Exploring Linguistically-Lightweight Keyword Extraction Techniques for Indexing News Articles in a Multilingual Set-up

Jakub Piskorski
Institute for Computer Science
Polish Academy of Sciences
Warsaw, Poland
jpiskorski@gmail.com

Nicolas Stefanovitch
European Commission,
Joint Research Centre (JRC)
Ispra, Italy
nicolas.stefanovitch@ec.europa.eu

Guillaume Jacquet
European Commission,
Joint Research Centre (JRC)
Ispra, Italy
guillaume.jacquet@ec.europa.eu

Aldo Podavini
European Commission,
Joint Research Centre (JRC)
Ispra, Italy
aldo.podavini@ec.europa.eu

Abstract

This paper presents a study of state-of-the- art unsupervised and linguistically unsophisticated keyword extraction algorithms, based on statistic-, graph-, and embedding-based approaches, including, i.a., Total Keyword Frequency, TF-IDF, RAKE, KPMiner, YAKE, KeyBERT, and variants of TextRank-based keyword extraction algorithms.

The study was motivated by the need to select the most appropriate technique to extract keywords for indexing news articles in a real-world large-scale news analysis engine.

The algorithms were evaluated on a corpus of circa 330 news articles in 7 languages. The overall best F_1 scores for all languages on average were obtained using a combination of the recently introduced YAKE algorithm and KPMiner (20.1%, 46.6% and 47.2% for exact, partial and fuzzy matching resp.).

1 Introduction

Keyword Extraction (KE) is the task of automated extraction of single or multiple-token phrases from a textual document that best express all key aspectsof its content and can be seen as automated generation of a short document summary. It constitutes an enabling technology for document indexing, clustering, classification, summarization, etc.

This paper presents a comparative study of the performance of some state-of-the-art unsupervised linguistically-lightweight keyword extraction methods and combinations thereof applied on news articles in seven languages. The main drive behind the reported work was to explore the usability of these methods for adding another level of indexing of news articles gathered and analysed by the Europe Media Monitor (EMM)[1] (Steinberger et al., 2017), a large-scale multilingual real-time news gathering and analysis system, which processes an average of 300,000 online news articles per day in up to 70 languages and is serving several EU institutions and international organisations.

While a vast bulk of research and tools for KE have been reported in the past, the specific focus of our research was to select the most suitable KE methods for indexing news articles taking specifically into account the operational, multilingual and real-time processing character of EMM. Hence, only unsupervised, scalable vis-a-vis multilinguality and robust algorithms that do not require any sophisticated linguistic resources and are capable of processing single news article in a time-efficient manner were considered.

Keyword extraction has been the subject of research for decades. Both unsupervised and supervised approaches exist, the unsupervised being particularly popular due to the scarcity of annotated data as well as their domain independence.

The unsupervised approaches are usually divided in three phases: (a) selection of candidate tokens that can constitute part of a keyword using some heuristics based on statistics and/or certain linguistic features (e.g., belonging to a specific part-of-speech or not being a stop word, etc.), (b) rank-

[1]https://emm.newsbrief.eu/

Proceedings of the EACL Hackashop on News Media Content Analysis and Automated Report Generation, pages 35–44
April 19, 2021 © Association for Computational Linguistics

ing the selected tokens, and (c) generating keywords out of the selected tokens, where the final rank is computed using the scores of the individual tokens. The unsupervised methods are divided into: statistics-, graph-, embeddings- and language model-based ones. The statistics-based methods exploit frequency, positional and co-occurrence statistics in the process of selecting candidate keywords. The graph-based methods create a graph from textual documents with nodes representing the candidate keywords and edges representing some relatedness to other candidate keywords, and then deploy graph ranking algorithms, e.g. PageRank, TextRank, to rank the final set of keywords. Recently, a third group of methods emerged which are based on word (Mikolov et al., 2013) and sentence embeddings (Pagliardini et al., 2018). Linguistic sophistication constitutes another dimension to look at the keyword extraction algorithms. Some of the methods use barely any language-specific resources, e.g., only stop word lists, whereas others exploit part-of-speech tagging or even syntactic parsing.

The supervised methods are simply divided into shallow and deep learning methods. The shallow methods exploit either binary classifiers to decide whether a token sequence is a keyword, linear regression-based models to rank the candidate keywords, and sequence labelling techniques. The deep learning methods exploit encoder-decoder and sequence-to-sequence labelling approaches. Most of the supervised machine-learning approaches reported in the literature deploy more linguistic sophistication (i.e., linguistic features) vis-a-vis unsupervised methods.

Extensive surveys on keyword extraction methods and comparison of their relative performance are provided in (Papagiannopoulou and Tsoumakas, 2020; Hasan and Ng, 2014; Kilic and Cetin, 2019; Alami Merrouni et al., 2019).

Since only a few monolingual corpora with keyword annotation of news articles exist (Marujo et al., 2013, 2012; Bougouin et al., 2013) that use different approaches to keyword annotation, we have created a new multilingual corpus of circa 330 news articles annotated with keywords covering 7 languages which is used for evaluation purposes in our study. We are not aware of any similar multilingual resource available for research purposes.

The paper is organized as follows. First, Section 2 introduces the Keyword Extraction task for news article indexing. Section 3 gives an overview of the methods explored. Next, Section 4 describes the creation of a multi-lingual data set and experiment results. Finally, we end up with conclusions and an outlook on future work in Section 5.

2 Keyword Extraction Task

The purpose of KE might vary depending on the domain in which it is deployed. In media monitoring and analysis the main objective is to capture from the text of each news article the main topics discussed therein, the key events reported, the entities involved in these events and what is the outcome, impact and significance thereof. For the sake of specifying what the expected output of KE should be, and in order to guide human annotators tasked to create test datasets, the following constraints on keyword selection were introduced (here in simplified form):

- a keyword can be a single word or a sequence of **up to 5 consecutive words** (unless it is a long proper name) **as they appear in the news article or the title** thereof,
- a **minimum of 5** and ideally **not more than 15** keywords (with ca 30% margin - to provide some flexibility) should be selected, however the set of selected keywords **may not** constitute more than 50% of the body of the news article,
- a single keyword **may not** include more than **one** entity,
- a keyword has to be either a **noun phrase**, **proper name**, **verb**, **adjective**, **phrasal verb**, or **part of a clause** (e.g., '*Trump died*'),
- a **stand-alone** adverb, conjunction, determiner, number, preposition or pronoun **may not** constitute a keyword,
- **a full sentence can never constitute a keyword**,
- keywords **should not** be converted into their corresponding base forms, disregarding the fact that a base form would appear more natural,
- if there are many candidate keywords to represent the same concept, only one of them should be selected.

3 Methods

Given the specific context of real-time media monitoring, our experiments imposed the following main selection criteria to the keyword extraction techniques to explore and evaluate:

- **efficiency:** ability to process a single news article within a fraction of a second,
- **multi-linguality:** ability to quickly adapt the method to the processing of many different languages,
- **robustness:** ability to process corrupted data without impacting performance.

Consequently, we have selected methods that: (a) do not require any language-specific resources except stop word lists and off-the-shelf pre-computed word embeddings, (b) exploit only information that can be computed in a time-efficient manner, e.g., frequency statistics, co-occurrence, positional information, string similarity, etc., (c) do not require any external text corpora (with one exception for a baseline method). The pool of methods (and variants thereof) explored includes:

Total Keyword Frequency (TKF) exploits only frequency information to rank candidate keywords, where candidates are 1-3 word n-grams from text that do not contain punctuation marks, and which neither start nor end with a stop word.

Term Frequency–Inverse Document Frequency (TF-IDF) constitutes the main baseline algorithm in our study. For the computation of TF-IDF scores a corpus consisting of 34.5M news articles gathered by EMM that span over the first 6 months of 2020and covering ca. 70 languages was exploited.[2] A maximum of $\min(20, N/6)$ keywords with high-est TF-IDF scores are returned for a news article, where N stands for the total number of tokens in the article.

Rapid Automatic Keyword Extraction (RAKE) exploits both frequency and co-occurrence information about tokens to score candidate keyword phrases (token sequences that do contain neither stop words nor phrase delimiters) (Rose et al., 2010). More specifically, the score for a candidate keyword phrase is computed as the sum of its member word scores. We explored three options for scoring words: (a) $s(w) = frequency(w)$ (RAKE-FREQ), (b) $s(w) = degree(w)$ (RAKE-DEG), which stands for the number of other content words that co-occurr with w in any candidate keyword phrase, and (c) $s(w) = degree(w)/frequency(w)$ (RAKE-DEGFREQ).

Keyphrase Miner (KP-Miner) exploits frequency and positional information about candidate keywords (word n-grams that do not contain punctuation marks, and which neither start nor end with a stop word) with some weighting of multi-token keywords (El-Beltagy and Rafea, 2009). More precisely, the score of a candidate keyword (in the case of single document scenario) is computed as:

$$s(k) = freq(k) \cdot \max\left(\frac{|K|}{\alpha \cdot |K_m|}, \omega\right) \cdot \frac{1}{AvgPos(k)}$$

where $freq(k)$, K, K_m denote frequency of k, the set of all candidate keywords and the set of all multi-token candidate keywords resp., whereas α and ω are two weight adjustment constants, and $AvgPos(k)$ denotes the average position of the keyword in a text in terms of regions separated by punctuations. KP-Miner also has a specific *cut-off* parameter, which determines the number of tokens after which if the keyword appears for the first time it is filtered out and discarded as a candidate. Our version of KP-Miner does not include stemming different from the original one (El-Beltagy and Rafea, 2009) due to our multilingual context and the specification of KE task (see Section 2). Finally, KP-Miner scans the top n ranking candidates and removes the ones which constitute sub-parts of others and adjusts the scores accordingly. Based on the empirical observations the specific parameters, namely, α, ω and *cut-off* were set to 1.0, 3.0 and 1000 resp.

Yet Another Keyword Extraction (Yake) exploits a wider range of features (Campos et al., 2020) vis-a-vis RAKE and KP-Miner in the process of scoring single tokens. Like the two algorithms introduced earlier, YAKE selects as candidate keywords word n-grams that do not contain punctuation marks, and which neither start nor end with a stop word. However, on top of this, an additional token classification step is then carried out in order to filter out additional tokens that should

[2]In particular, the pool of 34.5M news articles included: 11309K English, 6746K Spanish, 2322K French, 2001K Italian, 1431K German, 760K Romanian and 183K Polish articles, which covers the languages of the evaluation dataset (see Section 4.1).

not constitute part of a keyword (e.g. non alphanumeric character sequences, etc.). Single tokens are scored using the following formula:

$$Score(t) = \frac{T_{rel-context}(t) \cdot T_{position}(t)}{T_{case}(t) + \frac{T_{freq-norm}(t)+T_{sentence}(t)}{T_{rel-context}(t)}}$$

where: (a) $T_{case}(t)$ is a feature that reflects statistics on case information of all occurrences of t based on the assumption that uppercase tokens are more relevant than lowercase ones, (b) $T_{position}(t)$ is a feature that exploits positional information and boosts tokens that tend to appear at the beginning of a text, (c) $T_{freq-norm}$ is a feature that gives higher value to tokens appearing more than the mean and balanced by the span provided by standard deviation, (d) $T_{sentence}(t)$ is a feature that boosts significance of tokens that appear in many different sentences, and (e) $T_{rel-context}(t)$ is a *relatedness to context* indicator that 'downgrades' tokens that co-occur with higher number of unique tokens in a given window (see (Campos et al., 2020) for details). The score for a candidate keyword $k = t_1 t_2 \ldots t_n$ is then computed as:

$$Score(k) = \frac{\prod_{i=1}^{n} Score(t_i)}{frequency(k) \cdot (1 + \sum_{i=1}^{n} Score(t_i))}$$

Once the candidate keywords are ranked, potential duplicates are removed by adding them in relevance order. When a new keyword is added it is compared against all more relevant candidates in terms of semantic similarity, and if this similarity is below a specified threshold it is discarded. While the original YAKE algorithm exploits for this purpose the Levenshtein distance, our implementation uses *Weighted Logest Common Substrings* string distance metric (Piskorski et al., 2009) which favours overlap in the initial part of the strings compared.

Embedding-based Keyword Extraction (KEYEMB) exploits document embeddings and cosine similarity in order to identify candidate keywords. First, a document embedding is computed, then word n-grams of different sizes are generated, which are subsequently ranked along their similarity to the embedding of the document (Grootendorst, 2020).

We tested three different out-of-the-box transformer-based sentence embeddings. BERT-based ones are taken from (Reimers and Gurevych, 2020), which are both multilingual and fine-tuned on natural language inference and semantic text similarity tasks. One version uses a basic BERT model (KEYEMB-BERT-B) and the other a lightweight BERT model (KEYEMB-BERT-D). Finally, KEYEMB-LASER is based on LASER (Artetxe and Schwenk, 2019) embeddings. Contrary to BERT, they have not been fine-tuned on semantic similarity tasks, but for the task of aligning similar multilingual concepts to the same semantic space.

Filtering stop words without applying any of the different post-processing steps proposed in (Grootendorst, 2020) provided the best results and therefore is the setting we used in the evaluation and comparison against other methods.

Graph-based Keyword Extraction: (GRAPH) exploits properties of a graph whose nodes are substrings extracted from the text in order to identify which are the most important (Litvak and Last, 2008). This approach differs from TextRank (Mihalcea and Tarau, 2004), in two ways: firstly, the graph is constructed in a fundamentally different way yielding smaller graphs and therefore faster processing time; secondly, different lower-complexity graph measures are also explored, allowing even faster processing time.

A node of the graph corresponds either to a sentence, a phrase delimited by any punctuation marks or a token sequence delimited by stop words. Two nodes are connected only if they share at least 20% of words after removal of stop words.

The importance of the nodes can be defined in different ways. In this study we looked at: (a) *degree* (GRAPH-DEGREE), which measures the absolute number of related sentences in the text, (b) *centrality* (GRAPH-CENTR) which intuitively measures the extent to which a specific node serves as a bridge to connect any unrelated pieces of information, (c) *clustering* (GRAPH-CLUST) which measure the level of interconnection between the neighbours of a node and itself, and finally, (d) the sum of the centrality and clustering measure (GRAPH-CE&CL). Please refer to (Brandes, 2005) for further details on these graph measures.

Although more sophisticated linguistic processing resources such as POS taggers and dependency parsers are available for at least several languages we did not consider KE techniques that exploit them since the range of languages covered would be still far away from the ca. 70 languages covered by EMM. Furthermore, although the BERT-based approaches to KE (even without any tuning) are known to be orders of magnitudes slower than the other methods, we explored them given the wide range of languages covered in terms of off-the-shelf embeddings.

4 Experiments

4.1 Dataset

For the evaluation of the KE algorithms we created random samples of circa 50 news articles published in 2020 for 7 languages: English, French, German, Italian, Polish, Romanian and Spanish. The selection of the languages was motivated to cover all three main Indo-European language families: Germanic, Romance and Slavic languages.

The news articles were annotated with keywords by two human experts for each language in the following manner. Initially, all annotators were presented with the task definition, keyword selection guidelines, and annotated a couple of trial articles. Next, the annotators were tasked to select keywords for the proper set of 50 news articles for each language. The annotation was done by each annotator separately since we were interested to measure the discrepancies between annotators and differences between the languages. The final sets of documents used for evaluation for some of the languages contained less than 50 news articles due to some near duplicates encountered, etc.

Table 1 shows the differences in terms of keyword annotation distribution across languages. The average number of keywords per article varies from 8.68 for French to 13.20 for German. At the token level, the average ranges from 20.66 annotated tokens (French) per article to 30.24 (Romanian). The discrepancies between annotators differ significantly across languages, e.g., for Polish, only 9.37% of the keywords are shared between the two annotators, whereas for Romanian, they are 48.68%. However, when one measures the differences at the token level the discrepancies are significantly smaller, i.e., for Polish, 49.67% of the tokens are shared between the annotators, whereas for Romanian, 69.16%. This comparison between annotators is completed by computing the percentage of "fuzzy" common tokens (Table 1), corresponding to the common 4-gram characters. As expected, the percentage of "fuzzy" common tokens is higher than for exact common tokens for all languages. It increases by ca. 2 points for English, French, Italian, Spanish and more than 4 points for German, Polish and Romanian.

Based on the relatively high level of discrepancies between each pair of annotators per language (see Table 1) we decided to create the ground truth for evaluation by merging the respective keyword sets for each languages. The statistics of the resulting ground truth data are summarized in Table 2. We can observe that the average number of keywords per article for Italian and French is significantly lower than for the other languages. The average number of tokens per keyword is quite stable, from 2.33 (Spanish) to 2.79 (English), except for German, 1.75 tokens per keyword, due to the frequent use of compounds in this language.

4.2 Evaluation Methodology

We have used the classical *precision* (P), *recall* (R) and F_1 metrics for the evaluation purposes. The overall P, R and F_1 scores were computed as an average over the respective scores for single news articles.

We have computed the scores in three different ways. In the *exact matching* mode, we consider that an extracted keyword is matched correctly only if exactly the same keyword occurs in the ground truth (or vice versa).

In the *partial matching* mode, the match of a given keyword c vis-a-vis Ground Truth $GT = \{k_1, \ldots, k_n\}$ is computed as follows:

$$match(c) = \max_{k \in GT} \frac{2 \cdot commonTokens(c, k)}{|c|_T + |k|_T}$$

where *commonTokens*(*c, k*) denotes the number of tokens that appear both in *c* and *k*, and $|c|_T$ ($|k|_T$) denote the number of tokens the keyword *c* (*k*) consists of. The value of *match*(*c*) is between 0 and 1.

Analogously, in the *fuzzy matching* mode, the match of a given keyword *c* vis-a-vis Ground Truth $GT = \{k_1, \ldots, k_n\}$ is computed as follows:

$$match(c) = \max_{k \in GT} Similarity(c, k)$$

where *Similarity*(*c, k*) is computed using *Longest Common Substring* similarity metric (Bergroth et al., 2000), whose value is between

Language	average number of annotated keywords per article	percentage of common keywords	average number of annotated tokens per article	percentage of exact common tokens	percentage of fuzzy common tokens
English	11.63	17.35%	28.95	53.08%	53.77%
French	8.68	42.97%	20.66	63.95%	65.00%
German	13.20	44.10%	22.37	62.91%	67.56%
Italian	9.81	43.86%	21.87	63.94%	64.74%
Polish	11.01	9.37%	27.77	49.67%	55.28%
Romanian	12.72	48.68%	30.24	69.16%	72.19%
Spanish	12.72	24.13%	26.71	58.06%	59.19%

Table 1: Exact and fuzzy overlap of keywords and tokens for annotator pairs for each language.

Language	#articles	avg. nb of keywords per article	avg. nb of tokens per keyword
English	50	22.04	2.79
French	47	14.34	2.70
German	50	21.36	1.75
Italian	50	16.16	2.34
Polish	39	21.18	2.67
Romanian	49	20.61	2.62
Spanish	48	22.75	2.33

Table 2: Ground Truth statistics.

0 and 1. Both P and R are computed analogously using the concept of partial and fuzzy matching.

The main rationale behind using the partial and fuzzy matching mode was the fact that exact matching is simply too strict in terms of penalisation of automatically extracted keywords which do have strong overlap with keywords in the ground truth.

Finally, we have also computed standard deviation (SD) for all metrics in order to observe whether any of the algorithms is prone to producing response outliers.

4.3 Results

We have evaluated all the algorithms described in Section 3 with the following settings, unless specified elsewhere differently: (a) the max. number of tokens per keyword is 3, whereas the minimum (maximum) number of characters is set to 2 (80), (b) keywords can neither start nor end with a stop word, (c) keywords cannot contain tokens composed only of non-alphanumeric characters, and (d) the default maximum number of keywords to return is 15. The main drive behind setting the maximum number of keywords to 15 is based on empirical observation, optimizing both F_1 score and not returning too long list of keywords.

The overall performance of each algorithm averaged across languages, in term of P , R and F_1 scores is listed in Table 3, respectively for exact, partial and fuzzy matching. In general, only the results for the best settings per algorithm type are provided except for YAKE and KPMINER, which performed overall best. More specifically, the table contains results of some additional variants of YAKE and its combinations with KPMiner, namely: (a) YAKE-15 and YAKE-20 which return 15 and 20 keywords resp., (b) YAKE-KPMINER-I (inter-section) which returns the intersection of the results returned by YAKE-15 and KP-Miner, (c) YAKE- KPMINER-U (union) which merges up to 10 topkeywords returned by YAKE and KP-Miner output, and (d) YAKE-KPMINER-R (re-ranking) which sums the ranks of the keywords returned by YAKE- 15 and KPMINER and selects top 15 keywords after the re-ranking.

Across the three types of matching, the list of algorithms obtaining good results is quite stable (cf. Table 3). YAKE-KPMINER-R constantly obtaining the best F_1, respectively 20.1%, 46.6% and 47.2% for the exact, partial and fuzzy matching, followed or equaled by the YAKE-KPMINER-U.

YAKE-KPMINER-I obtained the best precision, respectively 28.5%, 55.9% and 57.2%. In terms of standard deviation (SD), YAKE-KPMINER-I appears to be the most unstable since it is constantly the algorithm with the highest SD, for P, R and F_1, and for all types of matching.

As expected, the results obtained with partial and fuzzy matching are better than with exact matching. More interestingly, the fuzzy matching also allows to smooth the discrepancy between languages. Figure 1 highlights for YAKE-KPMINER-R algorithm how some languages like Polish, a highly inflected language, have a poor F_1 for exact matching, but are close to the all-language average for fuzzy matching. Figure 2 aims at comparing the results obtained in each language with a selection of algorithms for the fuzzy matching. The KPMINER algorithm appears to be best suited for the French language, whereas German the group of YAKE algorithms appears to be a better choice. There are some other language specific aspects according to the different algorithms, but less significant. As a matter of fact, the observations on YAKE and KPMINER strengths when applying on texts in specific languages were the main drive to introduce the various variants of combining these KE algorithms.

One can also conclude from the evaluation figures that YAKE-KP-MINER-R appears to be the best "all-rounder" algorithm. In this context it is also important to emphasize that the performance of the various algorithms relies on the quality and coverage of the stop word lists, which are used by almost all algorithms compared here. In particular, the respective algorithms used identical stop word lists, covering: English (583 words), French (464), German (604), Italian (397), Polish (355), Romanian (282), and Spanish (352).

KEYEMB-based approaches tend to focus only on the most important sentence in the news article. As such, frequently, several 3-grams candidates originating from the same sentence are returned, where most of them are redundant. Interestingly, as regards fuzzy matching KEYEMB-LASER performs better than BERT-based ones despite not being specially trained on similarity tasks, while KEYEMB-BERT-D performs overall best out of the three. It is worth mentioning that this approach is by far the slowest of the reported approaches in terms of time efficiency.

GRAPH-based approaches suffer from a similar focusing bias: they tend to focus on the most important concepts, as such they are always present but so are some variations thereof, e.g. reporting most frequent words within all the different contexts they appear in, therefore generating redundant keywords. Among this family of algorithms, the GRAPH-DEGREE performed best, meaning that a high co-occurrence count is a good indicator of relevance for KE.

Embedding and graph-based approaches overfocus on the key concepts of a text. The fact that they are based on an indirect form of counting the most important words, without any further post-processing, may in part explain why their performance is comparable to TF-IDF, which relies directly on frequency count. An advantage of graph-based approaches compared to embedding-based ones and TF-IDF is that they don’t need to be trained in advance on any corpora.

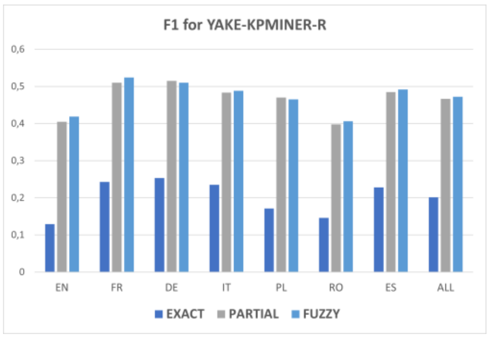


Figure 1: F_1 scores for exact, partial and fuzzy matching for YAKE-KPMINER-R.

4.3.1 Deduplication

Based on the results presented in the previous Section we carried out some additional experiments in order to explore whether the best performing algorithm, namely, YAKE-KPMINER-R, could be improved. In particular, given that this algorithm combines merging of keywords of two different algorithms, we have added an additional deduplication step. To be more precise, all keyword candidates that are properly included in other keyword candidates are discarded. We evaluated this new variant with different settings as regards the maximum allowed number of keywords returned. While we have not observed significant improvements in terms of the F_1 score when increasing the number of keywords returned by the algorithms described in the previous Section, the evaluation of YAKE-KPMINER-R with deduplication revealed that increasing this parameter yields some gains. Figure 3 and 4 provide P , R and F_1 curves for fuzzy match- ing according to the maximum number of keywords allowed to be returned for the English and German subcorpus.

One can observe that shifting the maximum number of keywords to ca. 25 results in some improvement for F_1 and R. While these findings pave the way for some future explorations on parameter tuning to improve F_1 figures, one needs to emphasize here that increasing the number of keywords, even if resulting in some small gains in F_1 is not a desired feature from an application point of view, where analysts expect and prefer to ‘see less than more’.

4.4 Time efficiency performance

We have carried out a small comparison of the run-time behaviour of the algorithms with respect to the time needed to process a collection of 16983

	Exact (%)				Partial (%)				Fuzzy (%)			
Algorithm	P	R	F_1	SD	P	R	F_1	SD	P	R	F_1	SD
TF-IDF	14.2	12.1	12.6	09.2	33.5	30.7	31.2	10.1	34.0	31.1	31.6	10.8
KPMINER	17.8	15.4	15.9	08.6	49.4	38.1	41.9	12.3	51.3	37.2	41.7	14.5
RAKE-DEG	13.1	11.9	12.0	09.0	45.9	33.3	37.1	14.8	47.5	30.6	35.0	18.3
RAKE-DEGFREQ	10.4	09.5	09.5	09.3	37.6	30.5	32.6	13.4	35.2	27.0	29.2	16.1
RAKE-FREQ	14.0	12.6	12.8	09.4	46.8	34.1	38.0	14.3	49.2	31.6	36.3	18.2
KTF	16.8	14.5	14.8	09.6	39.4	30.7	33.5	12.2	43.3	31.5	35.3	13.6
KEYEMB-BERT-D	08.1	08.1	07.6	07.0	38.6	24.5	28.6	16.0	40.3	27.5	31.7	13.9
KEYEMB-BERT-B	04.5	05.0	04.5	05.8	22.6	17.2	18.6	10.7	36.2	27.1	29.9	12.1
KEYEMB-LASER	02.9	03.5	03.0	05.6	18.8	15.1	16.1	11.2	39.4	29.0	32.4	13.2
GRAPH-CENTR	03.2	04.0	03.7	05.9	17.7	12.1	14.3	12.2	29.1	20.0	22.9	12.6
GRAPH-CLUST	04.1	03.8	03.8	**05.4**	17.2	13.0	14.2	**09.2**	29.0	24.6	25.9	10.7
GRAPH-CE&CL	03.9	03.9	03.8	05.5	19.1	13.4	14.7	10.4	31.4	24.8	26.7	12.2
GRAPH-DEGREE	04.2	04.4	04.1	05.9	21.2	15.3	16.8	11.2	34.5	27.3	29.4	11.6
YAKE-15	22.0	17.8	19.1	10.3	45.9	42.3	43.1	10.6	46.6	42.9	43.5	11.3
YAKE-20	19.2	20.3	19.2	09.1	41.9	**47.2**	43.6	09.5	42.1	**48.3**	44.0	**10.0**
YAKE-KPMINER-I	**28.5**	08.8	12.6	18.3	**55.9**	24.3	32.2	26.4	**57.2**	23.5	31.3	28.7
YAKE-KPMINER-R	19.9	**21.9**	**20.1**	08.5	48.2	47.1	**46.6**	09.4	49.8	47.2	**47.2**	10.6
YAKE-KPMINER-U	19.4	21.7	19.8	08.6	48.7	46.6	**46.6**	09.6	50.4	46.4	47.0	11.0

Table 3: Overall performance overview: exact, partial and fuzzy matching.

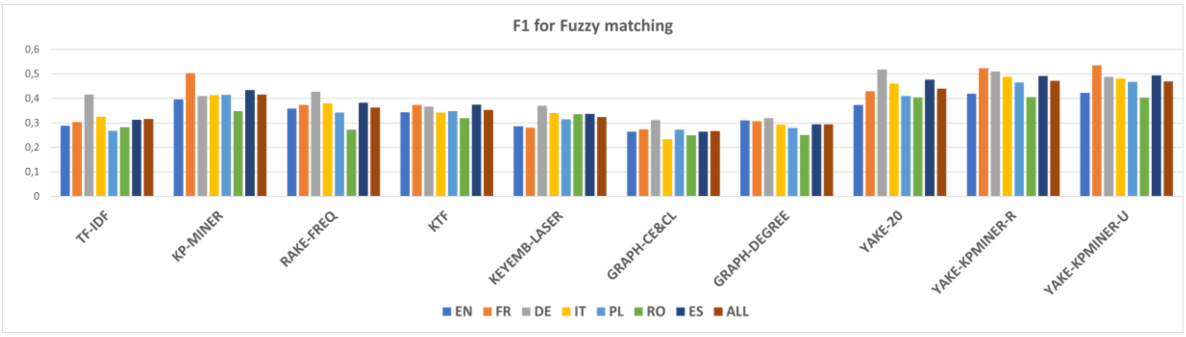
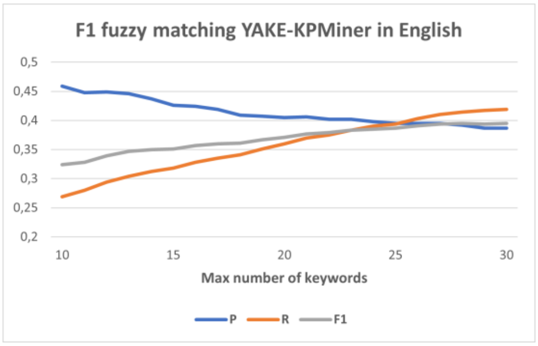


Figure 2: F_1 fuzzy matching figures for a selection of algorithms and all languages.



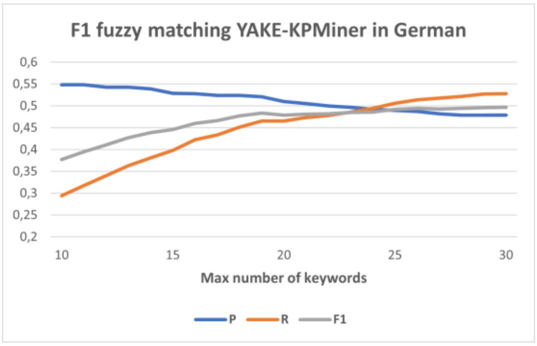


Figure 3: P, R and F_1 curves for fuzzy matching for the varying number of maximum number of keywords returned for the English subcorpus.

Figure 4: P, R and F_1 curves based on varying num- ber of maximum number of keywords returned for the German news subcorpus.

Algorithm	time (seconds)
KTF	12.13
RAKE-DEG	9.36
RAKE-FREQ	9.33
RAKE-DEGFREQ	9.37
KPMINER	21.12
YAKE-15	21.04
YAKE-KPMINER-R	42.56

Table 4: Time efficiency comparison on a set of circa 17K news articles in English on Covid-19.

news articles on Covid-19 in English (84.9 MB of space on disk). The time given in seconds to run KTF, Rake, KPMiner, Yake and some variants thereof are provided in Table 4. All the aforementioned algorithms have been implemented in Java and optimized in term of efficient data structures used that correspond to the upper bounds of the respective time complexity of these algorithms. Both embedding- and graph-based algorithms explored in our study were implemented in Python, using some existing libraries, and were not optimized for speed. For these reasons, it is not meaningful to report their exact time performance. As before, on a given CPU, embedding-based approaches run an order of magnitude slower than graph based algorithms, which themselves run a magnitude slower than the simpler algorithms, whose performance is reported in Table 4.

5 Conclusions and Outlook

This paper presented the results of a small comparative study of the performance of some state-of-the-art knowledge-lightweight keyword extraction methods in the context of indexing news articles in various languages with keywords. The best performing method, namely, a combination of Yake and KPMiner algorithms, obtained F_1 score of 20.1%, 46.6% and 47.2% for the exact, partial and fuzzy matching respectively. Since both of these algorithms exploit neither any language-specific (except stop word lists) nor other external resources like domain-specific corpora, this solution can be easily adapted to the processing of many languages and constitutes a strong baseline for further explorations.

The comparison presented in this paper is not exhaustive, other linguistically-lightweight unsupervised approaches could be explored, e.g., the graph-centric approach presented in (Skrlj et al., 2019), and some post-processing filters to merge redundant keywords going beyond exploiting string similarity metrics, and simultaneously, techniques to improve diversification of the keywords returned.

Extending the approaches explored in this study, e.g., through use of part-of-speech-based patterns to filter out implausible keywords (e.g., imposing constraints to include only adjectives and nouns as elements of keywords), use of more elaborated graph-based keyword ranking methods (e.g. Page Rank), integration of semantics (e.g., linking semantic meaning to text sequences through using knowledge bases and semantic networks (Papagiannopoulou and Tsoumakas, 2020; Hasan and Ng, 2014; Kilic and Cetin, 2019; Alami Merrouni et al., 2019)) would potentially allow to improve the performance. However, these extensions would require significantly more linguistic sophistication, and consequently would be more difficult to port across languages.

For matters related to accessing the ground truth dataset created for the sake of carrying out the evaluation presented in this paper please contact the authors.

Acknowledgments

We are greatly indebted to Stefano Bucci, Florentina Ciltu, Corrado Mirra, Monica De Paola, Teófilio Garcia, Camelia Ignat, Jens Linge, Manuel Marker, Małgorzata Piskorska, Camille Schaeffer, Jessica Scornavacche and Beatriz Torighelli for helping us with the keyword annotation of news articles in various languages. We are also thankful to Martin Atkinson who contributed to the work presented in this report, and to Charles MacMillan for proofreading the paper.

References

Zakariae Alami Merrouni, Bouchra Frikh, and Brahim Ouhbi. 2019. Automatic keyphrase extraction: a survey and trends. *Journal of Intelligent Information Systems*, 54.

Mikel Artetxe and Holger Schwenk. 2019. Mas-sively multilingual sentence embeddings for zero- shot cross-lingual transfer and beyond. *Transac- tions of the Association for Computational Linguis- tics*, 7:597–610.

Lasse Bergroth, H. Hakonen, and T. Raita. 2000. A survey of longest common subsequence algorithms. pages 39–48.

Adrien Bougouin, Florian Boudin, and Béatrice Daille. 2013. TopicRank: Graph-based topic ranking for keyphrase extraction. In *Proceedings of the* 6^{th} *International Joint Conference on NLP*, pages 543–551, Nagoya, Japan.

Ulrik Brandes. 2005. *Network analysis: methodological foundations*, volume 3418. Springer Science & Business Media.

Ricardo Campos, Vítor Mangaravite, Arian Pasquali, A. Jorge, C. Nunes, and Adam Jatowt. 2020. Yake! keyword extraction from single documents using multiple local features. *Inf. Sci.*, 509:257–289.

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

Maarten Grootendorst. 2020. Keybert: Minimal keyword extraction with bert.

Kazi Saidul Hasan and Vincent Ng. 2014. Automatic keyphrase extraction: A survey of the state of the art. In *Proceedings of the* 52^{nd} *ACL Conference*, pages 1262–1273, Baltimore, Maryland. ACL.

Ozlem Kilic and Aydın Cetin. 2019. A survey on keyword and key phrase extraction with deep learning. pages 1–6.

Marina Litvak and Mark Last. 2008. Graph-based keyword extraction for single-document summarization. In *Coling 2008: Proceedings of the work- shop Multi-source Multilingual Information Extrac- tion and Summarization*, pages 17–24.

Luís Marujo, Anatole Gershman, G. Jaime Carbonell, E. Robert Frederking, and Paulo João Neto. 2012. Supervised topical key phrase extraction of news stories using crowdsourcing, light filtering and coreference normalization. *Language Resources and Evaluation*, pages 399–403.

Luís Marujo, Márcio Viveiros, and João Neto. 2013. Keyphrase cloud generation of broadcast news. *Proceedings of INTERSPEECH 2013*.

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119. Curran Associates, Inc.

Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. 2018. Unsupervised learning of sentence embeddings using compositional n-gram features. In *Proceedings of NAACL 2018*, pages 528–540, New Orleans, Louisiana. ACL.

Eirini Papagiannopoulou and Grigorios Tsoumakas. 2020. A review of keyphrase extraction. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 10.

Jakub Piskorski, Karol Wieloch, and Marcin Sydow. 2009. On knowledge-poor methods for person name matching and lemmatization for highly inflectional languages. *Information Retrieval*, 12(3):275–299.

Nils Reimers and Iryna Gurevych. 2020. Making monolingual sentence embeddings multilingual using knowledge distillation. In *Proceedings of the 2020 Conference on Empirical Methods in Natu- ral Language Processing*. Association for Computational Linguistics.

Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. 2010. Automatic keyword extraction from individual documents. In Michael W. Berry and Jacob Kogan, editors, *Text Mining. Applications and Theory*, pages 1–20. John Wiley and Sons, Ltd.

Blaz Skrlj, Andraz Repar, and Senja Pollak. 2019. Rakun: Rank-based keyword extraction via unsupervised learning and meta vertex aggregation. In *SLSP*.

Ralf Steinberger, Martin Atkinson, Teofilo Garcia, Erik van der Goot, Jens Linge, Charles Macmillan, Hristo Tanev, Marco Verile, and Gerhard Wagner. 2017. EMM: Supporting the analyst by turning multilingual text into structured data. In *Transparenz Aus Verantwortung: Neue Herausforderungen Für Die Digitale Datenanalyse*. Erich Schmidt Verlag.