

Local Word Vectors Guiding Keyphrase Extraction

Eirini Papagiannopoulou, Grigorios Tsoumakas

School of Informatics, Aristotle University of Thessaloniki, 54124, Greece

Abstract

Automated keyphrase extraction is a fundamental textual information processing task concerned with the selection of representative phrases from a document that summarize its content. This work presents a novel unsupervised method for keyphrase extraction, whose main innovation is the use of *local* word embeddings (in particular GloVe vectors), i.e., embeddings trained from the single document under consideration. We argue that such local representation of words and keyphrases are able to accurately capture their semantics in the context of the document they are part of, and therefore can help in improving keyphrase extraction quality. Empirical results offer evidence that indeed local representations lead to better keyphrase extraction results compared to both embeddings trained on very large third corpora or larger corpora consisting of several documents of the same scientific field and to other state-of-the-art unsupervised keyphrase extraction methods.

Keywords: keyphrase extraction, unsupervised method, GloVe, local word vectors, Reference Vector Algorithm

2010 MSC: 68T50

1. Introduction

Keyphrase extraction is concerned with the selection of a set of phrases from within a document that together summarize the main topics discussed in that

Email addresses: epapagia@csd.auth.gr (Eirini Papagiannopoulou), greg@csd.auth.gr (Grigorios Tsoumakas)

document (Hasan & Ng, 2014). Automatic keyphrase extraction is a fundamental task in digital content management as it can be used for document indexing, which in turns enables calculating semantic similarity between documents (and hence document clustering), and can improve browsing of digital libraries (Gutwin et al., 1999; Witten, 2003). In addition, automatic keyphrase extraction offers an approach to document summarization. Keyphrase extraction is particularly important in academic publishing, where it is used as a technological building block to recommend articles to readers, to highlight missing citations to authors and to analyze research trends (Augenstein et al., 2017).

Supervised machine learning approaches for automatic keyphrase extraction rely on annotated corpora. However, manual selection of the keyphrases of each document by humans requires the investment of time and money and is characterized by great subjectivity. In many cases, the extracted keyphrases cover one or more non-core topics due to misunderstandings, or they miss one or more of the important topics discussed in the document. Using multiple annotators can partially address the problem of subjectivity by collecting more keyphrases (Chuang et al., 2012; Sterckx et al., 2016). This, however, comes at the expense of additional annotation effort. In addition, supervised methods often fail to generalize well to documents coming from a different content domain than the training corpus, may require retraining to address concept drift, and are more susceptible to varying vocabularies across documents and different personal writing styles across authors.

In contrast, this work takes a novel unsupervised path to keyphrase extraction. To be able to take into account the semantic similarity among words we consider word embeddings, in particular the one generated by GloVe (Pennington et al., 2014). Different however from past approaches that exploit word embeddings in keyphrase extraction Wang et al. (2014), we do not use pretrained vectors, but instead learn *local* GloVe representations in the context of *single documents*, in particular full-texts of academic publications. Our main hypothesis is that such local representations will be able to more accurately capture the semantic similarity of the different words and phrases in the context of each

document, and help us extract more representative keyphrases, compared to global representations and other state-of-the-art unsupervised keyphrase extraction methods. Our research objective is to investigate whether this hypothesis holds.

Our approach extracts keyphrases from the title and abstract of an academic publication, which constitute a clear and concise summary of the whole publication, in order to avoid the noise and redundancy found in the full-text. Once local word vectors have been learned from the full-text of a given academic publication, we compute the mean vector of the words in its title and abstract, dubbed *reference vector*, which we can intuitively consider as a vector representation of the semantics of the whole publication. We then extract candidate keyphrases from the title and abstract, and rank them in terms of their cosine similarity with the reference vector, assuming that the closer to the reference vector is a word vector, the more representative is the corresponding word for the publication.

The rest of the paper is organized as follows. Section 2 gives a review of the related work in the field of keyphrase extraction as well as a brief overview of methods that produce word embeddings. Section 3 presents the proposed approach. Section 4 describes empirical results highlighting different aspects of our approach and comparing it with other state-of-the-art unsupervised keyphrase extraction methods. Finally, Section 5 presents the conclusions of this work and points to future work directions.

2. Related Work

2.1. Automatic Keyphrase Extraction

Automatic keyphrase extraction is a well-studied task and a variety of techniques have been proposed in the past. In this section, we present both supervised and unsupervised methods in a comprehensive and structured way.

2.1.1. Unsupervised Approaches

Unsupervised keyphrase extraction approaches typically follow a standard three-stage process (Hasan & Ng, 2010, 2014). The first stage concerns choosing the candidate lexical units with respect to some heuristics, such as the exclusion of stop words or the selection of words that are nouns or adjectives. The second stage concerns ranking these lexical units by measuring their *importance* through co-occurrence statistics or syntactic rules. The final stage concerns keyphrase formation, where the top-ranked lexical units are used either as keywords or as components of keyphrases.

The baseline approach for unsupervised keyphrase extraction is *TfIdf* (Jones, 1972). It ranks phrases in a particular document according to their frequency in this document (tf), multiplied by the inverse of their frequency in all documents of a collection (idf). Recently, Florescu & Caragea (2017a) proposed an approach for combining TfIdf with any other word-scoring approach. In their approach, a phrase’s score is computed by multiplying its frequency within the document (tf) with the mean of the scores of the phrase’s words.

Graph-based ranking algorithms are based on the following idea: first, a graph from a document is created that has as nodes the candidate keyphrases, and then edges are added between *related* candidate keyphrases. The final goal is the ranking of the nodes using a graph-based ranking method, such as PageRank (Brin & Page, 1998), Positional Function (Herings et al., 2005), and HITS (Kleinberg, 1999). *TextRank* (Mihalcea & Tarau, 2004) builds an undirected and unweighted graph with candidate lexical units as nodes for a specific text and adds connections (edges) between those nodes that co-occur within a window of N words. The ranking algorithm runs iteratively until it converges. Once the algorithm converges, nodes are sorted by decreasing order and the top T nodes form the final keyphrases. Variations of TextRank include *SingleRank* (Wan & Xiao, 2008), where edges have a weight equal to the number of co-occurrences of their corresponding nodes within a window, and *ExpandRank* (Wan & Xiao, 2008), where the graph includes as nodes not only the lexical

units of a specific document but also the lexical units of the k nearest neighboring documents of the initial document. In ExpandRank, an edge between two nodes exists if the corresponding words co-occur within a window of W words in the whole document set. Once the graph is constructed, ExpandRank’s procedure is identical to SingleRank. Recently, another unsupervised graph-based model, called PositionRank, was proposed by Florescu & Caragea (2017b). This method tries to capture frequent phrases taking into account, at the same time, their corresponding position in the text. More specifically, it incorporates all word’s positions into a biased PageRank. Finally, the keyphrases are scored and ranked. Wang et al. (2014) propose a graph-based ranking model that takes into consideration information coming from distributed word representations. In particular, again a graph of words is initially created with edges that represent the co-existence between the words within a window of W consecutive words. Then, a weight (the *word attraction score*) is assigned to every edge, which is the product of two individual scores: a) the *attraction force* between two words which uses the frequencies of the words as well as the distance between the corresponding word embeddings, and b) the *dice coefficient* (Dice, 1945; Stubbs, 2003). Once more, a weighted PageRank algorithm is utilized to rank the words. A similar approach that uses a personalized weighted PageRank model with pretrained word embeddings, but with different edge weights is proposed in Wang et al. (2015).

RAKE (Rose et al., 2010) is a domain-independent and language-independent method for extracting keyphrases from individual documents. Given a list of stop words, a set of phrase delimiters, and a set of word delimiters, RAKE cuts the document text up to candidate sequences of content words and then builds a graph of word co-occurrences. Afterwards, word scores are calculated for each candidate keyword. The basic difference in comparison with the previous approaches is that RAKE is able to identify keyphrases that contain *interior* stop words. Specifically, RAKE detects pairs of keywords that adjoin one another at least twice in the same document, in the same order, and creates a new candidate keyphrase that contains the corresponding interior stop words.

There exists a group of approaches that incorporate knowledge from citation networks. A typical method of this group is CiteTextRank (Gollapalli & Caragea, 2014), which constructs a weighted graph considering the information of short text descriptions surrounding mentions of papers (citation contexts).

Topic-based clustering methods aim at extracting keyphrases that cover all the major topics of a document. A known technique of this family is KeyCluster (Liu et al., 2009), which clusters similar candidate keywords utilizing Wikipedia and co-occurrence statistics. The basic idea is that each cluster corresponds to a specific topic of the document and by selecting candidate keyphrases from each cluster, all the topics are covered. TopicRank (Bougouin et al., 2013) is another method that extracts keyphrases from the most significant topics of a document. First, the text of interest is preprocessed and then keyphrase candidates are grouped into separate topics using hierarchical agglomerative clustering. In the next stage, a graph of topics is constructed whose edges are weighted based on a measure that considers phrases' offset positions in the text. As a final step, TextRank is used to rank the topics. Topical PageRank (TPR) (Liu et al., 2010) is an alternative methodology which first obtains the topics of words and documents using Latent Dirichlet Allocation (LDA) (Blei et al., 2003) and then begins the construction of the word graph for a given document. The idea of TPR is to run a PageRank for each topic separately by modifying the basic PageRank score function utilizing the word topic distributions calculated earlier for the given document.

Tomokiyo & Hurst (2003) create both unigram and n-gram language models on a foreground corpus (target document) and a background corpus (document set). Their main idea is based on the fact that the loss between two language models can be measured using the Kullback-Leibler divergence. Particularly, in a phrase level, for each phrase, they compute the *phraseness* as the divergence between the unigram and n-gram language models on the foreground corpus and the *informativeness* as the divergence between the n-gram language models on the foreground and the background corpus. Finally, they sum the phraseness and informativeness to obtain a final score for each phrase and sort them by

this score.

2.1.2. Supervised Approaches

In supervised learning, a classifier is trained on annotated with keyphrases documents in order to determine whether a candidate phrase is a keyphrase or not. These keyphrases and non-keyphrases are used to generate positive and negative examples.

The famous KEA system (Witten et al., 1999) is one of the first supervised keyphrase extraction systems which uses only two features during training and extraction process: TfIdf and first occurrence attribute. The training stage uses documents whose the keyphrases are known. Then, for each document, candidate keyphrases are identified and their feature values are calculated. Finally, KEA uses an expression to rank the candidates, that incorporates the corresponding features, based on Naive Bayes. Later, another system which uses linguistic knowledge has been proposed by Hulth (2003). For each candidate phrase of the training data, that has been selected in an earlier stage, four features are calculated: the within-document frequency, the collection frequency, the relative position of the first occurrence, and POS tag(s). Finally, the machine learning approach is a rule induction system with bagging. The popular keyphrase extraction system, called Maui (Medelyan et al., 2009), first determines all n-grams up to 3 words and then calculates a set of meaningful features such as TfIdf, the position of the first occurrence, *keyphraseness*, phrase length, and features based on Wikipedia statistics which are used in its classification model. In Caragea et al. (2014), a binary classification model, CeKE, has been proposed (Naive Bayes classifier with decision threshold 0.9) which utilizes *novel* features from the information of citation contexts and existing features from previous works. Recently, Sterckx et al. (2016) conduct an interesting study where they conclude that unlabeled keyphrase candidates are not reliable as negative examples. For this reason, they propose to treat supervised keyphrase extraction as Positive Unlabeled Learning by assigning weights to training examples, modeling in this way the uncertainty. Firstly, they train a classifier on a single

annotator’s data and use the predictions on the negative/unlabeled phrases as weights. Then, another classifier is trained on the weighted data mentioned above in order to predict the labels of the candidates.

Other approaches that use neural network models have been proposed, with the most recent work to be a generative model for keyphrase prediction using an encoder-decoder framework that tries to capture the semantic meaning of the content via a deep learning method (Meng et al., 2017). In fact, it applies a recurrent neural network (RNN) Encoder-Decoder model in order to learn the mapping from the source text to its corresponding target keyphrases. The main drawback with such approaches is that the model is expected to work well on text documents that have the same domain with the training data.

Another point of view is to see keyphrase extraction as a learning to rank task such as in Jiang et al. (2009). The basic reason to adopt this approach is the fact that it is easier to determine if a candidate phrase is a keyphrase in comparison with another candidate phrase than to classify it as a keyphrase or not, by taking such hard decisions.

We should not omit the recent work on keyphrase extraction where the task has been treated as a sequence tagging task using Conditional Random Fields (Gollapalli et al., 2017). The features used represent linguistic, orthographic, and structure information from the document. Furthermore, they investigate feature-labeling and posterior regularization in order to integrate expert/domain-knowledge throughout the keyphrase extraction process.

2.2. Dense Vectors

Since 1990, a great number of methods have been proposed for words’ representation, such as the popular Latent Dirichlet Allocation (LDA) (Blei et al., 2003) and Latent Semantic Analysis (LSA) (Deerwester et al., 1989, 1990). Generally, such approaches that are based on co-occurrences’ matrix (Deerwester et al., 1990; Lund & Burgess, 1996; Blei et al., 2003) are able to capture semantics and are also used for further dimensionality reduction. However, Bengio et al. (2003) invented the term “word embeddings”, proposing a simple feed-

forward neural network which predicts the next word in a sequence of words. In fact, word embeddings came to the foreground by Mikolov et al. (2013), who presented the well-known Continuous Bag-of-Words Model (CBOW) and the Continuous Skip-gram Model, establishing widely the use of pretrained embeddings.

In this work, we utilize the GloVe (Global Vectors) (Pennington et al., 2014) method for the generation of the word vectors. This methodology exploits statistical information by training only on the non-zero elements in a word-word co-occurrence matrix in an efficient way and finally, creates a meaningful word vector space.

3. The Reference Vector Algorithm

This section describes thoroughly our approach, called Reference Vector Algorithm (RVA), for extracting keyphrases from the titles and abstracts of scientific articles. Our approach exploits the GloVe word vector representation to detect the candidate keywords and to provide a complete set of representative keyphrases for a particular title and abstract. Fig. 1 summarizes the processing pipeline of RVA.

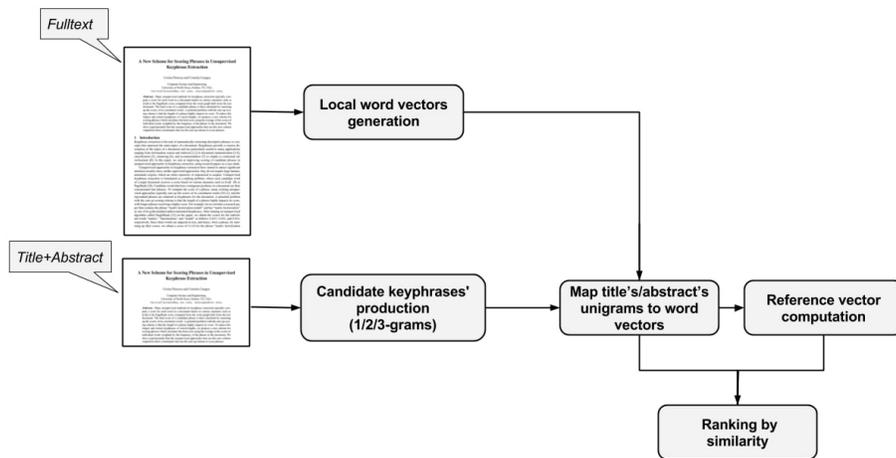


Figure 1: System processing pipeline.

3.1. Candidate Keyphrases' Production

We follow the choice of previous keyphrase extraction systems (Hulth, 2003; Medelyan et al., 2009) by extracting only unigrams, bigrams, and trigrams, as these are the most frequent lengths of keyphrases that are met in the datasets used in the experimental study (see Section 4.1 for more details about the datasets' statistics). In this way, we can effectively reduce the number of possible n -grams that are candidates as keyphrases by restricting the value of n to $\{1, 2, 3\}$, with respect to the observation that, in general, a document's keyphrases tend to be up to trigrams (Gollapalli et al., 2017).

Candidate Unigrams: Unigrams constitute the smallest but the most significant parts that form the longer keyphrases. The criteria for the selection of the appropriate unigrams are the following:

- candidates should have word length lower than 36 and greater than 2 characters (a quite wide range, as the longest word in the well-known Oxford English Dictionary contains 30 characters),
- they do not belong to the stop words list defined by us,
- they are not numbers,
- they do not include the following set of characters: !, @, #, \$, *, =, +, ,, ,, ?, >, <, &, (,), {, }, [,], |

Candidate Bigrams: We choose as candidate bigrams those whose words are in candidate unigrams and appear in the text in that specific sequence. We do not keep as candidate bigrams those whose the length of both words is lower than 4.

Candidate Trigrams: We apply the same procedure as above (for bigrams).

3.2. Scoring the Candidate Keyphrases

As a first step, we produce the local word vectors by applying the GloVe model to the target full-text scientific publication (that's why we call them *local* vectors). The word vectors generated by GloVe, in the context of only one

article, encode the role of words as expressive means of writing, via a vector representation. This type of representation can capture how a limited vocabulary is structured and extended within the narrow limits of a scientific paper. We choose the GloVe technique, instead of other word vector representations, as it is based on the full-text co-occurrence statistics within a predetermined window. GloVe builds an overview of the neighborhood of each one word and, simultaneously, provides us with a picture of its *local contexts*. For the purposes discussed above, we utilize the implementation of the GloVe model for learning word vector representations that is publicly available by Stanford University on GitHub ¹.

We compute the mean vector of the title and the abstract, called *reference vector* by averaging the individual local word vectors that appear in that text segment. First, we sum all the word vectors which match to the candidate unigrams formed in the previous step. Then, the reference vector is derived by dividing with the number of candidate unigrams contained in the title and the abstract. We should take into account that the more often a word shows up in the target-text, the more it affects the reference vector. Finally, we calculate the cosine similarity between each candidate unigram’s local vector that appears in the text segment and the reference vector, creating a *mapping* between the words and their corresponding cosine similarity scores.

As a scoring function for a candidate bigram or trigram, we choose the sum of the individual words’ scores, as we prefer the informativeness come from the longer keyphrases rather than the shorter ones e.g. the unigrams. A great number of existing unsupervised approaches sum up the individual word scores to produce the final phrase score (Mihalcea & Tarau, 2004; Wan & Xiao, 2008). In this way, we expand the *mapping* mentioned above with the bigrams/trigrams and their corresponding score.

Note that most approaches include the Part-of-Speech (PoS) tagging stage based on the observation that the lexical units which belong to a keyphrase

¹<https://github.com/stanfordnlp/GloVe>

often are nouns, adjectives or adverbs (see Sections 2.1.1, 2.1.2). In addition to PoS tagging, stemming is another basic preprocessing step that is suggested by some approaches such as in Hulth (2003). Our decision, to use the word vectors' representation mentioned above, provides us with the advantage to avoid such additional and time-consuming processes, as GloVe is designed to capture in a quantitative way the nuance necessary to discriminate two individual words by associating more than a single number to them, utilizing the vector difference between the two corresponding word vectors.

4. Experiments

We first present the two collections that were used in our empirical study along with some interesting statistics. Then, we describe the evaluation framework and the experimental setup. Finally, we discuss in detail the results, providing both a quantitative and a qualitative evaluation of the proposed approach.

4.1. Data Sets and their Statistics

Our empirical study is based on 2 popular collections of scientific publications: a) Krapivin (Krapivin et al., 2008), which contains 2304 scientific full-text articles from computer science domain published by ACM, along with author-assigned and editor-corrected keyphrases, and b) Semeval (Kim et al., 2010), which contains 244 scientific full-text articles from the ACM Digital Library, along with author-assigned, as well as reader-assigned keyphrases. We apply a preprocessing stage on both datasets in order to separate the upper part (title, abstract for Krapivin and title, abstract, Categories/Subject Descriptors as well as General Terms of the ACM's Computing Classification System for Semeval) of each document from the remaining part (main text body). The refinement process of the Krapivin dataset was quite simple as the title and the abstract are clearly indicated. However, the corresponding separation process for the Semeval dataset was based on heuristic rules (the main body usually starts with

a section that contains in its title derivatives of the word “introduction” and it is located below the Categories/Subject Descriptors and the General Terms).

Figure 2 presents box and whisker plots of the percentage of the *gold*, i.e., ground truth keyphrases appearing in the abstract, and in the full-text of each scientific publication of both collections. We notice that full-texts include the great majority of the gold keyphrases with a mean value approximately equal to 90%. Abstracts, on the other hand, include approximately half of the gold keyphrases on average.

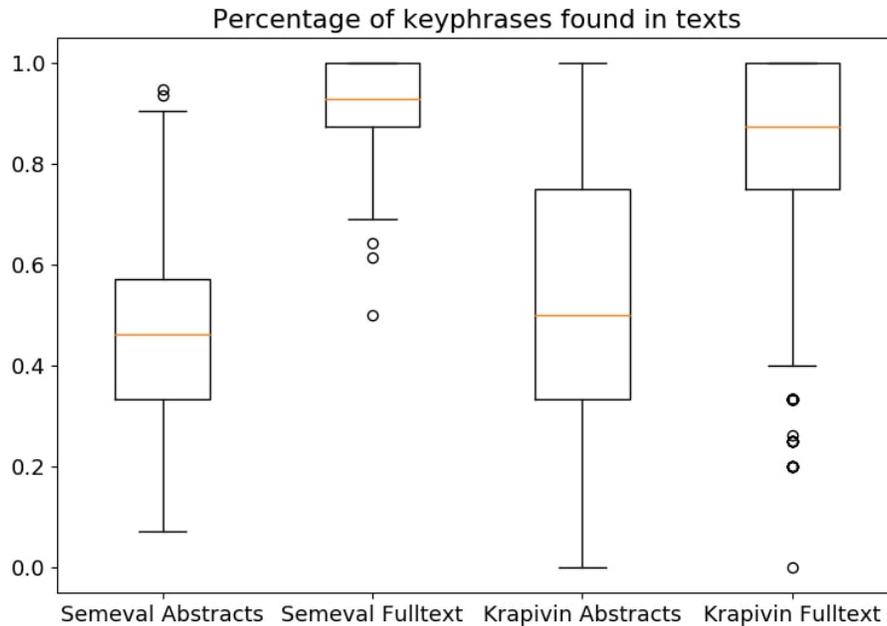


Figure 2: Box plots of the percentage of keyphrases found in the abstract and full-text of each of the two collections. The two box plots on the left correspond to the *Semeval* collection, while the two on the right correspond to the *Krapivin* collection.

Figure 3 presents box and whisker plots of the number of gold keyphrases that are associated with each article of the two collections. We can see that in *Semeval* an average number of 14 keyphrases is assigned per document, whereas the main range of values is from 13 to 17 keyphrases. In the case of *Krapivin*, the main range of values varies from 4 to 6, with a mean value approximately

equal to 5.

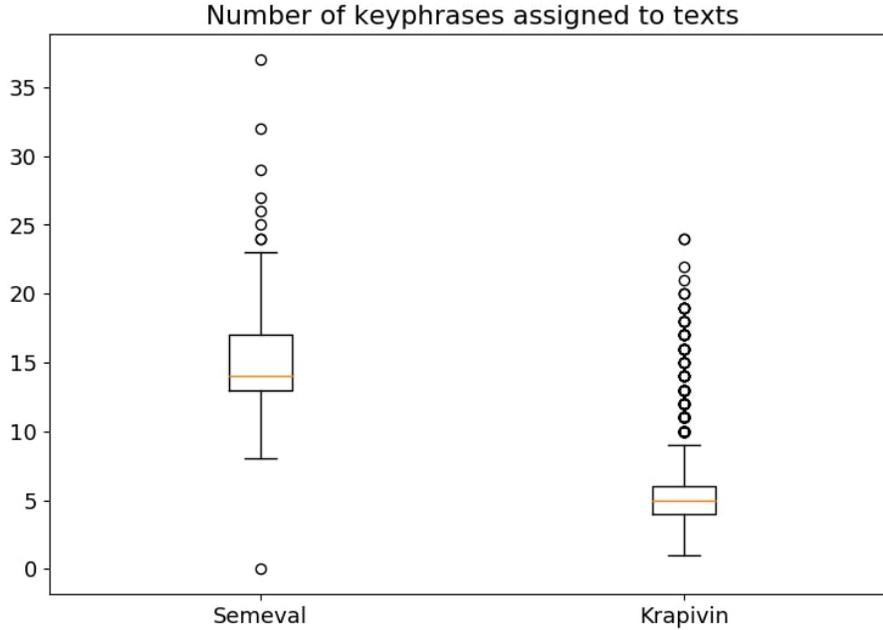


Figure 3: Box plots of the number of keyphrases for Semeval (left) and Krapivin (right).

Figure 4 presents box and whisker plots of the percentage of “gold” keyphrases per article that include at least one stop word. The keyphrases of *Semeval* dataset have quite high percentages of phrases with stop words in comparison with the *Krapivin* dataset.

Finally, Table 1 presents the frequency of the gold keyphrases in each collection per different length (number of words). We can see that most keyphrases are bigrams, followed by unigrams/trigrams. Keyphrases with 4 to 6 words are less frequent, while there also exist a couple of outliers with 7 to 9 words.

Data sets	1	2	3	4	5	6	7	8	9
Semeval	759	2005	782	171	46	16	3	2	1
Krapivin	2330	7575	1936	364	70	18	2	1	0

Table 1: Frequency of gold keyphrases per length (number of words), ranging from 1 to 9.

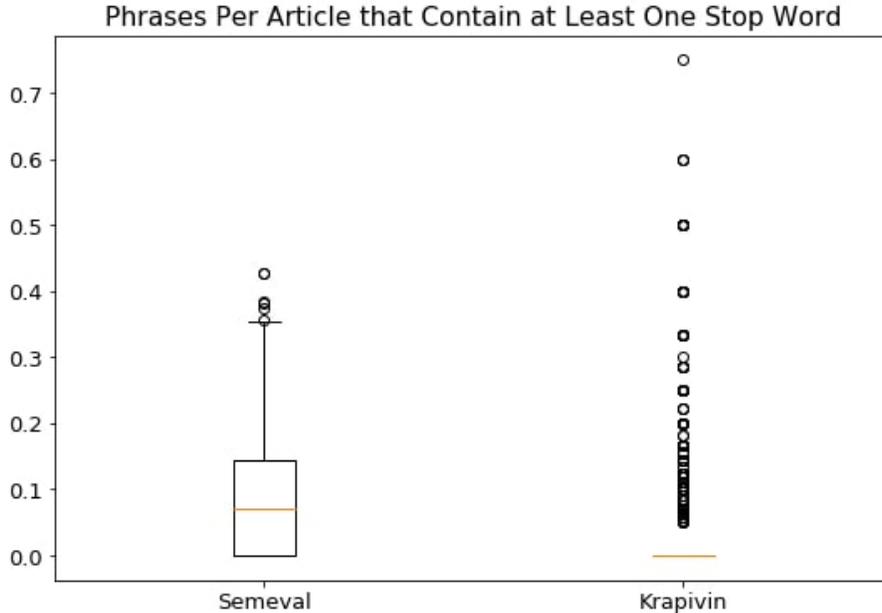


Figure 4: Box plots of the percentage of “gold” keyphrases per article that include at least one stop word for Semeval (left) and Krapivin (right).

4.2. Experimental Setup

It is not clear how to properly evaluate a returned n-gram phrase given a golden multiword keyphrase, especially in cases where the returned n-gram is part of or longer than the golden keyphrase. For this reason, we follow the evaluation process of Rousseau & Vazirgiannis (2015), which calculates the F_1 -measure between the set of words found in all golden keyphrases and the set of words found in all extracted keyphrases.

Regarding the GloVe setup, we used the default parameters ($x_{max} = 100$, $\alpha = \frac{3}{4}$, $window\ size = 10$), as they are set in the experiments of Pennington et al. (2014) who used empirically parameter tuning to find the best values. Finally, we produce 50 and 200-dimensional vectors with 50 iterations as indicated in Pennington et al. (2014) for vectors smaller than 300 dimensions. Such dimensions are appropriate as there are also pretrained GloVe word embeddings for 50 and 200-dimensional vectors, useful for the comparison between local and

pretrained word vectors.

We propose the number of keyphrases to be determined based on the title’s and abstract’s size. Particularly, we choose a more flexible threshold for the number of the representative phrases that will be returned as keyphrases which is inspired by Mihalcea & Tarau (2004), i.e., the selection of the top-scored N phrases as keyphrases, where N is equal to $\frac{1}{3}$ of the number of the different words in the title and abstract, rather than to set a fixed number, which would be a quite strict decision considering Fig. 3.

We utilize the PKE, which is an open source python-based keyphrase extraction toolkit (Boudin, 2016), for the experiments with TfIdf and the graph-based approaches. The code for the RVA method will be uploaded to our Github repository ². The datasets with the abstracts of Krapivin and Semeval are available for research purposes ³

4.3. RVA Variants Evaluation Based on Text Size for GloVe Training

In this section, we give a general view of the performance of the RVA algorithm by changing the dimension of the word vectors and training the GloVe model on different corpus sizes, including the use of word vectors trained on massive web datasets (pretrained word vectors). Specifically, we ran experiments with the pretrained word vectors that were created by training on Wikipedia 2014 + Gigaword 5 which have 400000 vocabulary size, uncased. Furthermore, according to the RVA’s methodology, we generated local word vectors from each one scientific publication of the 2 collections, keeping them in separate files. Finally, we trained a GloVe model on the smaller dataset collections of Semeval and Krapivin, separately, as an intermediate size of the corpus. In all cases, except for those of the pretrained word vectors, we have included in the vocabulary all the possible words that appear in the texts.

As this is the first time that such local word vectors are utilized in this way,

²<https://github.com/epapagia/RVA>

³<https://github.com/epapagia/Datasets-Keyphrase-Extraction>

we prepared an experimental study appropriate to provide us with comprehensive conclusions about the functionality of the reference vectors as guides of the keyphrase extraction process. For this reason, we have designed 2 additional different versions of the proposed RVA approach called *Full-text Reference Vector Algorithm (RVA-F-F)* and *Reference Vector Algorithm using Full-text Candidates (RVA-A-F)*, respectively. The RVA-F-F follows exactly the same process as the RVA. The main difference is that the reference vector, i.e., the mean vector, is computed by averaging the individual local word vectors that appear in the full-text, not only in the title and the abstract. The candidate words are, also, extracted from the full-text article. On the other hand, the main difference between RVA and RVA-A-F is that the latter uses candidate unigrams, bigrams and trigrams from the whole article without being limited to the article’s summary like RVA. However, the reference vector is still calculated based only on the title and the abstract. Conventionally, the first letter after the first dash indicates the part of the text from where the reference vector is calculated (A for abstract and title and F for full-text), whereas the second letter refers to the part of the text from which the candidate keywords come. For consistency reasons, the proposed method is denoted as RVA-A-A. Table 2 describes in detail the different word vector settings with respect to the vector dimensions and the text size used for training of GloVe, providing the corresponding abbreviations that are used in the results’ Table 3.

Abbreviation	Description
LOC-50	local 50-dim vectors - trained on individual files
LOC-200	local 200-dim vectors - trained on individual files
CV-50	50-dim vectors - trained on each collection separately
CV-200	200-dim vectors - trained on each collection separately
PV-50	50-dim pretrained vectors
PV-200	200-dim pretrained vectors

Table 2: Explanation of the abbreviations used in the tables with the experimental results to describe the settings of the proposed method and its variants.

Experiments conducted for word vectors with dimensions 50 and 200 for all RVA variants (RVA-A-A, RVA-F-F, RVA-A-F) and the corresponding results show that the vectors’ dimensions do not substantially affect the methods’ performance. Regarding the size of the text on which the method was trained to produce the respective word vectors, we see that the usage of local word vectors in all RVA variants outperforms the experimental results where collection word vectors or pretrained word vectors are used. More specifically, in Semeval RVA-A-A with local word vectors achieves approximately 0.37 in the F_1 score, whereas with the utilization of collection word vectors or pretrained ones performs worse, with 0.34 and 0.30 scores, respectively. For the large dataset of Krapivin, the results of RVA-A-A have again the same ranking, i.e., the settings with local word vectors are the winners (0.32), followed by the setup of collection word vectors (0.28). The cases with pretrained word vectors are once more last in the ranking (0.26 F_1 -measure). The results of RVA-F-F and RVA-A-F follow the same ordering. However, the differences in the usage of local, collection and pretrained word vectors range at higher levels for RVA-F-F and lower for RVA-A-F.

F_1 -measure						
Setup	Semeval			Krapivin		
	RVA-A-A	RVA-F-F	RVA-A-F	RVA-A-A	RVA-F-F	RVA-A-F
LOC-50	0.36815	0.29543	0.11353	0.32062	0.21171	0.06265
LOC-200	0.36493	0.29641	0.11259	0.31999	0.20984	0.06212
CV-50	0.34202	0.23608	0.06779	0.28149	0.14415	0.03893
CV-200	0.34122	0.23535	0.06904	0.28267	0.15075	0.03944
PV-50	0.30188	0.15535	0.02026	0.25903	0.09089	0.01488
PV-200	0.30015	0.15505	0.02209	0.25804	0.09078	0.01583

Table 3: Experimental results for RVA-A-A, RVA-F-F and RVA-A-F using different settings for the GloVe method. The 2nd, 3rd and 4th columns concern the Semeval dataset, whereas the results of the last 3 columns correspond to the articles of the Krapivin dataset.

Regarding the use of local word vectors instead of the pretrained ones, the

experimental results essentially confirm that a small text (e.g. in the size of a scientific publication that refers to a specific subject) is possible to offer a sufficient basis to GloVe for the keyphrase extraction task. That is due to the fact that a scientific article includes a complete textual description of a specific topic, usually structured, using a limited but adequate vocabulary to reflect the semantic context of its words. Despite the fact that the pretrained word vectors succeed in capturing more general meanings as well as the underlying concepts that distinguish the words, performing well in tasks like the one of word analogy, we note that in the task of keyphrase extraction, they have low performance. Apparently, the generalization that is included in the pretrained word vectors is unnecessary and incorporates probably “noise” (redundant information). The above claim is also corroborated by the results when collection word vectors are used; the more local the word vectors, the better are the results.

At this point, we focus on the first two columns of each dataset trying to explain why we prefer the title and the abstract as the most suitable part of the article for the calculation of the reference vector rather than the full-text. As we can see, RVA-A-A clearly outperforms RVA-F-F in all cases. Indicatively, we mention that for the Semeval the RVA-A-A LOC-50 achieves approximately 0.37 while for the RVA-F-F LOC-50 the F_1 score equals to 0.30. For the Krapivin dataset, we have exactly the same ranking, however, with a higher difference between the two RVA variants (greater than 0.10). The intuition is that, possibly, there are frequent words in the full-text which play an important role in the text structure and in the content’s presentation. However, these words have an auxiliary role and they could not be considered as keywords. Unfortunately, when we extract candidates from the whole article, the importance of such words is reflected in the computation of the reference vector. For this reason, we conclude that it is much safer to use only the title and the abstract instead of the full-text, or generally speaking, parts of a limited text that contain some semantically significant and meaningful words, avoiding in this way the noise of the full-text.

The final issue that we discuss in this section is whether the candidate key-

words should come from abstracts or from full-texts regardless of the calculation of the reference vector, i.e., keeping as reference vector the mean word vector of the title and the abstract. Focusing on the 1st and the 3rd column of each dataset, we notice that there is a dramatic reduction in the rates in all the cases of RVA-A-F where we take into account as candidate keywords, the words from the full-text article (greater than -0.24). The above results lead us to the conclusion that it is more efficient for all the candidate words to participate in the calculation of the reference vector, as this facilitates the keywords’ detection.

4.4. Comparison with Other Methods

We compare the standard version of RVA (GloVe vectors of dimension 50 trained on each full-text) to TfIdf and 4 graph-based approaches, namely SingleRank (Wan & Xiao, 2008), TopicRank (Bougouin et al., 2013), WordAttractionRank (WARank) (Wang et al., 2014) and its extended version (WARank2015) (Wang et al., 2015). TfIdf, SingleRank and TopicRank are considered state-of-the-art methods for keyphrase extraction (Kim et al., 2013; Hasan & Ng, 2014). The *document frequency* used by TfIdf approach is calculated separately for each dataset collection. Graph-based methods are employed using their default parameters as finally set in the corresponding papers. For WARank and WARank2015, we used the pretrained word embeddings from (Collobert et al., 2011), which were also used in (Wang et al., 2014) and (Wang et al., 2015). We experimented with two versions of each competitor of RVA, one using the abstracts and one using the full-texts of each article.

4.4.1. Results

Table 4 shows the F_1 score of each method in each of the two datasets, sorted in descending order. We first notice that in both datasets the full-text version of TfIdf is much better than the abstract version. This is no surprise, as abstracts do not contain enough text to enable the separation of keyphrases from non-keyphrases in contrast with the full-text of articles. For the 4 graph-based methods, the opposite is observed, similarly to what we noticed for RVA

in Section 4.3. It appears that, despite their smaller size, abstracts capture adequately the co-occurrence (proximity) of words that is necessary for graph creation, avoiding at the same time the noise (lots of unimportant words) in the full-texts. We focus on the best versions of TfIdf and the 4 graph-based approaches in the rest of this section.

We then notice that RVA achieves the best place in both datasets. SingleRank-ab is 2nd in Krapivin and 3rd in Semeval. TfIdf-ft is 2nd in Semeval (without large difference from RVA) and 3rd in Krapivin (without large difference from SingleRank-ab). The other 3 graph-based methods follow in positions 4 to 6, without large differences among them. WARank-ab is better than WARank2015-ab in both datasets. We therefore focus on WARank-ab only in the rest of this section.

Method	Semeval	Method	Krapivin
RVA	0.36815	RVA	0.32062
TfIdf-ft	0.36114	SingleRank-ab	0.27795
SingleRank-ab	0.33043	TfIdf-ft	0.27668
WARank-ab	0.32797	WARank-ab	0.27436
TopicRank-ab	0.32571	WARank2015-ab	0.27365
WARank2015-ab	0.32553	TopicRank-ab	0.27038
TopicRank-ft	0.32044	TfIdf-ab	0.23196
SingleRank-ft	0.28401	SingleRank-ft	0.23088
WARank2015-ft	0.27799	TopicRank-ft	0.23032
TfIdf-ab	0.26102	WARank2015-ft	0.18934
WARank-ft	0.22005	WARank-ft	0.16869

Table 4: Experimental results (F_1 measure) for the baseline TfIdf and the methods TopicRank, SingleRank, WARank, and WARank2015 as well as the proposed method RVA. The “-ab” at the end of the methods’ names implies that they are applied only on titles and abstracts, whereas the “-ft” means that keyphrases are extracted from the full-text of the articles. Methods are ordered in descending F_1 measure.

We further employ statistical tests to compare RVA against each one of

TfIdf-ft, SingleRank-ab, TopicRank-ab and WARank-ab. For each dataset and pair of methods, we test if the differences of the F_1 scores across articles are statistically significant. For Semeval, we used the paired-t-test, as the differences in the F_1 score of RVA and each other method in each article are approximately normally distributed according to the Shapiro-Wilk test at the 0.01 significance level (Table 5). For Krapivin, we used the Wilcoxon test, as the normality assumption on the differences of the F_1 scores is rejected by the Shapiro-Wilk test. Table 6 shows that RVA is significantly better than the competing methods in both datasets with the exception of TfIdf-ft in Semeval. Note however, that Semeval is a much smaller dataset (244 articles) compared to Krapivin (2304 articles).

A possible reason for the quite high performance of TfIdf-ft in Semeval is the fact that a considerable ratio of its articles’ “gold” keyphrases include stop words (see Fig. 4 in Section 4.1). RVA does not return phrases with stop words, while TfIdf-ft extracts all possible n-grams ($n \in \{1, 2, 3\}$) without excluding stop words.

Shapiro-Wilk (p-values)		
Method	Semeval	Krapivin
TopicRank-ab	0.153	≈ 0.000
SingleRank-ab	0.129	≈ 0.000
TfIdf-ft	0.221	≈ 0.000
WARank-ab	0.113	≈ 0.000

Table 5: Results from the Shapiro-Wilk test on the performance differences between RVA and the other methods.

4.4.2. Discussion

The graph-based methods with which we have experimented, first, construct a graph of words (SingleRank, WARank) or topics (TopicRank) based on the position of the words in the text. For example, SingleRank assigns weights to

P-values		
Method	Semeval (Paired-t-test)	Krapivin (Wilcoxon)
TopicRank-ab	≈ 0.000	≈ 0.000
SingleRank-ab	≈ 0.000	≈ 0.000
TfIdf-ft	0.240	≈ 0.000
WARank-ab	≈ 0.000	≈ 0.000

Table 6: Paired-t-test and Wilcoxon test results between RVA and the other methods. Each row of the table contains the comparison with a specific method.

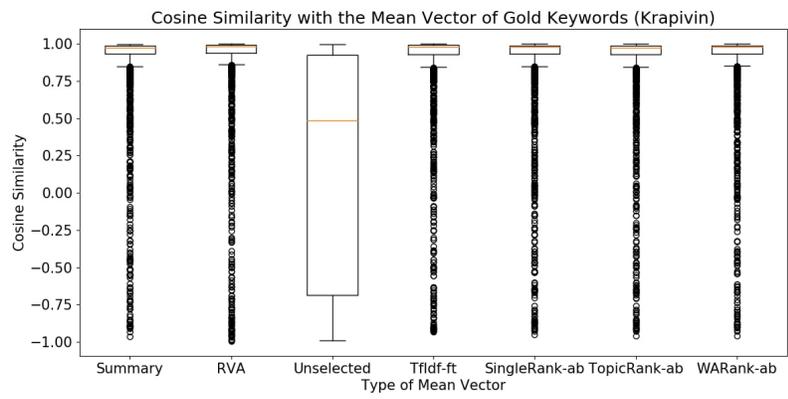
the graph edges utilizing the co-occurrence of words in a given window; TopicRank uses distances between words’ offset positions in the document; WARank exploits information that incorporates word frequencies as well as semantic distances between pretrained word embeddings. Then, the PageRank algorithm determines the final score of the words/topics (graph vertices), recursively, using information coming from the graph’s links. On the other hand, GloVe is an unsupervised method that produces word vector representations. Its aim is to learn word vectors such that their dot product is equal to the logarithm of the words’ probability of co-occurrence. Both graph-based approaches and GloVe capture information from the neighborhood of each one word (text statistics). However, in the context of our method, GloVe produces local word vectors, which is a *more expressive representation* than the assignment of a simple number as a score to each word by PageRank. This word vector representation allows us to express the article’s summary (title and abstract) with only one vector, which then guides the ranking of the candidate words as keywords. Moreover, the strong baseline TfIdf focuses on each word separately without taking into account any information related to the context of the words, but only to their frequency. As a result, TfIdf cannot capture any interrelationships between words in the text as well as word semantics.

Moreover, we propose an alternative (*semantic*) evaluation, focused on the “gold” keyphrases’ comparison with the returned keyphrases of the best systems

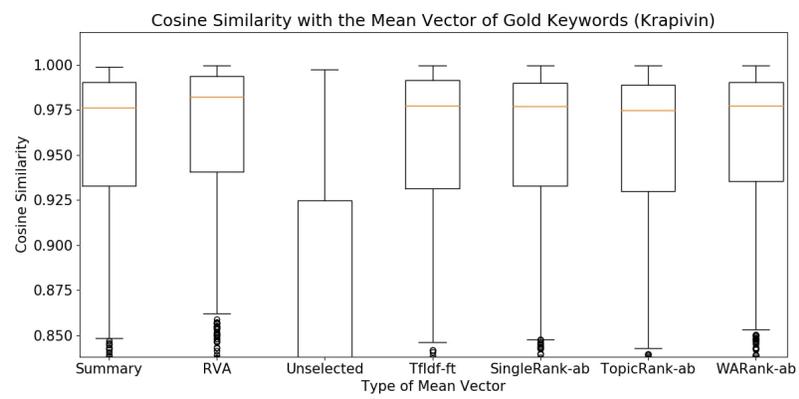
given above, that exploits the representation of the words as vectors. Particularly, considering for each article as the “gold” reference vector the mean word vector derived from the ground truth’s keyphrases, we compute the cosine similarity between this vector and the following:

- i. the mean vector of the summary (i.e. title and abstract) which is used as a reference vector by RVA,
- ii. the mean vector of the suggested keyphrases by RVA,
- iii. the mean vector of the rejected candidates by RVA, as they appear in a low ranking,
- iv. the mean vectors that come from the proposed keyphrases by the four best systems (i.e. TfIdf-ft, SingleRank-ab, WARank-ab, and TopicRank-ab), using the same local word vectors that are produced in the context of RVA.

Figures 5a, 5b and 6a, 6b show the results of the Krapivin and Semeval articles, respectively. In both datasets, the mean vector of RVA’s keyphrases achieves higher cosine similarity (RVA) than the reference vector (Summary) and the mean vector of the low ranked candidate n-grams (Unselected). Furthermore, the four mean vectors of the other methods have lower cosine similarities in the majority of the articles than RVA, except for TfIdf-ft in the Semeval where the similarity values are at the same levels. Generally, the results based on the F_1 -measure are consistent with the results of the semantic evaluation. The 50% of the mean vectors that are included in the “box” part of the plot, i.e., from the first (Q_1) to the third (Q_3) quartile seem to interpret the methods’ ranking according to the F_1 -measure which goes almost hand in hand with the semantic evaluation. Even the slight superiority in the Semeval dataset of RVA over TfIdf-ft seems to be determined by this “box” part of the plot. Particularly, for RVA’s box plot the Q_1 is equal to 0.9619 and the Q_3 is 0.9930, whereas for TfIdf the corresponding values are 0.9611 and 0.9924, respectively.

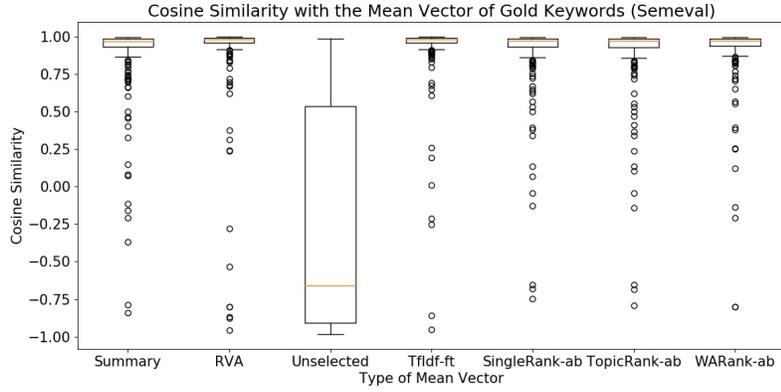


(a)

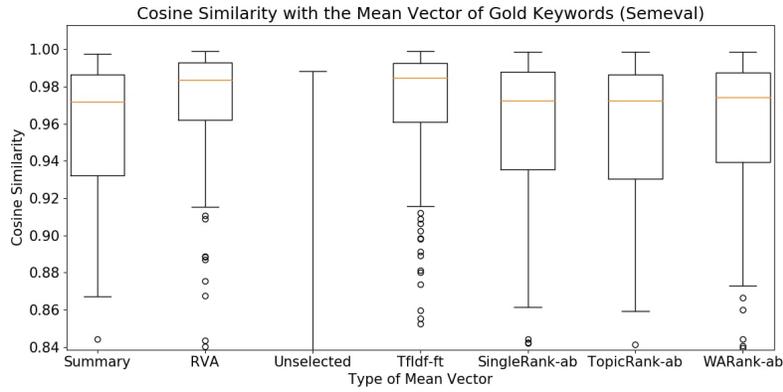


(b)

Figure 5: Semantic evaluation on the Krapivin dataset. (a) Quality evaluation of methods based on the word vector representation. (b) Close-up view.



(a)



(b)

Figure 6: Semantic evaluation on the Semeval dataset. (a) Quality evaluation of methods based on the word vector representation. (b) Close-up view.

4.5. Qualitative Results: RVA in Practice

In this section, we use RVA to extract the keyphrases of a publication based only on its title and abstract. This scientific article belongs to the Krapivin data collection. We quote its content below:

Title: Clustering for Approximate Similarity Search in High-Dimensional Spaces.
 Abstract: In this paper, we present a clustering and indexing paradigm (called Clindex) for high-dimensional search spaces. The scheme is designed for approximate similarity searches, where one would like to find many of the data points near a target point, but where one can tolerate missing a few near points. For such searches, our scheme can find near points with high recall in very few IOs and perform significantly better than other approaches. Our scheme is based on finding clusters and, then, building a simple but efficient index for them. We analyze the trade-offs involved in clustering and building such an index structure, and present extensive experimental results.

The corresponding set of the “gold” keyphrases are: {*clustering, approximate search, high-dimensional index, similarity search*}. For evaluation purposes, we transform the set of “gold” keyphrases into the following one (stemmed keyphrases):

$$\{(cluster), (approxim, search), (highdimension, index)(similar, search)\}$$

The RVA’s result set is given in the first box below, followed by its stemmed version in the second box. The candidate keyphrases are presented by descending cosine similarity score. The words that are both in the golden set and in the set of our candidates are highlighted with bold typeface:

{approximate similarity search, data points near, index structure, similarity search, approximate similarity, high recall, near points, points near, data points, finding clusters, search spaces, highdimensional search spaces, efficient index, target point, approximate similarity searches, near, indexing, search, highdimensional search, structure, present, data, clustering, recall}

{(**approxim, similar, search**), (data, point, near), (**index**, structur), (**similar, search**), (**approxim, similar**), (high, recal), (near, point), (point, near), (data, point), (find, **cluster**), (**search**, space), (**highdimension**, search, space), (effici, **index**), (target, point), (near), (**index**), (**search**), (**highdimension, search**), (structur), (present), (data), (**cluster**), (recal)}

The set of the returned keyphrases include all the words (unigrams) appearing in the “gold” keyphrases as well as additional keywords that have quite a strong role in the central meaning of the text. Furthermore, we notice that the RVA output is a set of keyphrases that are quite similar to each other, using

a quite limited set of words: {index, search, recal, target, point, similar, approxim, space, highdimension, structur, high, cluster, near, effici, data, find, present}.

We also give two box plots in Fig. 7d, which summarize in a simple way that the “gold” keywords (bigrams and trigrams are also flattened as unigrams) that appear in the candidates accumulate at high cosine similarity values. On the contrary, candidates that are not keywords cover a great range of cosine similarity values, as those words are not so close to the mean vector that is affected by words with a critical role (keywords).

Moreover, in Fig. 7, we give the box plots of the extracted candidate n-grams, the RVA’s output n-grams, and the “gold” keywords’ n-grams with their similarities to the reference vector, in a more thorough view. We provide 3 separate figures (7a, 7b, 7c) to present the similarity of the unigrams, bigrams, and trigrams with the reference vector. We see that all types of n-grams have an expected behavior; RVA’s n-grams are quite close to those of “gold” n-grams. In this way, we also confirm that the sum is a quite appropriate scoring function for bigrams and trigrams, as the corresponding similarities of the bigrams’ and trigrams’ mean vectors with the reference vectors are quite high, too.

For comparison reasons, we give the output produced by the baseline and the graph-based methods to provide a view of their keyphrases’ quality. In the previous section, we saw that it is better to use the articles summaries instead of the full-text for the keyphrase extraction task in order to avoid the noise of the full-text. For this reason, we present here, the best results of the graph-based algorithms (TopicRank-ab, SingleRank-ab, and WARank-ab) and the baseline (TfIdf-ab).

TopicRank-ab: {(scheme), (**cluster**), (**approxim**, **similar**, **search**), (near, point), (**highdimension**, **search**, space), (clindex), (paradigm), (high, recal), (data, point), (effici, **index**), (simpl), (**index**, structur), (mani), (build), (target, point), (**highdimension**, space), (paper), (tradeoff), (better), (present, extens, experiment, result), (point)}

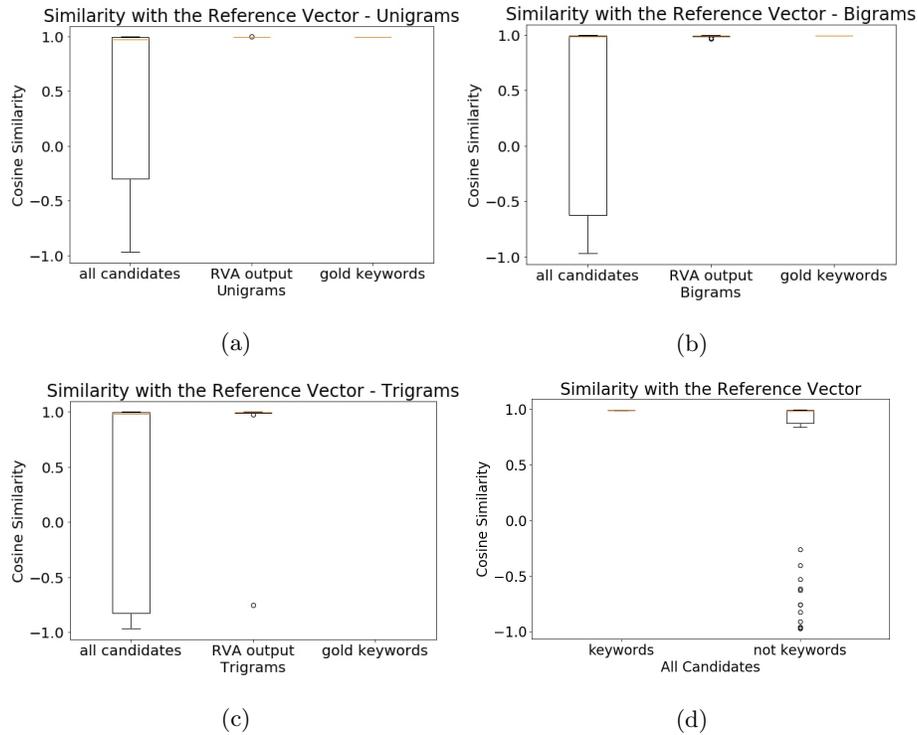


Figure 7: Boxplots of the extracted ngrams, (a) unigrams, (b) bigrams, (c) trigrams and their similarities to the reference vector. Figure (d) shows the similarity scores of all candidates as unigrams, grouped in “Keywords” and “not keywords”

SingleRank-ab: {(highdimension, search, space), (approxim, similar, search), (present, extens, experiment, result), (near, point), (highdimension, space), (index, structur), (data, point), (target, point), (effici, index), (point), (scheme), (high, recal), (cluster), (build), (tradeoff), (better), (mani), (paradigm), (clindex), (paper), (simpl)}

TfIdf-ab: {(**approxim**, **similar**, **search**), (near, point), (**approxim**, **similar**), (**similar**, **search**), (**highdimension**), (clindex), (**index**, paradigm), (**highdimension**, **search**, space), (**highdimension**, **search**), (call, clindex), (toler, miss), (data, point, near), (present, extens, experiment), (**cluster**), (find, **cluster**), (extens, experiment, result), (high, recal), (tradeoff, involv), (target, point), (effici, **index**), (**highdimension**, space), (io), (present, extens), (point, near)}

WARank-ab: {(**highdimension**, **search**, space), (**approxim**, **similar**, **search**), (such, **search**), (near, point), (target, point), (data, point), (point), (**highdimension**, space), (present, extens, experiment, result), (scheme), (few, io), (**index**, structur), (effici, **index**), (high, recal), (few), (build), (other, approach), (clindex), (t), (tradeoff), (better), (mani), (simpl), (**cluster**)}

The corresponding sets of unigrams that create the bigrams and the trigrams given above for each one method are presented below:

TopicRank-keywords: {*effici, point, approxim, high, cluster, paper, result, index, space, better, experiment, build, scheme, simpl, recal, highdimension, extens, data, present, clindex, search, target, structur, tradeoff, near, mani, paradigm, similar*}

SingleRank-keywords: {*effici, point, approxim, high, cluster, paper, result, index, space, better, experiment, build, scheme, simpl, recal, highdimension, extens, data, present, clindex, search, target, structur, tradeoff, near, mani, paradigm, similar*}

TfIdf-keywords: {*effici, point, approxim, high, cluster, result, io, miss, find, involv, index, space, experiment, call, recal, highdimension, extens, toler, data, present, clindex, search, target, tradeoff, near, paradigm, similar*}

WARank-keywords: {*near, point, data, target, approxim, similar, search, such, highdimension, space, present, extens, experiment, result, scheme, few, io, index, structur, effici, other, approach, clindex, build, high, recal, mani, better, simpl, cluster, t, tradeoff*}

We see that the number of words involved in the formation of the keyphrases returned by TopicRank, SingleRank, WARank, and TfIdf algorithms is 28, 28, 32, 27, respectively, whereas for RVA the number of words is equal to 17, i.e., RVA’s keyphrases revolve around a very specific and limited number of words instead of including additional redundant and irrelevant words like the other methods.

5. Conclusions and Future Work

This work presented a new unsupervised keyphrase extraction method, whose main innovation is the use of *local* word embeddings, in particular GloVe vectors, to represent candidate keyphrases. Our empirical study offered evidence that such a representation can lead to better keyphrase extraction results, compared to using global representations, either pretrained on large corpora or focused on a given target corpus, as well as compared to popular state-of-the-art unsupervised keyphrase extraction approaches.

We hope this work inspires other researchers to further investigate this novel local perspective of word embeddings. In particular we envisage implications of our work towards improved keyphrase extraction methods based on local word embeddings, towards applying local word embeddings to other information processing tasks, and towards developing novel methods for learning local word embeddings.

In the near future we intend to build on top of this work, and develop graph-based unsupervised keyphrase extraction methods as well as supervised keyphrase extraction methods that rely on local GloVe vectors. We would also like to develop a solution that manages to extract keyphrases from the full-text of academic publications without being affected by the noise and redundancy that it contains.

Acknowledgements

This work was partially funded by Atypon Systems Inc⁴.

⁴<https://www.atypon.com/>

References

References

- Augenstein, I., Das, M., Riedel, S., Vikraman, L., & McCallum, A. (2017). Semeval 2017 task 10: Scienceie - extracting keyphrases and relations from scientific publications. In *Proceedings of the 11th International Workshop on Semantic Evaluation, SemEval@ACL 2017, Vancouver, Canada, August 3-4, 2017* (pp. 546–555). URL: <https://doi.org/10.18653/v1/S17-2091>. doi:10.18653/v1/S17-2091.
- Bengio, Y., Ducharme, R., Vincent, P., & Janvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research*, 3, 1137–1155. URL: <http://www.jmlr.org/papers/v3/bengio03a.html>.
- Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, 3, 993–1022. URL: <http://www.jmlr.org/papers/v3/blei03a.html>.
- Boudin, F. (2016). pke: an open source python-based keyphrase extraction toolkit. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference System Demonstrations, December 11-16, 2016, Osaka, Japan* (pp. 69–73). URL: <http://aclweb.org/anthology/C/C16/C16-2015.pdf>.
- Bougouin, A., Boudin, F., & Daille, B. (2013). TopicRank: Graph-based topic ranking for keyphrase extraction. In *Proceedings of the 6th International Joint Conference on Natural Language Processing, IJCNLP 2013, Nagoya, Japan, October 14-18, 2013* (pp. 543–551). URL: <http://aclweb.org/anthology/I/I13/I13-1062.pdf>.
- Brin, S., & Page, L. (1998). The anatomy of a large-scale hypertextual web search engine. *Computer Networks*, 30, 107–117. URL: [https://doi.org/10.1016/S0169-7552\(98\)00110-X](https://doi.org/10.1016/S0169-7552(98)00110-X). doi:10.1016/S0169-7552(98)00110-X.

- Caragea, C., Bulgarov, F. A., Godea, A., & Gollapalli, S. D. (2014). Citation-enhanced keyphrase extraction from research papers: A supervised approach. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, Doha, Qatar, October 25-29, 2014, A meeting of SIGDAT, a Special Interest Group of the ACL* (pp. 1435–1446). URL: <http://aclweb.org/anthology/D/D14/D14-1150.pdf>.
- Chuang, J., Manning, C. D., & Heer, J. (2012). "without the clutter of unimportant words": Descriptive keyphrases for text visualization. *ACM Trans. Comput.-Hum. Interact.*, *19*, 19:1–19:29. URL: <http://doi.acm.org/10.1145/2362364.2362367>. doi:10.1145/2362364.2362367.
- Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. P. (2011). Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, *12*, 2493–2537. URL: <http://dl.acm.org/citation.cfm?id=2078186>.
- Deerwester, S. C., Dumais, S. T., Furnas, G. W., Harshman, R. A., Landauer, T. K., Lochbaum, K. E., & Streeter, L. A. (1989). Computer information retrieval using latent semantic structure. US Patent 4,839,853.
- Deerwester, S. C., Dumais, S. T., Landauer, T. K., Furnas, G. W., & Harshman, R. A. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, *41*, 391–407. URL: [https://doi.org/10.1002/\(SICI\)1097-4571\(199009\)41:6<391::AID-ASI1>3.0.CO;2-9](https://doi.org/10.1002/(SICI)1097-4571(199009)41:6<391::AID-ASI1>3.0.CO;2-9). doi:10.1002/(SICI)1097-4571(199009)41:6<391::AID-ASI1>3.0.CO;2-9.
- Dice, L. R. (1945). Measures of the amount of ecologic association between species. *Ecology*, *26*, 297–302. URL: <http://dx.doi.org/10.2307/1932409>. doi:10.2307/1932409.
- Florescu, C., & Caragea, C. (2017a). A new scheme for scoring phrases in unsupervised keyphrase extraction. In *Advances in Information Retrieval -*

- 39th European Conference on IR Research, ECIR 2017, Aberdeen, UK, April 8-13, 2017, Proceedings* (pp. 477–483). URL: https://doi.org/10.1007/978-3-319-56608-5_37. doi:10.1007/978-3-319-56608-5_37.
- Florescu, C., & Caragea, C. (2017b). PositionRank: An unsupervised approach to keyphrase extraction from scholarly documents. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, 2017, Volume 1: Long Papers* (pp. 1105–1115). URL: <https://doi.org/10.18653/v1/P17-1102>. doi:10.18653/v1/P17-1102.
- Gollapalli, S. D., & Caragea, C. (2014). Extracting keyphrases from research papers using citation networks. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence, Québec City, Québec, Canada, July 27 -31, 2014* (pp. 1629–1635). URL: <http://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8662>.
- Gollapalli, S. D., Li, X., & Yang, P. (2017). Incorporating expert knowledge into keyphrase extraction. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence, San Francisco, California, USA, February 4-9, 2017* (pp. 3180–3187). URL: <http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14628>.
- Gutwin, C., Paynter, G. W., Witten, I. H., Nevill-Manning, C. G., & Frank, E. (1999). Improving browsing in digital libraries with keyphrase indexes. *Decision Support Systems*, 27, 81–104. URL: [https://doi.org/10.1016/S0167-9236\(99\)00038-X](https://doi.org/10.1016/S0167-9236(99)00038-X). doi:10.1016/S0167-9236(99)00038-X.
- Hasan, K. S., & Ng, V. (2010). Conundrums in unsupervised keyphrase extraction: Making sense of the state-of-the-art. In *Proceedings of the 23rd International Conference on Computational Linguistics, COLING 2010, Beijing, China, August 23-27, 2010, Posters Volume* (pp. 365–373). URL: <http://aclweb.org/anthology/C/C10/C10-2042.pdf>.

- Hasan, K. S., & Ng, V. (2014). 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 2014, Baltimore, MD, USA, June 22-27, 2014, Volume 1: Long Papers* (pp. 1262–1273). URL: <http://aclweb.org/anthology/P/P14/P14-1119.pdf>.
- Herings, P. J.-J., Laan, G. v. d., & Talman, D. (2005). Measuring the power of nodes in digraphs. *Social Choice and Welfare*, 24, 439–454. URL: <https://doi.org/10.1007/s00355-003-0308-9>. doi:10.1007/s00355-003-0308-9.
- Hulth, A. (2003). Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing, EMNLP 2003, Stroudsburg, PA, USA, 2003* (pp. 216–223). Association for Computational Linguistics. URL: <https://doi.org/10.3115/1119355.1119383>. doi:10.3115/1119355.1119383.
- Jiang, X., Hu, Y., & Li, H. (2009). A ranking approach to keyphrase extraction. In *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2009, Boston, MA, USA, July 19-23, 2009* (pp. 756–757). URL: <http://doi.acm.org/10.1145/1571941.1572113>. doi:10.1145/1571941.1572113.
- Jones, K. S. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28, 11–21. URL: <https://doi.org/10.1108/00220410410560573>. doi:10.1108/00220410410560573.
- Kim, S. N., Medelyan, O., Kan, M., & Baldwin, T. (2010). Semeval-2010 task 5 : Automatic keyphrase extraction from scientific articles. In *Proceedings of the 5th International Workshop on Semantic Evaluation, SemEval@ACL 2010, Uppsala, Sweden, July 15-16, 2010* (pp. 21–26). URL: <http://aclweb.org/anthology/S/S10/S10-1004.pdf>.
- Kim, S. N., Medelyan, O., Kan, M., & Baldwin, T. (2013). Automatic keyphrase extraction from scientific articles. *Language Resources and Evalu-*

- ation, 47, 723–742. URL: <https://doi.org/10.1007/s10579-012-9210-3>. doi:10.1007/s10579-012-9210-3.
- Kleinberg, J. M. (1999). Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46, 604–632. URL: <http://doi.acm.org/10.1145/324133.324140>. doi:10.1145/324133.324140.
- Krapivin, M., Autayeu, A., & Marchese, M. (2008). Large dataset for keyphrases extraction. In *Technical Report DISI-09-055*. Trento, Italy.
- Liu, Z., Huang, W., Zheng, Y., & Sun, M. (2010). Automatic keyphrase extraction via topic decomposition. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, EMNLP 2010, MIT Stata Center, Massachusetts, USA, October 9-11, 2010, A meeting of SIGDAT, a Special Interest Group of the ACL* (pp. 366–376). URL: <http://www.aclweb.org/anthology/D10-1036>.
- Liu, Z., Li, P., Zheng, Y., & Sun, M. (2009). Clustering to find exemplar terms for keyphrase extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, EMNLP 2009, Singapore, August 6-7, 2009, A meeting of SIGDAT, a Special Interest Group of the ACL* (pp. 257–266). URL: <http://www.aclweb.org/anthology/D09-1027>.
- Lund, K., & Burgess, C. (1996). Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, & Computers*, 28, 203–208. URL: <https://doi.org/10.3758/BF03204766>. doi:10.3758/BF03204766.
- Medelyan, O., Frank, E., & Witten, I. H. (2009). Human-competitive tagging using automatic keyphrase extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, EMNLP 2009, Singapore, August 6-7, 2009, A meeting of SIGDAT, a Special Interest Group of the ACL* (pp. 1318–1327). URL: <http://www.aclweb.org/anthology/D09-1137>.

- Meng, R., Zhao, S., Han, S., He, D., Brusilovsky, P., & Chi, Y. (2017). Deep keyphrase generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, 2017, Volume 1: Long Papers* (pp. 582–592). URL: <https://doi.org/10.18653/v1/P17-1054>. doi:10.18653/v1/P17-1054.
- Mihalcea, R., & Tarau, P. (2004). TextRank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing, EMNLP 2004, Barcelona, Spain, July 25-26, 2004, A meeting of SIGDAT, a Special Interest Group of the ACL, held in conjunction with ACL 2004* (pp. 404–411). URL: <http://www.aclweb.org/anthology/W04-3252>.
- Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. In *Proceedings of the 2013 International Conference on Learning Representations, ICLR 2013, Workshop Track, Scottsdale, Arizona, USA, May 2-4, 2013*. URL: <http://arxiv.org/abs/1301.3781>.
- Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, Doha, Qatar, October 25-29, 2014, A meeting of SIGDAT, a Special Interest Group of the ACL* (pp. 1532–1543). URL: <http://aclweb.org/anthology/D/D14/D14-1162.pdf>.
- Rose, S., Engel, D., Cramer, N., & Cowley, W. (2010). Automatic keyword extraction from individual documents. *Text Mining: Applications and Theory*, (pp. 1–20). URL: <http://dx.doi.org/10.1002/9780470689646.ch1>. doi:10.1002/9780470689646.ch1.
- Rousseau, F., & Vazirgiannis, M. (2015). Main core retention on graph-of-words for single-document keyword extraction. In *Advances in Information Retrieval - 37th European Conference on IR Research, ECIR*

- 2015, Vienna, Austria, March 29 - April 2, 2015. *Proceedings* (pp. 382–393). URL: https://doi.org/10.1007/978-3-319-16354-3_42. doi:10.1007/978-3-319-16354-3_42.
- Sterckx, L., Caragea, C., Demeester, T., & Develder, C. (2016). Supervised keyphrase extraction as positive unlabeled learning. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016* (pp. 1924–1929). URL: <http://aclweb.org/anthology/D/D16/D16-1198.pdf>.
- Stubbs, M. (2003). Two quantitative methods of studying phraseology in english. *International Journal of Corpus Linguistics*, 7, 215–244. doi:10.1075/ijcl.7.2.04stu.
- Tomokiyo, T., & Hurst, M. (2003). A language model approach to keyphrase extraction. In *Proceedings of the 2003 ACL Workshop on Multiword Expressions: Analysis, Acquisition and Treatment, MWE 2003, Sapporo, Japan, 2003, Volume - 18* (pp. 33–40). URL: <https://doi.org/10.3115/1119282.1119287>. doi:10.3115/1119282.1119287.
- Wan, X., & Xiao, J. (2008). Single document keyphrase extraction using neighborhood knowledge. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008* (pp. 855–860). URL: <http://www.aaai.org/Library/AAAI/2008/aaai08-136.php>.
- Wang, R., Liu, W., & McDonald, C. (2014). Corpus-independent generic keyphrase extraction using word embedding vectors. *Software Engineering Research Conference*, (p. 39).
- Wang, R., Liu, W., & McDonald, C. (2015). Using word embeddings to enhance keyword identification for scientific publications. In *Databases Theory and Applications - 26th Australasian Database Conference, ADC 2015, Melbourne, VIC, Australia, June 4-7, 2015. Proceedings* (pp. 257–268). URL: https://doi.org/10.1007/978-3-319-19548-3_21. doi:10.1007/978-3-319-19548-3_21.

Witten, I. H. (2003). Browsing around a digital library. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms, Baltimore, Maryland, USA, January 12-14, 2003* (pp. 99–99). URL: <http://dl.acm.org/citation.cfm?id=644108.644125>.

Witten, I. H., Paynter, G. W., Frank, E., Gutwin, C., & Nevill-Manning, C. G. (1999). KEA: practical automatic keyphrase extraction. In *Proceedings of the 4th ACM conference on Digital Libraries, Berkeley, CA, USA, August 11-14, 1999* (pp. 254–255). URL: <http://doi.acm.org/10.1145/313238.313437>. doi:10.1145/313238.313437.