rel}, and $T_{Sentence}$). This is an important analysis, not only to understand the features' impact in the current approach but also to understand their potential with regards to a supervised approach that we may eventually design in the future.

Table 15
document #1 (left). document #n (left). Gold Keywords are at the bottom of the table.

Important information: roads will be closed this night. Important! **Gold Keywords**: important information, roads, closed	Information theory is very important for Information Retrieval. **Gold Keywords**: information theory, information retrieval

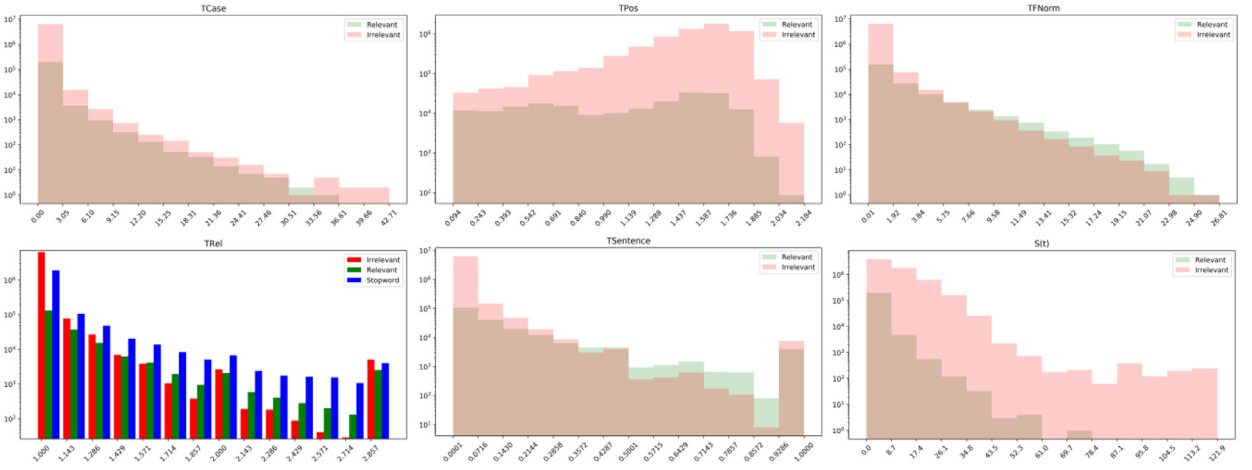


Fig. 10. Analysis of $\mathbf{T_{Case}}$, $\mathbf{T_{Position}}$, $\mathbf{T_{TFNorm}}$, $\mathbf{T_{Rel}}$, $\mathbf{T_{Sentence}}$, $\mathbf{S(t)}$.

In this first experiment, we look at all individual terms of each document of each of the 20 datasets considered in our experiments and check whether each term is relevant or not. Table 15 illustrates this process by listing two excerpts of two documents and their respective gold keywords. A relevant term is a term that appears in a gold keyword of a given document, and an irrelevant term is a non-stopword term that does not appear in a gold keyword of a document. For example, the term "important", which occurs in doc #1 and doc #2, but is only relevant (that is, part of a gold keyword) in doc #1, is considered both a relevant and an irrelevant instance. "Theory" and "retrieval", which only occur in doc #2 as part of a gold keyword, are both considered a relevant instance. The term "night", which occurs in doc #1, is considered an irrelevant instance as it does not appear in any of the gold keywords of doc #1. Finally, the term "be" (and similarly "will", "this", "is", "very", and "for") is considered a stop-word instance. In summary, for all the documents of all 20 datasets, we were able to find 2,109,679 instances of stopwords (23.78%), 241,952 instances that are relevant (2.73%) and 6,520,330 instances that are irrelevant (73.49%). In the following, we analyze the distribution of relevant and irrelevant terms for each feature. That is, we determine the number of relevant and irrelevant instances (y-axis – log scale) that occur within a given interval feature score (x-axis). Note that while this analysis is an important first attempt to understand the behavior of each statistical feature, conclusions about the features' importance should be made with caution, as determining a keyword strongly depends on the features' combination in a single and composed term weight. This will be discussed below.

In these experiments, we began by analyzing T_{Case} (which was introduced in Section 3.2.1). Looking at Fig. 10 we can conclude that, on its own, this feature does not distinguish between relevant and irrelevant terms, as there are a vast number of instances where, despite being uppercase or acronyms, terms are irrelevant.

Next, we evaluated the behavior of the $T_{Position}$ feature (Section 3.2.2), which aims to give a lower value for terms occurring at the beginning of a document (likely to be more relevant). Looking at the figure, we can quickly see that, while the scores of relevant terms seem to remain stable over the x-axis (except for the last two bins), irrelevant terms seem to have a strong tendency to occur with higher values, likely indicative that they are less relevant.

In addition, the histogram plotted for TF_{Norm} (from Section 3.2.3) shows that, although there are far fewer relevant terms than irrelevant ones, there is a tendency for relevant term instances to obtain a higher score than irrelevant term instances, thus corroborating our belief that the higher the frequency is, the more important the candidate term is.

Next, we aim to understand how T_{Rel} (Section 3.2.4) behaves on its own. To this purpose, we also included stopwords in our analysis, as T_{Rel} aims to penalize terms that occur with a high term frequency (as stopwords do) and with many different terms on both their left and right sides. A careful analysis of the bar plot enables us to conclude that a significant majority of the terms (especially irrelevant terms and stopwords) occur in the first bin, between 1 and 1.143. Recall that a value of 1 in this equation will only occur in the case of individual terms that do not appear with any terms on their left or right side (i.e., for which DL and DR = 0), for instance terms that are found within parentheses or that are at the beginning or end of a sentence. Thus, in order to evaluate our assumption that relevant terms should most often have lower scores, we should focus on all the bins but the first. With regards to this, several inferences can be made. First, we can conclude that for the second, third, and fourth bins, which correspond to very low values, there is nearly the same number of relevant terms as irrelevant ones. On the one hand, this may suggest that while there are only around 3% relevant terms and 74% irrelevant

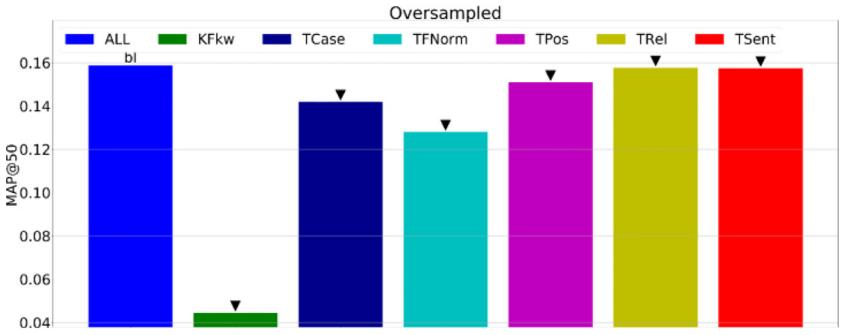


Fig. 11. Evaluating feature importance on top of the oversampled collection.

ones, the significant majority of which are concentrated in the first bin, still we may find a vast number of relevant terms with lower scores than irrelevant terms. On the other hand, we can see that a considerable proportion of irrelevant terms (those concentrated in these bins) occur with several different terms on both sides, but to a small extent (that is with a low term frequency) when compared to the maximum TF (MaxTF). Interestingly, one can also observe that in the last bin (those with higher scores) there are more irrelevant terms and stopwords than relevant terms, in line with the assumption that irrelevant terms should, predominantly, have higher scores. Still, a multitude of relevant terms can be found in this bin. Two reasons can be advanced for this. First, irrelevant terms, beyond being mostly concentrated in the first bin, will hardly occur to a great extent, that is with a TF close to the MaxTF. Second, looking at the right side of the plot we can see that, interestingly, a few relevant terms resemble stopwords, in that they occur as frequently as the MaxTF (usually given by a stopword) and with different terms on both sides, and yet they are relevant terms. One may think of these terms as words that, because of their importance, might appear with several different terms, as is the case of “Information”, which in an Information Retrieval text may be found together with the terms “gain”, “theory”, “retrieval”, “set”, “entropy”, “extraction”, “entity”, “technology”, “overload”, “systems”, “query”, “question”, “science”, “ratio”, “architecture”, “content”, “gap”, “society”, “security”, or “visualization”, to name but a few.

Finally, we aim to understand the behavior of the $T_{Sentence}$ feature. The results clearly show that there is a tendency for irrelevant terms to appear more to the left when compared to relevant terms, thus reflecting the assumption that candidates that appear in many different sentences, which will likely obtain a higher $T_{Sentence}$ score, have a higher probability of being important. An exception to this, however, can be observed in the last bin where a slightly higher number of irrelevant terms can be found. This may be due to the occurrence of stopwords, which tend to occur in every sentence and yet be useless.

While the above analysis is an important first step, to fully understand the importance of each feature we need to look at the two scores S(t) and S(kw) and investigate how they change when a particular feature is removed from the model. First we investigate how the S(t) score behaves. We plotted the results for the S(t) score on the assumption that the lower the score, the more relevant the term will be. Overall, we can see that, as expected, relevant terms have a strong tendency to be concentrated on the left, particularly in the first bin, while there are no relevant terms on the right. In contrast, while there are a large number of irrelevant terms in the first bin, there is clear evidence of the occurrence of irrelevant terms to the right. Thus, while it remains difficult to distinguish between relevant and irrelevant terms when the score is low, one may be confident that the higher the score, the higher the probability that one is dealing with an irrelevant term. In the following section, we investigate the importance of each feature in terms of its contribution to the final S(kw).

6.2. Features' contribution to the generation of keywords

In this section, we estimate the importance of each feature with regard to its contribution to the weights of the keywords S(kw) by conducting an ablation study which follows a backward-like elimination approach, where each feature is individually removed from the single term weight $S(t)$ equation (recall Section 3.3). That is, we consider a zero value for the corresponding feature in the equation $S(t)$ when talking about sums of features, and a 1 value if the feature is to be multiplied by another one. To reach a conclusion, we conduct two experiments: oversampling and a more fine-grained analysis using each dataset. As shown in Fig. 11, removing term-frequency related features such as KF(kw) and TF_{Norm} may negatively impact the results with statistical significance (through a paired t-test considering a p-value $<= 0.01$ (black triangle) or p-value $<= 0.05$ (white triangle)) when compared to the baseline (which considers all the features). The first feature, KF(kw), is applied in the S(kw) equation to balance and give higher weight to those important sequences of terms that frequently appear together. The second, TF_{Norm}, is the individual term frequency feature considered in the S(t) equation. The results shown in Fig. 11 suggest that removing T_{Case}, T_{Pos}, T_{Rel} and T_{Sent}, one feature at a time, may hurt the final results, although to a minimal extent.

To test this assertion, we evaluated the features' behavior under each of the 20 datasets. Fig. 12 confirms that removing the KF(kw) feature negatively impacts the results in all the collections, thus proving once again that relating the relevance of a sequence of terms (as determined by our system) to its frequency in the text plays an important role in our approach.

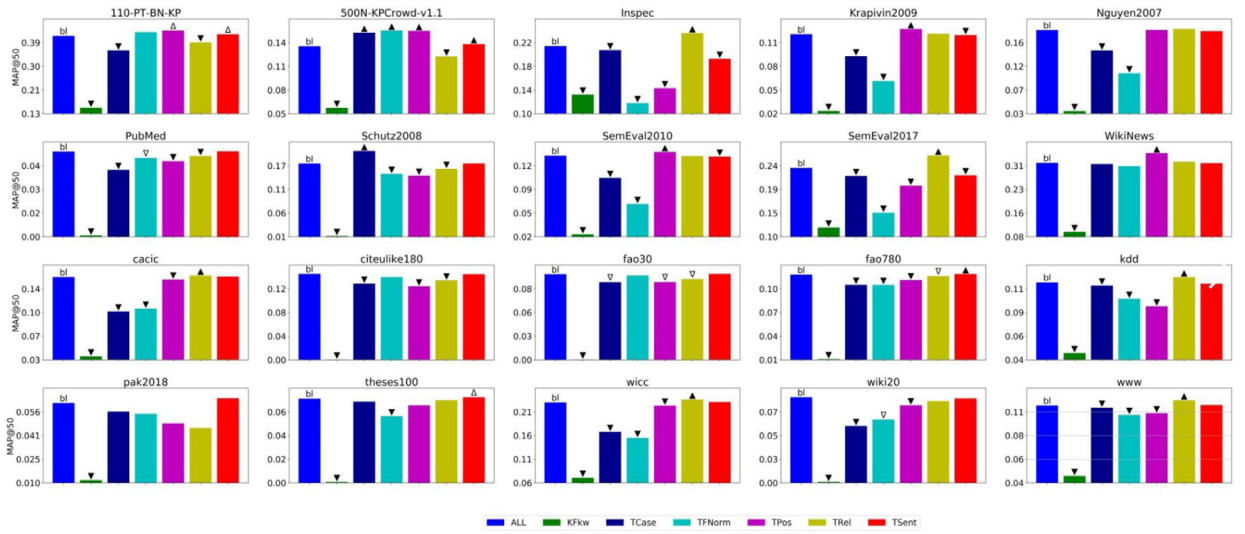


Fig. 12. Evaluating feature importance on top of each collection.

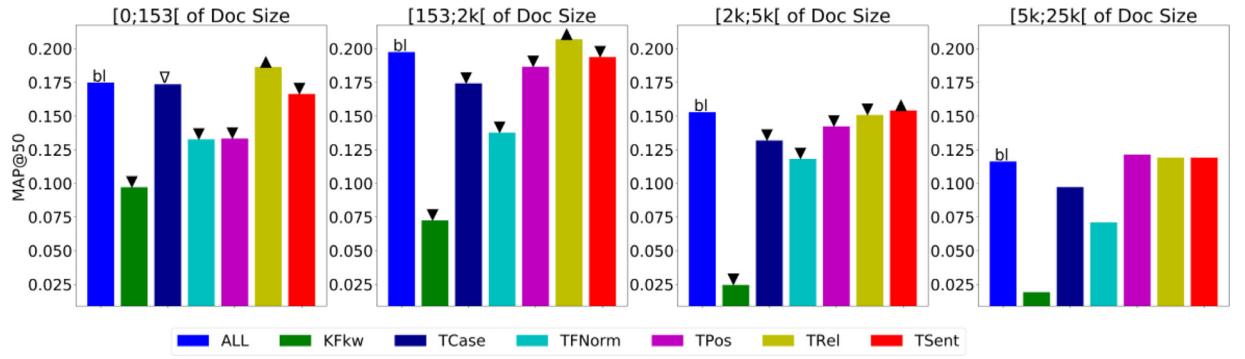


Fig. 13. Evaluating feature importance on top of different text sizes.

Similar behavior can be observed for TF_{Norm}, as removing this feature leads, in most of the cases, to a worsening of the results. The same behavior can be observed for T_{Case} and T_{Pos}, though in these cases to a lesser extent. In contrast, removing T_{Rel} and T_{Sent} (particularly the latter) seems to improve the results in most of the cases, although only to a tiny extent (this is in line with the results observed in Fig. 11). Overall, one may conclude that KF(kw), as well as TF_{Norm}, are the most important features for the keyword assignment. In the following section, we evaluate feature importance with respect to the assignment of keywords over different text lengths to understand whether document size affects different features' behavior.

6.3. Feature importance over different text lengths

In this section, we aim to understand feature importance over different text lengths. To explore this, we consider all the documents of all the collections and split them into four quartiles of fixed length ([0, Q1]; [Q1, Q2]; [Q2, Q3]; [Q3, 24k]) thus ensuring they are not biased by a dominant document size. For instance, [0, 153] means that 25% of all the documents have a size of between 0 and 153 terms, [153, 2k] between 153 and 2k terms, [2k, 5k] between 2k terms and 5k terms, and [5k, 24k] between 5k and 24k terms. Then we test the effectiveness of our method on each quartile. Fig. 13 shows that removing KF(kw), TF_{Norm}, T_{POS} and T_{Case} (one at a time) negatively affects the overall results under the first three quartiles with statistical significance. First, it can be seen that removing KF(kw) considerably weakens the results as document size increases. This is particularly evident for larger documents: see the large difference between the blue bar (all the features) and the green one (all but KF(kw)). This may be understood as due to the fact that keywords will naturally occur more frequently in larger texts. Likewise, removing TF_{Norm} negatively impacts the results, although in this case in the very same proportion no matter the document size; this may be explained by the fact that the terms in this feature are balanced by the mean of the terms plus their standard deviation. In contrast, the effects of removing T_{POS} are mostly noticed for short length texts, as the biggest difference between the blue bar (all the features) and the purple one (all but T_{POS}) occurs for [0, 153]. This indicates that this feature best suits abstract-like texts. We can also observe that the T_{Case} feature is more

Table 16
YAKE and baseline effectiveness in retrieving gold keywords containing interior stopwords.

Method	% of Keywords retrieved that have Stopwords	Precision	Recall	F1-Measure
YAKE-n3	6.76%	0.029	0.104	0.046
YAKE-n4	27.31%	0.010	0.138	0.018
YAKE-n5	42.89%	0.006	0.148	0.012
TF.IDF	33.86%	0.002	0.036	0.004
KP-Miner	6.40%	0.009	0.021	0.013
RAKE	45.51%	0.001	0.143	0.003
TextRank	27.97%	0.001	0.079	0.003
SingleRank	10.79%	0.008	0.038	0.013
TopicRank	7.92%	0.009	0.032	0.014
TopicalPageRank	35.83%	0.003	0.049	0.006
PositionRank	35.13%	0.004	0.056	0.007
MultipartiteRank	7.38%	0.012	0.039	0.018
ExpandRank	21.02%	0.007	0.028	0.011
KEA	9.82%	0.048	0.090	0.062

important on mid-size [153, 2k] and larger documents [2k, 5k], as casing terms are more likely to be found in bigger texts. It is also notable that removing T_{Rel} (yellow roasted color) positively impacts the results for short text documents [0, 153] and mid-text documents [153, 2k] with statistical significance, which can be easily explained by the fact that texts of short length fail to have a sufficiently large corpus to support the computation of the number of different terms that co-occur with the candidate term. From Fig. 13, we can also conclude that T_{Sent} may be best applied on short [0, 153] and mid-size texts [153, 2k], for which removing this feature may degrade the results.

An overall analysis of the results also indicates that the effectiveness of our system tends to be significantly worse when dealing with large texts only; this may be understood to be due to the higher number of available candidate terms in large texts and the resulting complexity of choosing the best ones. Next, we conducted an analysis of how our method and the baselines perform in the retrieval of keywords containing *interior* stopwords.

6.4. Stopword analysis

Finally, we aim to understand how YAKE! and the baseline methods perform in retrieving gold keywords containing *interior* stopwords, as these have always been the subject of special treatment in information retrieval tasks due to their low discrimination value. For this purpose, we conducted two different analyses. First, we compared the effectiveness of YAKE! against baseline methods in retrieving keywords with *interior* stopwords. To do this, we considered three- ($n=3$), four- ($n=4$), and five-term keywords ($n=5$), as keywords with $n=1$ and $n=2$ terms do not have any *interior* stopwords. To conduct this analysis, we began by computing the percentage of gold keywords having at least one stopword for all the datasets considered in our evaluation. The results obtained show that 6.74% of the gold keywords have at least one stopword: 2.3% at the beginning or end of the gold keyword and a further 4.45% in the middle. Based on this, we try to understand the percentage of gold keywords having stopwords retrieved with each method. Table 16 lists these values. We can observe that YAKE-n3 retrieves almost the same number of keywords (having stopwords) as the ground-truth does. This contrasts with the results of RAKE, TF.IDF, TopicalPageRank, PositionRank, and TextRank, where a considerable number of keywords containing stopwords are retrieved. A careful analysis of these results enables us to conclude, however, that most of these are not gold keywords, as the recall is quite low in both approaches. Overall, we can conclude that the best methods for retrieving keywords containing *interior* stopwords are KEA and our method (YAKE-n3), with F1 scores of 0.062 and 0.046, respectively. Indeed, while only 0.029 (2.9%) of the gold keywords with stopwords retrieved by YAKE-n3 are effectively relevant, 10.4% of all (correct) gold standard keywords with stopwords in the dataset were retrieved by YAKE!, which is an interesting number, considering that only 6.76% of the keywords retrieved by our system have stopwords.

As a further analysis, we decided to evaluate the merits of our bigram probability stopword approach (discussed in Section 3.4) compared to not using it. To do this, we compared the results of YAKE! under two different versions: (1) an intermediate version which, instead of applying the bigram probability, considers the original S(t) score of the stopword; and (2) a version named "NoStopwordScore" where the scores of *interior* stopwords are not taken into account. This means that, when dealing with a 3-gram that has an *interior* stopword, we only take into account the scores of the two terms that are not stopwords. Fig. 14 plots the effectiveness of YAKE! for MAP@50 on top of the oversampled collection, under these two approaches. Note that again 1- and 2-grams are not considered as we do not consider keywords beginning or ending with a stopword. Looking at the figure, we can observe that the current bigram probability approach is the most appropriate solution; in the case of either applying the S(t) scores or not considering the scores of stopwords (NoStopwordScore), the results are degraded considerably.

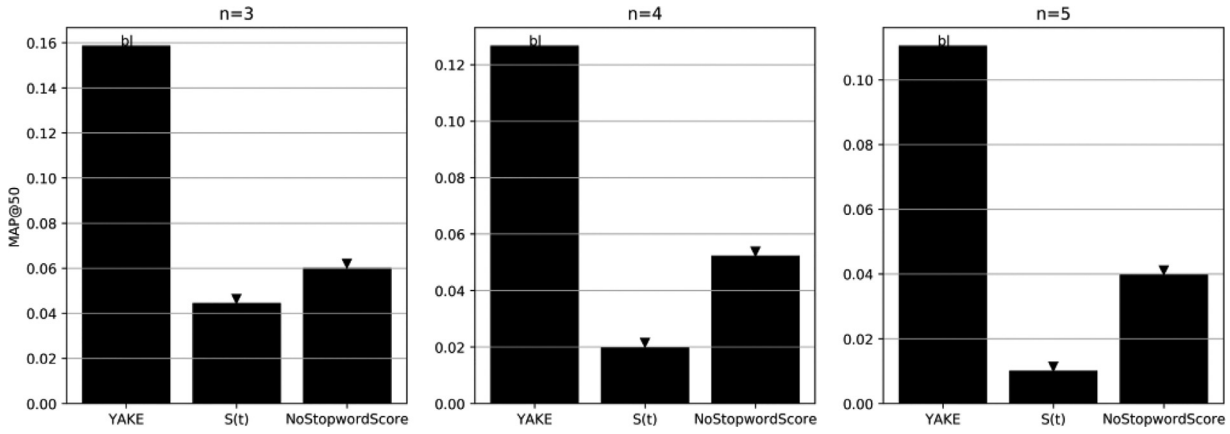


Fig. 14. Effectiveness of YAKE! when comparing different stopword approaches for $3 \leq n \leq 5$ on top of the oversampled collection.

7. Conclusion and future work

Keyword extraction algorithms have become the key components in many computer science applications. In this article, we have presented YAKE!, a novel feature-based online system for multi-lingual keyword extraction from individual documents that can be used both as a stand-alone tool and as a support for several applications, including summarization, clustering, indexing, and information visualization, to name just a few. YAKE! follows an *unsupervised* approach. It does not require a corpus or a dictionary (*corpus independent*), which means it can be quickly adapted across different languages or domains (*domain and language independent*). Our solution *scales* to any document length in a linear manner in the number of candidate terms identified and is *term frequency-free*, meaning that no conditions are set with respect to the minimum frequency or sentence frequency that a candidate keyword must have. Our experiments confirm that YAKE! can lead to superior keyword extraction results compared with ten popular state-of-the-art unsupervised keyword extraction algorithms, under a large number of text documents belonging to twenty different datasets. It also offers similar or superior effectiveness to state-of-the-art supervised methods, without the need for an extensive training set for multiple languages and domains. As a further contribution, we provide an app on Google Play, a demo, an API, and a python package so that YAKE! can be tested and used by the research community. YAKE! has already been used for temporal clustering and ranking, and won first prize at the Arquivo.pt 2018 Portuguese internet archive contest with "Tell me Stories" [archive.tellmestories.pt], a news summarization platform which generates a timeline summary for a given news topic in a storytelling-style fashion. In the future, we plan to develop a supervised solution, which will be built using the statistical features introduced in this work. An analysis of the importance of each feature has been already conducted in this research. Furthermore, we are currently working on an extended version of YAKE! to tackle the problem of absent keywords, for instance when a large number of manually-assigned keywords cannot be found in the text itself. We plan to do this based on word embeddings.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research was funded by Project "TEC4Growth - Pervasive Intelligence, Enhancers and Proofs of Concept with Industrial Impact/NORTE-01–0145-FEDER-000020", which is financed by the North Portugal Regional Operational Program (NORTE 2020), under the PORTUGAL 2020 Partnership Agreement, as well as by the European Regional Development Fund (ERDF) through the Compete 2020 Program within project POCI-01–0145-FEDER-006961, and by National Funds through the Portuguese funding agency, FCT - Fundação para a Ciência e a Tecnologia within project: UID/EEA/50014/2019 and UID/MAT/00212/2019.

References

[1] I. Augenstein, M. Das, S. Riedel, L. Vikraman, A. McCallum, SemEval 2017 task 10: scienceIE - Extracting Keyphrases and relations from scientific publications, SemEval'17. Vancouver, Canada. August 3-4 (2017) 546–555.
[2] G. Aquino, L. Lanzarini, Keyword identification in Spanish documents using neural networks, J. Comp. Sci. Tech. 15 (2) (2015) 55–60.
[3] S. Beliga, A. Meštrović, S. Martinčić-Ipšić, Selectivity-based keyword extraction method, Int. J. Semantic Web Inform. Syst. (IJSWIS) 12 (3) (2016) 1–26.
[4] A. Bougouin, F. Boudin, B. Daille, TopicRank: graph-Based topic ranking for keyphrase extraction, in: Proceedings of the 6th International Joint Conference on Natural Language Processing (IJCNLP'13), Nagoya, Japan, 2013, pp. 543–551. October 14-18.

[5] F. Boudin, PKE: an open source python-based keyphrase extraction toolkit, in: Proceedings of the 26th International Conference on Computational Linguistics (COLING'16), Osaka, Japan, 2016, pp. 69–73. December 13-16.
[6] F. Boudin, Unsupervised keyphrase extraction with multipartite graphs, in: 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT'18), USA, 2018, pp. 667–672. June 1-6.
[7] R. Campos, G. Dias, A.M. Jorge, C. Nunes, Identifying top relevant dates for implicit time sensitive queries, In Inform. Retriev. J. Springer 20 (4) (2017) 363–398.
[8] R. Campos, G. Dias, A.M. Jorge, C. Nunes, GTE-Rank: a time-aware search engine to answer time-sensitive queries, Inform. Process. Manage. Int. J. Elsevier 52 (2) (2016) 273–298.
[9] R. Campos, V. Mangaravite, A. Pasquali, A.M. Jorge, C. Nunes, A. Jatowt, YAKE! collection-independent automatic keyword extractor, in: G. Pasi, B. Piwowarski, L. Azzopardi, A. Hanbury (Eds.), Advances in Information Retrieval. ECIR 2018, 10772, Lecture Notes in Computer Science, Grenoble, France, 2018, pp. 806–810. March 26 – 29.
[10] R. Campos, V. Mangaravite, A. Pasquali, A.M. Jorge, C. Nunes, A. Jatowt, A text feature based automatic keyword extraction method for single documents, in: G. Pasi, B. Piwowarski, L. Azzopardi, A. Hanbury (Eds.), Advances in Information Retrieval. ECIR 2018, 10772, Lecture Notes in Computer Science, Grenoble, France, 2018, pp. 684–691. March 26 – 29.
[11] C. Caragea, F. Bulgarov, A. Godea, S. Gollapalli, Citation-enhanced keyphrase extraction from research papers: a supervised approach, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP'14), Doha, Qatar, 2014, pp. 1435–1446. October 26–28.
[12] W. Cohen, P. Ravikumar, S. Fienberg, A comparison of string metrics for matching names and records, in: Proceedings of the International Workshop on Data Cleaning and Object Consolidation (DCOC@KDD), Washington, USA, 2003, pp. 88–96. August 24-27.
[13] S. El-Beltagy, A. Rafea, KP-miner: a keyphrase extraction system for English and Arabic documents, In Inform. Syst. Elsevier 34 (1) (2009) 132–144.
[14] C. Florescu, C. Caragea, A new scheme for scoring phrases in unsupervised keyphrase extraction, in: Proceedings of the 39th European Conference on Information Retrieval (ECIR'17), Aberdeen, Scotland, 2017, pp. 477–483. April 9-13.
[15] C. Florescu, C. Caragea, PositionRank: an unsupervised approach to keyphrase extraction from scholarly documents, in: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL'17), Vancouver, Canada, 2017, pp. 1105–1115. July 20 – August 4.
[16] S. Gollapalli, C. Caragea, Extracting keyphrases from research papers using citation networks, in: Proceedings of the 28 th AAAI Conference on Artificial Intelligence (AAAI'14), Québec City, Québec, Canada, 2014, pp. 1629–1635. July 27-31.
[17] S. Gollapalli, X.-.L. Li, P. Yang, Incorporating expert knowledge into keyphrase extraction, in: Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI'17), San Francisco, USA, 2017, pp. 3180–3187. February 4-9.
[18] D. Gomes, D. Cruz, J. Miranda, Search the past with the Portuguese web archive, in: Proceedings of the 22nd International Conference on World Wide Web (WWW'13), Rio de Janeiro, Brazil, 2013, pp. 321–324. May 13-17.
[19] M. Grineva, M. Grinev, D. Lizorkin, Extracting key terms from noisy and multitheme documents, in: Proceedings of the 18th International Conference on World Wide Web (WWW'09), Madrid, Spain, 2009, pp. 661–670. April 20-24.
[20] K. Hasan, V. Ng, Automatic keyphrase extraction: a survey of the state of the art, in: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL'14), Baltimore, Maryland, USA, 2014, pp. 1262–1273. June 22-27.
[21] A. Hulth, Improved automatic keyword extraction given more linguistic knowledge, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'03), Sapporo, Japan, 2003, pp. 216–223. July 11-12.
[22] K.S. Jones, A statistical interpretation of term specificity and its application in retrieval, J. Document. 28 (1) (1972) 11–21.
[23] M. Jorge A, R. Campos, A. Jatowt, S. Nunes, Special issue on narrative extraction from texts (Text2Story): preface, Inform. Process. Manage. Int. J. Elsevier 56 (5) (2019) 1771–1774.
[24] V. Levenshtein, Binary codes capable of correcting deletions, insertions, and reversals, Soviet Phys. Doklady 10 (8) (1966) 707–710.
[25] Z. Liu, W. Huang, Y. Zheng, M. Sun, Automatic keyphrase extraction via topic decomposition, in: Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP '10), Cambridge, Massachusetts, USA, 2010, pp. 366–376. October 09-11.
[26] H.P. Luhn, The automatic creation of literature abstracts, In IBM J. Res. Dev. 2 (2) (1958) 159–165.
[27] S. Kim, O. Medelyan, M.-.Y. Kan, T. Baldwin, SemEval-2010 task 5: automatic keyphrase extraction from scientific articles, in: Proceedings of the 5th International Workshop on Semantic Evaluation (SemEval'10), Los Angeles, USA, 2010, pp. 21–26. July 15-16.
[28] M. Krapivin, A. Autaeu, M. Marchese, Large Dataset for Keyphrases Extraction, University of Trento, 2009 Tech Report # DISI-09-055.
[29] D. Machado, T. Barbosa, S. Pais, B. Martins, G. Dias, Universal mobile information retrieval, in: Proceedings of the 13th International Conference on Human Computer Interaction (HCII'09), San Diego, USA, 2009, pp. 345–354. July 19-24.
[30] D. Mahata, J. Kuriakose, R. Shah, R. Zimmermann, Key2Vec: automatic ranked keyphrase extraction from scientific articles using phrase embeddings, in: 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT'18), New Orlean, USA, 2018, pp. 634–639. June 1-6.
[31] Marujo L., Viveiros M., Neto J. (2013) Keyphrase cloud generation of broadcast news. In: arXiv preprint arXiv:1306.4606
[32] Marujo L., Gershman A., Carbonell J., Frederking R., Neto J. (2013) Supervised topical key phrase extraction of news stories using crowdsourcing, light filtering and co-reference normalization. In: arXiv preprint arXiv:1306.4886
[33] O. Medelyan, I. Witten, D. Milne, Topic indexing with wikipedia, in: Proceedings of the Workshop on Wikipedia and Artificial Intelligence: An Evolving Synergy (WikiAI@AAAI), Illinois, UK, 2008, pp. 19–24. July 13–17.
[34] O. Medelyan, E. Frank, I. Witten, Human-competitive tagging using automatic keyphrase extraction, in: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP '09), Singapore, 2009, pp. 1318–1327. August 6-7.
[35] O. Medelyan, I. Witten, Domain-independent automatic keyphrase indexing with small training sets, J. Assoc. Inf. Sci. Technol. 59 (7) (2008) 1026–1040.
[36] R. Meng, S. Zhao, S. Han, D. He, P. Brusilovsky, Y. Chi, Deep keyphrase generation, in: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL'17), Vancouver, Canada, 2017, pp. 582–592. July 30-August 4.
[37] R. Mihalcea, P. Tarau, TextRank: bringing order into texts, in: Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP '04), Barcelona, Spain, 2004, pp. 404–411. July 25-26.
[38] T. Nguyen, M.-.Y. Kan, Keyphrase extraction in scientific publications, in: Proceedings of the 10th International Conference on Asian Digital Libraries (ICADL'07), Hanoi, Vietnam, 2007, pp. 317–326. December 10-13.
[39] A. Pasquali, V. Mangaravite, R. Campos, A. Jorge, A. Jatowt, Interactive system for automatically generating temporal narratives, in: L. Azzopardi, B. Stein, N. Fuhr, P. Mayr, C. Hauff, D. Hiemstra (Eds.), Advances in Information Retrieval. ECIR'19, 11438, Lecture Notes in Computer Science, Cologne, Germany, 2019, pp. 251–255. April 14 – 18.
[40] E. Papagiannopoulou, G. Tsoumakas, Local word vectors guiding keyphrase extraction, Inform. Process. Manage. Int. J. Elsevier 54 (6) (2018) 888–902.
[41] Papagiannopoulou E., Tsoumakas G. (2019) A review of keyphrase extraction. https://arxiv.org/abs/1905.05044v1
[42] S. Rose, D. Engel, N. Cramer, W. Cowley, Automatic keyword extraction from individual documents, in: M.W. Berry, J. Kogan (Eds.), Text Mining: Theory and Applications, John Wiley & Sons, 2010.
[43] G. Salton, C.S. Yang, C.T. Yu, A theory of term importance in automatic text analysis, J. Am. Soc. Inf. Sci. 26 (1) (1975) 33–44.
[44] T. Schutz A, Keyphrase Extraction from Single Documents in the Open Domain Exploiting Linguistic and Statistical Methods Master Thesis., National University of Ireland, 2008.
[45] A. Spink, D. Wolfram, M. Jansen, T. Saracevic, Searching the web: the public and their queries, In J. Am. Soc. Inform. Sci. Tech. 52 (3) (2011) 226–234.
[46] L. Sterckx, T. Demeester, J. Deleu, C. Develder, Topical word importance for fast keyphrase extraction, in: Proceedings of the 24th International Conference on World Wide Web (WWW'15), Florence, Italy, 2015, pp. 121–122. May 18–22.
[47] P. Turney, Learning algorithms for keyphrase extraction, Inform. Retriev. J. Kluwer Academic Publishers 2 (4) (2000) 303–336.

[48] X. Wan, J. Xiao, Single document keyphrase extraction using neighborhood knowledge, in: Proceedings of the 23 rd Conference on Artificial Intelligence (AAAI'08), Chicago, US, 2008, pp. 855–860. July 13-17.
[49] R. Wang, W. Liu, C. McDonald, Corpus-independent generic keyphrase extraction using word embedding vectors, in: Proceedings of the Software Engineering Research Conference, 39, 2014.
[50] I. Witten, G. Paynter, E. Frank, C. Gutwin, C. Nevill-Manning, KEA: practical automatic keyphrase extraction, in: Proceedings of the 4th ACM Conference on Digital Libraries (JCDL'04), Tucson, US., 1999, pp. 254–255. June 7-11.