Local Word Vectors Guide Keyphrase Extraction

Eirini Papagiannopoulou · Grigorios Tsoumakas

Received: date / Accepted: date

Abstract Word vector representation techniques, built on word-word co-occurrence statistics, often provide representations that decode the differences in meaning between various words. This significant fact is a powerful tool that can be exploited to a great deal of natural language processing tasks. In this work, we propose a simple and efficient unsupervised approach for keyphrase extraction, called Reference Vector Algorithm (RVA) which utilizes a local word vector representation by applying the GloVe method in the context of one scientific publication at a time. Then, the mean word vector (reference vector) of the article’s abstract guides the candidate keywords’ selection process, using the cosine similarity. The experimental results that emerged through a thorough evaluation process show that our method outperforms the state-of-the-art methods by providing high quality keyphrases in most cases, proposing in this way an additional mode for the exploitation of GloVe word vectors.

Keywords Keyphrase extraction · unsupervised method · global vectors representation · GloVe · cosine similarity · mean word vector · reference vector · Reference Vector Algorithm · RVA · local word vectors

Mathematics Subject Classification (2000) 68T50

1 Introduction

Keyphrase extraction process constitutes one of the most important tasks as it is the cornerstone to obtain a general view of a text, to organize its information and finally, to hold on to the memory the basic knowledge it offers. Keyphrases can be assigned to a document by humans, after a careful reading of it. However,
the process that is followed to extract those phrases is not strictly defined as it is a mechanism connected with the comprehension of the text and characterized by great subjectivity. In most cases arbitrary number of phrases are chosen as keywords/keyphrases of a document, struggling to summarize in them all the basic topics covered. Certainly, the efforts for extracting all the important phrases with objectivity are often characterized by shortages and misunderstandings. Furthermore, the personal style of authors’ writing, the different vocabulary ranges, the peculiarities of each domain where a text refers to etc., are a few of the obstacles we have to overcome.

The usefulness of keyphrases is evident not only from the popularity of this research task but also from the benefits that such informative descriptions can offer to a great number of considerable applications. For example, text indexing could take advantage of the text keywords by exploiting them as a basis or websites’ keyphrases could guide their users during browsing collections, to get a feel of their content or to extend their queries. These few use cases mentioned above indicate the crucial position of the keyphrases in information retrieval field both in query expansion and document similarity measurements [9, 29]. Furthermore, scientific publishers are also very interested in this field as they can exploit this very useful compact information to recommend articles to readers, to highlight missing citations to authors, to analyse research trends etc [1].

Trying to describe clearly the task, some of the most common ways for automated keyphrase recommendation is either to select the appropriate concepts from a controlled vocabulary, i.e., a predefined set of keyphrases, or to extract the most representative terms from the body of the text document. Another available possibility is a hybrid approach which generates keyphrases that are not limited to those appeared in the body of the text document but also include keyphrases that use synonyms out of text or have the right keywords in a different, meaningful order.

In this paper, inspired by the unsupervised graph-based methods (see Section 2.1.1) that are built on the word co-occurrence relations within a window, we investigate whether a word vector representation based on word-word co-occurrence statistics, particularly the one generated by the GloVe (Global Vectors) technique, could reveal the special role of some words as keywords. We study the performance of the GloVe method in cases where the word vectors are produced in the context of one scientific publication at a time. Actually, keyphrase extraction from the abstracts of scientific publications is an ideal task in order to study whether the word vector representations generated by GloVe, within the narrow limits of a scientific publication, can encode the use and the role of the words in the specific text. The main idea is to depict a text segment with the mean word vector, called reference vector, by averaging the individual “local” word vectors of the corresponding segment. The fact that the top part of an article includes usually the title and the abstract implies that a great percentage of the total keywords appear on that piece of text. Therefore, the reference vector of this top part constitutes its central reference point and the cosine similarity between that and each individual local word vector of the abstract is sufficient to determine the presence of a word in the set of the candidate keyphrases or its removal from it.

In the context of the evaluation process, we apply a preprocessing stage to popular data sets used in the task of keyphrase extraction in order to separate the upper part (title, abstract, author’s keywords where they exist) of each doc-
ument from the remaining part (main text body), making them available to the community for future reuse. We conduct an experimental study by comparing our approach with other state-of-the-art methods followed in the last decades in this challenging research area. For the most accurate and thorough evaluation, we adopt a classic approach to measure the performance using the traditional metrics of precision, recall and $F_1$-score, whereas we also propose an additional evaluation framework, oriented to a self-evaluation of each candidate document for keyphrase extraction. The results suggest that our approach outperforms all the other methods providing steadily high quality keywords and keyphrases. We should notice that this methodology can be applied on texts of any scientific domain.

The rest of the paper is organized as follows. Section 2 gives a review of the related work on the field of keyphrase extraction as well as a brief reference to the methods that produce word embeddings. Section 3 discusses the subjectivity that characterizes the task and Section 4 introduces the proposed method. Section 5 describes the data set collections, the experimental set up and presents the results we obtained based on two evaluation frameworks, a standard and a proposed one. Finally, Section 6 concludes our paper and summarizes its contributions, whereas Section 7 refers to future work directions.

2 Related Work

2.1 Automatic Keyphrase Extraction

Automatic keyphrase extraction is a well studied task and a lot of techniques have been proposed in the past. In this section, we present both supervised and unsupervised methods in a comprehensive, structured way describing the most representative approaches of each category.

2.1.1 Unsupervised Approaches

Unsupervised approaches usually follow a standard process [10, 11]. The first stage of an unsupervised keyphrase extraction system is to choose the candidate lexical units with respect to some heuristics e.g. exclusion of stopwords or selection of words that are only nouns and adjectives. The second stage is to rank these lexical units by measuring the “importances” of them through co-occurrence statistics or syntactic rules etc. The final stage is the keyphrase formation where top-ranked lexical units are selected either as keywords or for the formation of longer keyphrases.

First of all, $TfIdf$ approach is used as a baseline and proposes a way to have the most representative words according to their frequency in a particular document, considering simultaneously their occurrences in other documents of a collection. The family of the 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 e.g. Google’s PageRank [27], Positional Function [12], HITS [16] etc. Particularly, TextRank [22] 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 starts to run
iteratively until it converges. Once the algorithm converges, nodes are sorted by decreasing order and the top $T$ nodes are chosen to form the final keyphrases. Variations of TextRank are SingleRank \[28\] where edges have a weight equal to the number of co-occurrences of their corresponding nodes, within a window and ExpandRank \[28\] where the graph constructed 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, and 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.

RAKE \[25\] is an interesting, domain-independent, and language-independent method for extracting keywords from individual documents. Particularly, 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, by detecting pairs of keywords that adjoin one another at least twice in the same document, in the same order, creating, a new candidate keyphrase which contains the corresponding interior stop words.

Furthermore, there is a group of approaches that incorporate knowledge from citation networks in order to extract representative keyphrases for a specific document, by capturing the information of short text descriptions that surround other papers’ mentions (citation contexts). CiteTextRank \[8\] is a representative method of this family which constructs a weighted graph considering such type of information.

In addition to the unsupervised methods mentioned above, there is the topic-based clustering group of methods whose motivation is to extract keyphrases that cover all the major topics of a document. A known technique here is KeyCluster \[18\] 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 \[4\] is another method that extracts keyphrases from the most significant topics of a document. First, the text of our 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) \[19\] is an alternative methodology which first obtains the topics of words and documents using LDA \[3\] 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.

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 [30] 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 in [13]. For each candidate phrase of the training data, that has been selected in an earlier stage, four features are calculated: within-document frequency, collection frequency, 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 [20], first determines all n-grams up to 3 words and then calculates a set of meaningful features such as $TfIdf$, position of the first occurrence, keyphraseness, phrase length and features based on wikipedia statistics which are used in its classification model. In [5] a binary classification model, CeKE, has been proposed (Naive Bayes classifier with decision threshold 0.9) which utilizes novel features from information of citation contexts and existing features from previous works.

Other approaches that use neural network models have been also 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 [21]. 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 [14]. 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.

2.2 Dense Vectors

Since 1990, a great number of methods have been proposed for words’ representations such as the popular Latent Dirichlet Allocation (LDA) [3] and Latent Semantic Analysis (LSA) [7, 6]. However, Bengio et al. [2] 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 in 2013 by Mikolov et al. in [23] where they presented the well-known Continuous Bag-of-Words Model (CBOW) and the Continuous Skip-gram Model, establishing widely the use of pre-trained embeddings.

In this work, we utilize the GloVe (Global Vectors) [24] 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 A Deeper Look to the Subjectivity of the Task

Keywords or keyphrases are ubiquitous and are searched in any part of the text from the title, paper’s abstract and main body of the articles to books etc., keywords are everywhere, holding always a prominent position. Most techniques provide a quite small number of keyphrases (usually 5-20), a number that seems to be pretty subjective and arbitrary at the same time. This is obvious in a number of evaluation studies that have been conducted such as [30, 21]. Moreover, we have to admit that the nature of “what is a keyword and what not”, it is not well-defined with clarity, purely based in many cases on the readers’ or authors’ opinion.

As an example of the above claims, we can see the following text, an abstract with its title that summarizes an article of the citeulike180 data set (its file name is 43.txt). Below the text, we provide 3 alternative sets of keyphrases that have been assigned to the article by 3 different taggers (tagger1, tagger130, tagger423), during the creation of the data set.

Title: Reverse Engineering of Biological Complexity
Abstract: Advanced technologies and biology have extremely different physical implementations, but they are far more alike in systems-level organization than is widely appreciated. Convergent evolution in both domains produces modular architectures that are composed of elaborate hierarchies of protocols and layers of feedback regulation, are driven by demand for robustness to uncertain environments, and use often imprecise components. This complexity may be largely hidden in idealized laboratory settings and in normal operation, becoming conspicuous only when contributing to rare cascading failures. These puzzling and paradoxical features are neither accidental nor artificial, but derive from a deep and necessary interplay between complexity and robustness, modularity, feedback, and fragility. This review describes insights from engineering theory and practice that can shed some light on biological complexity.

Table 1: Keyphrases assigned by 3 different taggers to the article with title “Reverse Engineering of Biological Complexity” (i.e., the file name in the citeulike180 data set is 43.txt)

| 1st tagger       | 130th tagger       | 423rd tagger       |
|------------------|--------------------|--------------------|
| robustness       | systems biology    | systems biology    |
| engineering      | systems            | robustness         |
| control          | biological model   | review             |
|                  | robustness         | regulatory network |
|                  | emergence          |                    |

4 The Reference Vector Algorithm

In this section, we describe thoroughly our keyphrase extraction approach from scientific articles’ abstracts, called Reference Vector Algorithm (RVA), that exploits the GloVe word vector representation to detect the candidate keywords and finally, to provide a complete set of representative keyphrases for a particular abstract. This is a simple and cost-effective methodology that also helps evaluate our initial question, whether a word vector representation based on co-occurrence statistics could reveal the role of the word terms in a text segment. The fact that the title
and the abstract in the vast majority of the cases summarizes the document’s content indicates with certainty that a great percentage of the total keywords appear on that text segment.

4.1 The Methodology

The outline of the RVA, that extends in the subsections below, is rigorously followed in order to extract the keyphrases from an article's abstract and it is not limited to any scientific domain.

4.1.1 Local Word Vector Generation and Text Preprocessing

As a first step, we should produce the local word vectors by applying the GloVe model to the candidate full text scientific publication from which we would like to extract the abstract’s keyphrases. In this way, the word vector representations generated by GloVe in the context of one article only (that’s why we call them “local” word vectors) encode the role of words as expressive means of writing, via a vector representation, by capturing the way that 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, “building” in this way an overview of the neighborhood of each one word and, simultaneously, providing us with a picture for 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.

After the word vectors’ production based on the whole text of the document, it is very important to clean the text from any strange character. Specifically, we clear the text segment we are interested in (i.e., the title and the abstract) by keeping only the printable characters, i.e., digits, letters, punctuation, and whitespaces. Finally, the word tokenization is the last stage of the text preprocessing.

Note that most approaches include the Part-of-Speech (PoS) tagging stage (a process which often should be done on each sentence separately) based on the observation that the lexical units which belong to a keyphrase 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 [13]. 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.

4.1.2 Candidate Keyphrases’ Production

Our methodology focuses only on extracting unigrams, bigrams and trigrams, as these are the most frequent lengths of keyphrases that are met in the data sets used in the experimental study (see Section 5.1 for more details about the data sets’ statistics).
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, without any particular restrictions),
- they do not belong to the stopwords list defined by us,
- they are not numbers.

No other term frequency conditions are necessary.

Candidate Bigrams: We choose as candidate bigrams those whose words are in candidate unigrams and appear in the text in that specific sequence. Actually, the sequence of words in a bigram does not play an important role, since our method first assigns scores to unigrams and then uses the existing scores to assign a rating to bigrams.

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

4.1.3 Scoring the Candidate Keyphrases

We compute the mean vector of the text segment, called reference vector by averaging the individual local word vectors that appear in the text. First, we aggregate all the word vectors which match to lexical units that exist in the set of the candidate unigrams formed in the previous step. Then, the reference vector is derived by dividing with the number of unigrams contained in the text segment. 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 word stems and their corresponding cosine similarity scores.

As a scoring function for a candidate bigram or trigram, we chose 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. In this way, we expand the mapping mentioned above with the stemmed bigrams/trigrams and their corresponding score.

4.1.4 Final Set of Keyphrases

We propose the number of keyphrases to be determined based on the size of the text segment. We prefer to choose a more flexible threshold for the number of the representative phrases that will be returned as keyphrases which is inspired by [22], 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 text segment, rather than to set a fixed number.

4.2 RVA in Practice

In this section, we apply the RVA to extract the keyphrases of a publication based only on its title and abstract. This scientific article belongs to the Krapivin2009 data collection. We quote its content below:
Title: Asynchronous Parallel Prefix Computation.
Abstract: The prefix problem is to compute all the products \( x_1 \odot x_2 \odot \cdots \odot x_n \), for 1 \( \leq \) n, where \( \odot \) is an associative binary operation. We start with an asynchronous circuit to solve this problem with \( O(\log n) \) latency and \( O(n \log n) \) circuit size, with \( O(n) \) \( \odot \)-operations in the circuit. Our contributions are: 1) a modification to the circuit that improves its average-case latency from \( O(\log n) \) to \( O(\log \log n) \) time, and 2) a further modification that allows the circuit to run at full-throughput, i.e., with constant response time. The construction can be used to obtain a asynchronous adder with \( O(\log n) \) worst-case latency and \( O(\log \log n) \) average-case latency.

The corresponding set of the “gold” keyphrases are: \{binary addition, asynchronous circuits, prefix computation, average-case latency\}. For evaluation purposes, we transform the set of “gold” keyphrases into the following one (stemmed keyphrases):

\{\text{binary, addition}, \text{asynchronous, circuits}, \text{prefix, computation}, \text{average-case, latency}\}

The result set produced as output by our method is the following set of candidate keyphrases which are presented by descending cosine similarity score (the words that exist both in the golden set and in the set of our candidates are highlighted in bold):

\{(\text{parallel, prefix, computation}), (\text{asynchronous, parallel, prefix}), (\text{asynchronous, circuit}), (\text{prefix, computation}), (\text{prefix, problem}), (\text{abstract, prefix, problem}), (\text{circuit, size}), (\text{binary, operation}), (\text{asynchronous, parallel}), (\text{parallel, prefix}), (\text{asynchronous, adder}), (\text{associative, binary, operation}), (\text{circuit}), (\text{compute}), (\text{asynchronous}), (\text{worst-case, latency}), (\text{latency}), (\text{average-case, latency}), (\text{time}), (\text{problem}), (\text{binary}), (\text{use}), (\text{abstract, prefix})\}

We also give at this point two box plots in Fig. 1 that summarize in a simple way that the keywords (only unigrams) of the golden set accumulate at high cosine similarity values, with a mean value at equally high levels. On the contrary, words that are not keywords cover a great range of cosine similarity values which is reasonable enough, as those keywords are not so close to a mean vector that is affected by words with crucial role (keywords).

5 Experiments

5.1 Data Sets and their Statistics

There are some publicly available data sets for the evaluation of the keyphrase extraction task. We experimented with 2 quite popular data sets. The largest one is the Krapivin2009 [17], which contains 2304 scientific articles. It is a set of full texts scientific papers, which has author-assigned and editor-corrected keyphrases. The second data set that is used in our experiments is the Semeval2010 [15] which includes 244 scientific full text publications. Except for the author-assigned keyphrases, there are additional reader-assigned keyphrases for each paper of this collection.

As we are working on the keyphrase extraction part, i.e., we try to detect keyphrases that exist in texts, we studied in detail the annotated with keywords and keyphrases datasets in order to have a complete view of the annotations’ presence at the top part of the documents which contain the titles and the abstracts of the texts. According to those observations, we made various decisions about...
Fig. 1: The unigram keywords of the golden set accumulate at high cosine similarity values (left box plot), whereas words that are not keywords cover a great range of cosine similarity values (right box plot).

the methodology we propose. In a question-answer format, we present the useful knowledge offered by our data.

**Question 1:** Which is the percentage of the gold keyphrases that appears in abstracts?

In Fig. 2 we present the percentage of the “gold” keyphrases appeared in the scientific publications’ abstracts both in data collections of Semeval2010 and Krapivin2009. Each one of the four boxplots displays the full range of variation, the likely range of variation and the mean value of the original data sets and their corresponding subsets which are selected in order to create suitable experimental data sets. As we can see in that figure, there are some abstracts in the datasets that are not appropriate for the specific task, since they contain a small percentage of the “gold” keyphrases. The fact that we work on the keyphrase extraction from texts leads us to remove, from both collections, the documents that contain less than 50% of their corresponding keyphrases within their title and abstract, leaving the Semeval data set with 106 scientific publications (referred as Semeval Filtered in Fig. 2) and the Krapivin data set with 1468 scientific articles (Krapivin Filtered). In this way, we believe that the experimental results give a more explicit view and clear interpretation.

**Question 2:** What is the number of gold keyphrases that has been assigned to the datasets’ abstracts?

Figure 3 shows that in Semeval2010 data set an average number of 14 keyphrases is assigned per document, whereas the main range of values is from 13 to 17 keyphrases, a range that is generally preserved after the filtering of the documents.
Fig. 2: The four boxplots display the full range of variation, the likely range of variation and the mean value of the original data sets and their corresponding produced subsets (filtered). The 1st and the 2nd box plots correspond to the data set of Semeval2010 and 3rd and the 4th to the Krapivin2009 data set.

A similar situation, where in general terms the average number of keyphrases is maintained after the removal of the inappropriate documents from the initial data set, is observed in the Krapivin2009 dataset, too. In that case the main range of values varies from 4 to 6, with a mean value which approximates to 5.

**Question 3:** Which is the common length of a keyphrase?

In Table 2, we see the number of occurrences per keyphrase length expressed in n-grams for each one data set. As we can see, unigrams, bigrams and trigrams are the most frequent lengths of keyphrases that are met in the data sets. For this reason, RVA focuses only on extracting such types of n-grams.

| Data sets          | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | 10 |
|--------------------|----|----|----|----|----|----|----|----|----|----|
| Semeval            | 759| 2005| 782| 171| 46 | 16 | 3  | 2  | 1  | 0  |
| Semeval Filtered   | 390| 805 | 336| 70 | 23 | 9  | 1  | 0  | 0  | 0  |
| Krapivin           | 2530| 7575| 1936| 364| 70 | 18 | 2  | 1  | 0  | 0  |
| Krapivin Filtered  | 1640| 4676| 1164| 207| 37 | 9  | 0  | 1  | 0  | 0  |

Table 2: Length of keyphrases per data set in n-grams. The value of $n$ ranges from 1 to 10.
5.2 Evaluation Process

5.2.1 Basic Evaluation Approach

We convert the golden keyphrases into unigrams in order to easily calculate the $F_1$-measure between the golden keyphrase set and the set of extracted keyphrases which are also converted into unigrams, an evaluation process that is quite fair and followed in [26]:

$$F_1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}},$$

where

$$\text{precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}},$$

$$\text{recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}.$$

5.2.2 An Alternative Evaluation Framework

We propose an alternative, more flexible evaluation framework that assesses the quality of n-grams, i.e., the unigrams, bigrams and trigrams without “flattening”
the phrases by transforming them into unigrams, that computes the $F_1$-measure, counting the True Positives (TPs), False Positives (FPs) and False Negatives (FNs) in the following way:

TPs and FPs Calculation

For each keyphrase returned by our system, we conduct the following process:

- If the keyphrase is a unigram, we check whether it appears in any of the keyphrases (unigrams, bigrams, trigrams) of the golden set in order to increase the TPs counter by 1.
- If the keyphrase is a bigram, we check if both the 2 words of the bigram appear in the golden set of keyphrases as a single phrase, considering it as a TP (TPs counter increases by 1), otherwise we search for every word separately whether it appears as a unigram keyphrase in the golden set of keyphrases in order to increase the TPs counter by 1 for each such occurrence.
- If the keyphrase is a trigram, we check if 3 out of 3 the trigram’s words appear in the golden set of keyphrases as a single phrase, increasing the TPs counter by 1, otherwise for every possible pair of words, we check if it appears as a bigram keyphrase in the golden set of keyphrases in order to increase the TPs counter by 1 for each such occurrence. Finally, if none of the cases above holds, for each trigram’s word, we check if it appears as a unigram keyphrase in the golden set of keyphrases in order to increase the TPs counter by 1 for each such occurrence.
- In case the keyphrase is not a TP, the FPs counter is increased by 1.

FNs Calculation

For each golden keyphrase we conduct the following process:

- If the golden keyphrase is a unigram we check if it appears in any of the keyphrases (unigrams, bigrams, trigrams) returned by the system in order to consider it as TP.
- If the golden keyphrase is a bigram, we check if 2 out of 2 words of the bigram appear in the set of keyphrases returned by the system as a single phrase in order to consider it as TP.
- If the golden keyphrase is a trigram, we check if 3 out of 3 words of the trigram appear in the set of keyphrases returned by the system as a single phrase in order to consider it as TP.
- In case our keyphrase is not a TP, the FN counter is increased by 1.

5.3 Experimental Set up

5.3.1 GloVe Set up

For all our experiments, we used the default parameters ($x_{max} = 100$, $\alpha = \frac{3}{4}$, window size = 10), as they are set in the experiments of [24]. In addition, we ran experiments with the pretrained word vectors that were created by training on
Wikipedia 2014 + Gigaword 5 which have 400000 vocabulary, uncased. Particularly, we used the 50-dimensional vectors and the 200-dimensional vectors. 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, creating 10,50,200-dimensional vectors with 50 iterations as indicated in [24] for vectors smaller than 300 dimensions. Finally, we trained a model on the smaller data set collections of Semeval2010 and Krapivin2009, separately, as an intermediate size of corpus, generating 50,200-dimensional vectors with 50 iterations, as well. 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.

5.3.2 State-of-the-Art Approaches

To validate our methodology, we designed experiments that compare RVA to the strong baseline of TfIdf and 2 other stable and traditional unsupervised graph-based approaches that utilize PageRank algorithm to score the candidates, SingleRank and TopicRank, with their default parameters, as they have been finally set in [25] and [4], respectively. The document frequency used by TfIdf approach is calculated separately for each data set collection.

5.4 Results

5.4.1 Evaluation Based on Text Size for GloVe Training

In this section we give a general view of the performance of RVA algorithm by changing the dimension of the word vectors and training the GloVe model on different corpus sizes, including the usage of word vectors trained on massive web data sets. The Table 3 describes the different RVA’s settings, providing the corresponding abbreviations that are used in the results’ Tables 4 and 5. In all settings, the number of iterations for GloVe training is 50.

| Abbreviation | Description |
|--------------|-------------|
| RVA-10       | RVA - 10-dim vectors - trained on individual files |
| RVA-50       | RVA - 50-dim vectors - trained on individual files |
| RVA-200      | RVA - 200-dim vectors - trained on individual files |
| RVA-CV-50    | RVA - 50-dim vectors - trained on each collection separately |
| RVA-CV-200   | RVA - 200-dim vectors - trained on each collection separately |
| RVA-PV-50    | RVA - 50-dim pretrained vectors |
| RVA-PV-200   | RVA - 200-dim pretrained vectors |

Table 3: Explanation of the abbreviations used in the tables with the experimental results to describe the RVA’s settings.

Table 4 shows that based on the evaluation approach mentioned in [5.2.1], the RVA with local word vectors trained on individual files (RVA-10, RVA-50, RVA-200) achieves the highest $F_1$-measure across all the other RVA alternatives, i.e., with pretrained word vectors as well as with word vectors trained on the whole...
data set collections. More specifically, the RVA version with the pretrained vectors (RVA-PV-50, RVA-PV-200) has the worst performance in detecting the most representative words of a text, as keywords. In the second place, we meet the RVA with the word vectors from the collections’ text, a fact which indicates that the closer we move to the text boundaries of our interest’s publication, i.e., the GloVe builds the word vectors according to a specific syntactic structure and use of a limited vocabulary, the better reference word vectors are calculated that can distinguish the keywords from auxiliary words.

| Method     | SemEval | Krapivin |
|------------|---------|----------|
| RVA-10     | 0.40263 | 0.34185  |
| RVA-50     | 0.40199 | 0.34176  |
| RVA-200    | 0.39973 | 0.34088  |
| RVA-CV-50  | 0.37107 | 0.30320  |
| RVA-CV-200 | 0.37269 | 0.30535  |
| RVA-PV-50  | 0.33980 | 0.29097  |
| RVA-PV-200 | 0.33767 | 0.29235  |

Table 4: Evaluation results based on unigrams.

| Method     | SemEval | Krapivin |
|------------|---------|----------|
| RVA-10     | 0.42591 | 0.31058  |
| RVA-50     | 0.42101 | 0.31032  |
| RVA-200    | 0.42017 | 0.31072  |
| RVA-CV-50  | 0.31276 | 0.24536  |
| RVA-CV-200 | 0.31615 | 0.25126  |
| RVA-PV-50  | 0.25896 | 0.22838  |
| RVA-PV-200 | 0.26042 | 0.23184  |

Table 5: Evaluation results based on the proposed evaluation.

In Table 5 we give the experimental results based on the proposed evaluation framework, which is described in detail in Section 5.2.2. As we can see, we arrive at similar conclusions by observing the alternative evaluation results. In both evaluation approaches, we see that the differences in vector dimensions do not play an important role, causing only slight differences to the performance. The text, where GloVe is trained, decides the effectiveness of the word vectors as components of the basic reference vectors.

5.4.2 Comparison with the State-of-the-Art Algorithms

Table 6 shows the results of RVA with 10-dimensional word vectors trained on the individual files, which is an optimal strategy as far as the vectors’ dimension and especially the text size for GloVe training. According to the more classic evaluation framework, the results show that RVA significantly outperforms the baseline (TfIdf) and the 2 graph-based approaches, TopicRank and SingleRank.
whose performance is at the same level on the SemEval2010 and the Krapivin 2009 data sets. Furthermore, we observe a quite big gap between the results of the first position and the second position for both data sets. The efficiency and stability of the baseline to identify the keywords of a text even in size-limited documents is clearly reflected on the value of $F_1$-measure that is achieved by the TfIdf method on the summaries of Krapivin data collection. Actually, in that case the TfIdf deserves the second place in the overall ranking.

$$
\begin{array}{|c|c|c|}
\hline
\text{Method} & \text{Semeval} & \text{Krapivin} \\
\hline
\text{RVA-10} & 0.40263 & 0.34185 \\
\text{TopicRank} & 0.36892 & 0.29062 \\
\text{SingleRank} & 0.36908 & 0.29318 \\
\text{TfIdf} & 0.26647 & 0.31649 \\
\hline
\end{array}
$$

Table 6: Evaluation results based on unigrams.

The utility of the proposed evaluation framework is depicted in Table 7 where the general ranking of the methodologies is the same as that of the previous evaluation process shown above. However, this type of valuation of the TPs, FPs and FNs counters leads to calculation of metrics that emphasize the small differences between the various algorithms, making more stringent the counting of TPs through the adoption of rules, as they were formulated in the Section 5.2.2. Actually, such an evaluation framework described above defines a process that favors an approach when it extracts keyphrases that are correlated to each other and, at the same time, are in the right direction, based always on the golden set of keyphrases. On the other hand, it penalizes the methods that return related to each other keyphrases which are not in the right direction, by increasing the FPs and FNs counters.

$$
\begin{array}{|c|c|c|}
\hline
\text{Method} & \text{Semeval} & \text{Krapivin} \\
\hline
\text{RVA-10} & 0.42591 & 0.31058 \\
\text{TopicRank} & 0.36101 & 0.25334 \\
\text{SingleRank} & 0.36316 & 0.26097 \\
\text{TfIdf} & 0.33691 & 0.30868 \\
\hline
\end{array}
$$

Table 7: Evaluation results based on the proposed evaluation.

Again, RVA is the winner of the “battle”, leaving in the second place SingleRank which is followed by TopicRank and, finally, TfIdf, in the total rank for the Semeval2010 data set. As far as the Krapivin2009 data set concerned, RVA outperforms TfIdf but with a minor difference. Once again, the graph-based approaches are in the last places of the ranking. The fact that RVA with local word vectors wins the 2 graph-based approaches indicates that such a word vector representation probably can capture the information from the neighborhood of each one word better than state-of-the-art methods.
6 The Contributions of This Work

This work tries to give new perspectives in the usage of the word vector representations and especially, those produced by the GloVe, a method based on text co-occurrence statistics within a predetermined window. As we are interested in the Keyphrase Extraction task, we transfer the learning process of word vectors on one target text at a time; since our task is to build word vectors that contribute to the creation of a reference vector (mean vector) which “summarizes”, in some way, the relative positions of the words in the text, via a model that exploits the vector difference between two word vectors. The GloVe method is designed exactly for this purpose: to express such vector differences by the juxtaposition of two words.

Undoubtedly, the word vector representations produced in the context of Reference Vector Algorithm do not give suitable representations for tasks such as the word analogy task because there is not any training on large corpora in order to capture some form of meaning, i.e., the underlying concepts that distinguish one term from another. This is not an assumption we make, but arose after tests on data sets that are available on Github along with the complete code of the GloVe project.

Furthermore, we propose a new evaluation standard process that overcomes the obstacles of how to assess either n-grams that are part of a golden keyphrase or n-grams that are longer than and contain a golden keyphrase, proposing specific rules. Certainly, the refined data set collections that were created for the experimental purposes are online available for future use.

7 Future Work

In this paper, we explored the keyphrase extraction task from a different point of view. We let a non-generalized GloVe word vector representation to guide the keyphrase detection process by forming useful reference word vectors. We expect this work to inspire the research community in the direction of the document-oriented text representations. Finally, a possible extension of this work would be the investigation of an approach that would elicit keyphrases from full text articles utilizing a local word vector representation.

References

1. Augenstein I, Das M, Riedel S, Vikraman L, McCallum A (2017) Semeval 2017 task 10: ScienceIE-extracting keyphrases and relations from scientific publications. arXiv preprint arXiv:170402853
2. Bengio Y, Ducharme R, Vincent P, Janvin C (2003) A neural probabilistic language model. J Mach Learn Res 3:1137–1155, URL http://dl.acm.org/citation.cfm?id=944919.944966
3. Blei DM, Ng AY, Jordan MI (2003) Latent dirichlet allocation. Journal of machine Learning research 3(Jan):993–1022
4. Bougouin A, Boudin F, Daillé B (2013) TopicRank: Graph-based topic ranking for keyphrase extraction. In: International Joint Conference on Natural Language Processing (IJCNLP), pp 543–551
5. Caragea C, Bulgarov FA, Godea A, Gollapalli SD (2014) Citation-enhanced keyphrase extraction from research papers: A supervised approach. In: EMNLP, vol 14, pp 1435–1446
6. Deerwester S, Dumais ST, Furnas GW, Landauer TK, Harshman R (1990) Indexing by latent semantic analysis. Journal of the American society for information science 41(6):391
7. Deerwester SC, Dumais ST, Furnas GW, Harshman RA, Landauer TK, Lochbaum KE, Streeter LA (1989) Computer information retrieval using latent semantic structure. US Patent 4,839,853
8. Gollapalli SD, Caragea C (2014) Extracting keyphrases from research papers using citation networks. In: AAAI, pp 1629–1635
9. Gutwin C, Paynter G, Witten I, Nevill-Manning C, Frank E (1999) Improving browsing in digital libraries with keyphrase indexes. Decision Support Systems 27(1):81–104
10. Hasan KS, 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: Posters, Association for Computational Linguistics, pp 365–373
11. Hasan KS, Ng V (2014) Automatic keyphrase extraction: A survey of the state of the art. In: ACL(1), pp 1262–1273
12. Herings P, Van der Laan G, Talman D (2001) Measuring the power of nodes in digraphs. Technical Report
13. Hulth A (2003) Improved automatic keyword extraction given more linguistic knowledge. In: Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Stroudsburg, PA, USA, EMNLP ’03, pp 216–223, DOI 10.3115/1119355. 1119383, URL https://doi.org/10.3115/1119355.1119383
14. Jiang X, Hu Y, Li H (2009) A ranking approach to keyphrase extraction. In: Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval, ACM, pp 756–757
15. Kim SN, Medelyan O, Kan MY, Baldwin T (2010) Semeval-2010 task 5: Automatic keyphrase extraction from scientific articles. In: Proceedings of the 5th International Workshop on Semantic Evaluation, Association for Computational Linguistics, Stroudsburg, PA, USA, SemEval ’10, pp 21–26, URL http://dl.acm.org/citation.cfm?id=1859664.1859668
16. Kleinberg JM (1999) Authoritative sources in a hyperlinked environment. Journal of the ACM (JACM) 46(5):604–632
17. Krapivin M, Autayeu A, Marchese M (2008) Large dataset for keyphrases extraction. Technical Report
18. 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; Volume 1-Volume 1, Association for Computational Linguistics, pp 257–266
19. 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. Association for Computational Linguistics, pp 366–376
20. Medelyan O, Frank E, Witten IH (2009) Human-competitive tagging using automatic keyphrase extraction. In: Proceedings of the 2009 Conference on
Empirical Methods in Natural Language Processing: Volume 3-Volume 3, Association for Computational Linguistics, pp 1318–1327
21. 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 (Volume 1: Long Papers), Association for Computational Linguistics, pp 582–592, DOI 10.18653/v1/P17-1054, URL http://aclanthology.coli.uni-saarland.de/pdf/P/P17/P17-1054.pdf
22. Mihalcea T, Tarau P (2004) Textrank: Bringing order into texts. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, pp 404–411
23. Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient estimation of word representations in vector space. CoRR abs/1301.3781, URL http://arxiv.org/abs/1301.3781
24. Pennington J, Socher R, Manning CD (2014) Glove: Global vectors for word representation. In: In EMNLP
25. Rose S, Engel D, Cramer N, Cowley W (2010) Automatic keyword extraction from individual documents. Text Mining pp 1–20
26. 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, DOI 10.1007/978-3-319-16354-3_42, URL https://doi.org/10.1007/978-3-319-16354-3_42
27. Suresh K (1998) The anatomy of a large-scale hypertextual web search engine. Computer networks and ISDN systems 30(1–7):107–117
28. Wan X, Xiao J (2008) Single document keyphrase extraction using neighborhood knowledge. In: AAAI, vol 8, pp 855–860
29. Witten IH (1999) Browsing around a digital library. In: Proceedings of the Australasian Computer Science Conference, Auckland, New Zealand, Citeseer, pp 1–14
30. Witten IH, Paynter GW, Frank E, Gutwin C, Nevill-Manning CG (1999) KEA: Practical automatic keyphrase extraction. In: Proceedings of the fourth ACM conference on Digital libraries, ACM, pp 254–255