Key2Vec: Automatic Ranked Keyword Extraction from Scientific Articles using Keyphrase Embeddings

Anonymous
Anonymous

Anonymous
Anonymous

***Abstract*—Keyword extraction is a fundamental task in natural language processing that facilitates mapping of documents to a concise set of representative single and multi-word phrases. Keywords from text documents are primarily extracted using supervised and unsupervised approaches. In this paper, we present a novel and effective unsupervised technique named *Key2Vec* that extensively takes into account neural keyphrase embeddings for extracting and ranking keywords from scientific articles. We introduce an efficient way of processing text documents and training keyphrase embeddings for the purpose of extracting and ranking keywords. We share a new evaluation dataset used for choosing the underlying model for *Key2Vec*, which is based on semantic similarity of textual segments. The evaluations for ranked keyword extraction are performed on two benchmark datasets comprising of short abstracts (Inspec), and long scientific papers (SemEval 2010), and is shown to produce results better than the state-of-the-art systems. We also show how keyphrase embeddings could be further used for capturing semantic similarity between text documents, and for performing efficient extractive summarization using the *Key2Vec* framework.**

***Keywords*—keyword extraction, keyword assignment, ranked keyword extraction, information extraction, phrase embedding, keyphrase embedding, neural language model.**

I. Introduction

Keywords are single or multi-word linguistic units that represent the salient aspects of a document. In this paper, we use the term keyword uniformly to represent both single and multi-word phrases. Keywords are useful in many tasks such as indexing documents [1], summarization [2], clustering [3], ontology creation [4], classification [5], auto-tagging [6] and visualization of text [7]. Due to its widespread use, keyword extraction has received a lot of attention from researchers.

In order to push the state-of-the-art performances of keyword extraction systems, the research community has been hosting shared tasks like SemeEval 2010 Task 5 [8] and SemEval 2017 Task 10 [9]. However, the task is far from solved and the performances of the present systems are worse in comparison to many other NLP tasks [10]. Some of the major challenges are the varied length of the documents to be processed, their structural inconsistency and developing strategies that can perform well in different domains [11].

The task of ranked keyword extraction from scientific articles and identifying relationships between them is of great interest to scientific publishers as it helps to recommend articles to readers, highlight missing citations to authors, identify potential reviewers for submissions, and analyse research trends over time [9]. Although scientific articles usually provide them, most of the documents have no associated keywords. Therefore, the problem of automatically extracting the most representative keywords from scientific articles is an active field of research.

Methods for automatic keyword extraction are mainly divided into two categories: *supervised* and *unsupervised*. Supervised methods approach the problem of keyword extraction as a binary classification problem [11], whereas the unsupervised methods are mostly based on TFIDF, clustering, and graph-based ranking [12]. On presence of domain-specific data, supervised methods have shown better performance than the unsupervised methods. The unsupervised methods have the advantage of not requiring any training data and can produce results in any domain. However, the assumptions of unsupervised methods do not hold for every type of document.

With recent advancements in deep learning techniques applied to natural language processing, the latest trend is to represent words as dense real-valued vectors obtained by training a shallow neural network on a fixed vocabulary. These distributional representation of words are popularly known as word embeddings. These neural representations have been shown to equal or outperform other methods (e.g. LSA, SVD) [13]. The vector representation of words, are supposed to preserve the semantic and syntactic similarities between them. They have been shown to be useful for several natural language processing (NLP) tasks, like part-of-speech tagging, chunking, named entity recognition, semantic role labeling, syntactic parsing, and also for speech processing [14]. Some of the most popular approaches for generating word embeddings are Word2Vec [15], Glove [16] and Fasttext [17].

In this paper, we introduce the notion of *keyphrase embeddings*, and show how they could be trained in a simple and effective way using already existing algorithms belonging to the neural word embedding family (Word2Vec and Fasttext). Instead of distributional representation of words we produce distributional representation of keyphrases for the purpose of ranked keyword extraction from scientific articles. We call our methodology as *Key2Vec*. We also propose a text processing pipeline for training keyphrase embeddings, which enables intelligent tokenization of text into a mix of unigram and meaningful chunks of multi-gram tokens. We show how the embedding models trained using our strategy, can be advantageously used for representing keyphrases and text snippets as dense vectors that are further used for calculating semantic and syntactic similarities. We fully rely on training high quality embedding models and effectively use the similarities between the distributional representation of keyphrases for ranking representative keywords from scientific documents.

Table I, shows keywords extracted from a sample research article abstract, by using *Key2Vec*.

TABLE I: Keywords extracted by using *Key2Vec* from a sample research article abstract.

Title: Identification of states of complex systems with estimation of admissible measurement errors on the basis of fuzzy information.
Abstract: The problem of identification of states of complex systems on the basis of fuzzy values of informative attributes is considered. Some estimates of a maximally admissible degree of measurement error are obtained that make it possible, using the apparatus of fuzzy set theory, to correctly identify the current state of a system.
Automatically identified keywords: *complex systems, fuzzy information, admissible measurement errors, fuzzy values, informative attributes measurement error, maximally admissible degree, fuzzy set theory*
Manually assigned keywords: *complex system state identification, admissible measurement errors, informative attributes, measurement errors fuzzy set theory*

For the purpose of evaluating our embedding models on different similarity tasks, we develop a new evaluation dataset[1] (Section IV-A), from an exisiting dataset [18]. We share the dataset with the community believing that it would facilitate training of better keyphrase embedding models useful for ranked keyword extraction. Contrary, to the analogy task that is widely used for intrinsic evaluation of word embeddings, this dataset can be used for evaluating phrase and word embeddings on similarity tasks, whenever they are being trained for a downstream process that has semantic and syntactic similarity between distributional representation of text at its core. We perform such an evaluation and show that our keyphrase embeddings are more effective in capturing semantic similarity between text documents at lower dimensions, than normal word embeddings, when trained using the same parameters (Section V).

We not only aim at extracting the keywords that are statistically relevant for the given document, as mostly attempted by previous studies [11], but also to rightly identify meaningful keyphrases that are most semantically related to the main theme of the document. We are motivated by the following properties for the top-K keywords that we select using *Key2Vec* as mentioned in [19].

- *Understandability* - The ranked keywords should be understandable to readers, which is made possible by chosing grammatically correct and meaningful phrases that possess the characteristic of high readability by humans. For example, the phrase *scientific articles* has more understandability than the individual words *scientific* and *articles*.
- *Relevancy* - The top-K keywords should be semantically related to the central theme of the document.
- *Good Coverage* - The keywords should cover all the major topics discussed in the document.

This intrigued us to look at the route of training keyphrase embeddings as presented in this paper, rather than first training embedding models for unigram words and then combining their dense vector representations to obtain similar representations for multi-word phrases. Training keyphrase embeddings for extracting and ranking keywords is relatively a new problem and has not been thoroughly explored. Researchers have just started exploring the advantages of neural word embeddings for representing semantic similarities between words that are further leveraged in graph-based ranking algorithms, in order to rank extracted keywords [20]. To our knowledge, keyphrase embeddings have not been used for ranked keyword extraction and summarization, and this work is the first attempt to do so.

Some of the major contributions of this paper are,

- *Intelligent processing of text for training neural keyphrase embeddings*.
- Effective *application of keyphrase embeddings for extraction and ranking of keywords*, producing state-of-the-art results.
- *A dataset for evaluating the quality of embedding models*, trained for the purpose of capturing semantic similarities between different types of textual units.
- A new *generic framework* (*Key2Vec*), based on distributional semantics, for *ranked keyword extraction*, and *summarization*.

The rest of the paper is organized as follows. In Section II, we discuss about the background literature and works relevant to ours. We fully describe our methodology in Section III, and present our empirical experiment results in Section IV. We discuss our learnings and applications of our framework in Section V, followed by our conclusion and plans for future work in Section VI

II. Related Work

In this section, we give a brief overview of the previous works that are most relevant to this paper. As we use different word embedding techniques in order to train phrase embeddings, we first present a brief review of the popular neural word and phrase embedding approaches (Section II-A and Section **??**, respectively). Present keyword extraction systems primarily operate by using some heuristics for choosing candidate keywords (Section II-A1) that are further ranked by using supervised (Section II-A2) or unsupervised (Section II-A3) approaches. We point out some of the strategies and whenever necessary, explain where we differ.

A. Neural Word and Phrase Embeddings

One of the dominant trends in Natural Language Processing (NLP) at the moment is the use of word embeddings trained using shallow neural networks. also known as neural word embeddings. These word embeddings are distributional representations of words, whose relative similarities correlate with semantic similarities. The real-valued vector representation of words trained using neural networks have been widely used as representational basis for many downstream tasks like text classification [21], document clustering, part of speech tagging, named entity recognition, sentiment analysis, recommendations, etc.

The term word embedding was originally coined by Bengio et al. [22], in 2003. Initially, word embeddings were just a

[1] www.example.com

by-product of training neural language models. It was only in 2008, that Collobert and Weston [23], showed the effectiveness of using pre-trained word embeddings in various downstream tasks related to natural language processing. But it was after a series of works published by Mikolov et al, [24], [15], [25] along with the availability of *Word2Vec* tool, that word embeddings became mainstream and gained wide popularity.

1) Candidate Identification Approaches for Ranked Keyword Extraction: One of the first steps in any keyword extraction method is to identify a set of candidate phrases or words in an optimal way, which are further ranked for getting the top representative keywords for a text document. Typical heuristics include different ways of processing the text, like allowing phrases and words with certain parts-of-speech tag combinations [26], and allowing n-grams [27] that satisfy certain statistical criteria. External corpora have also been used for filtering out unimportant phrases and n-grams.

Researchers have devised a plethora of methods for distinguishing between good and bad (or better and worse) keyphrase candidates. The simplest rely solely on frequency statistics, such as TF*IDF or BM25, to score candidates, assuming that a documents keyphrases tend to be relatively frequent within the document as compared to an external reference corpus. Unfortunately, their performance is mediocre; researchers have demonstrated that the best keyphrases arent necessarily the most frequent within a document. (For a statistical analysis of human-generated keyphrases, check out Descriptive Keyphrases for Text Visualization.) A next attempt might score candidates using multiple statistical features combined in an ad hoc or heuristic manner, but this approach only goes so far. More sophisticated methods apply machine learning to the problem. They fall into two broad categories.

2) Supervised Approaches for Ranked Keyword Extraction: Supervised machine learning methods use training data to infer a function that maps a set of input variables called features to some desired (and known) output value; ideally, this function can correctly predict the (unknown) output values of new examples based on their features alone. The two primary developments in supervised approaches to automatic keyphrase extraction deal with task reformulation and feature design.

Early implementations recast the problem of extracting keyphrases from a document as a binary classification problem, in which some fraction of candidates are classified as keyphrases and the rest as non-keyphrases. This is a well-understood problem, and there are many methods to solve it: Naive Bayes, decision trees, and support vector machines, among others. However, this reformulation of the task is conceptually problematic; humans dont judge keyphrases independently of one another, instead they judge certain phrases as more key than others in a intrinsically relative sense. As such, more recently the problem has been reformulated as a ranking problem, in which a function is trained to rank candidates pairwise according to degree of keyness. The best candidates rise to the top, and the top N are taken to be the documents keyphrases.

The second line of research into supervised approaches has explored a wide variety of features used to discriminate between keyphrases and non-keyphrases. The most common are the aforementioned frequency statistics, along with a grab-bag of other statistical features: phrase length (number of constituent words), phrase position (normalized position within a document of first and/or last occurrence therein), and supervised keyphraseness (number of times a keyphrase appears as such in the training data). Some models take advantage of a documents structural features titles, abstracts, intros and conclusions, metadata, and so on because a candidate is more likely to be a keyphrase if it appears in notable sections. Others are external resource-based features: Wikipedia-based keyphraseness assumes that keyphrases are more likely to appear as Wiki article links and/or titles, while phrase commonness compares a candidates frequency in a document with respect to its frequency in an external corpus. The list of possible features goes on and on.

A well-known implementation of the binary classification method, KEA (as published in Practical Automatic Keyphrase Extraction), used TF*IDF and position of first occurrence (while filtering on phrase length) to identify keyphrases. In A Ranking Approach to Keyphrase Extraction, researchers used a Linear Ranking SVM to rank candidate keyphrases with much success (but failed to give their algorithm a catchy name).

Supervised approaches have generally achieved better performance than unsupervised approaches; however, good training data is hard to find (although heres a decent place to start), and the danger of training a model that doesnt generalize to unseen examples is something to always guard against (e.g. through cross-validation).

3) Unsupervised Approaches for Ranked Keyword Extraction: Unsupervised machine learning methods attempt to discover the underlying structure of a dataset without the assistance of already-labeled examples (training data). The canonical unsupervised approach to automatic keyphrase extraction uses a graph-based ranking method, in which the importance of a candidate is determined by its relatedness to other candidates, where relatedness may be measured by two terms frequency of co-occurrence or semantic relatedness. This method assumes that more important candidates are related to a greater number of other candidates, and that more of those related candidates are also considered important; it does not, however, ensure that selected keyphrases cover all major topics, although multiple variations try to compensate for this weakness.

Essentially, a document is represented as a network whose nodes are candidate keyphrases (typically only key words) and whose edges (optionally weighted by the degree of relatedness) connect related candidates. Then, a graph-based ranking algorithm, such as Googles famous PageRank, is run over the network, and the highest-scoring terms are taken to be the documents keyphrases.

The most famous instantiation of this approach is TextRank; a variation that attempts to ensure good topic coverage is DivRank. For a more extensive breakdown, see Conundrums in Unsupervised Keyphrase Extraction, which includes an example of a topic-based clustering method, the other main class of unsupervised keyphrase extraction algorithms (which Im not going to delve into).

Unsupervised approaches have at least one notable strength: No training data required! In an age of massive but unlabled datasets, this can be a huge advantage over

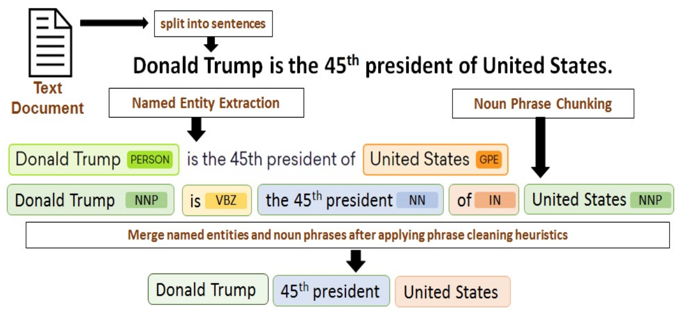

Fig. 1: Text processing pipeline

other approaches. As for disadvantages, unsupervised methods make assumptions that dont necessarily hold across different domains, and up until recently, their performance has been inferior to supervised methods. Which brings me to the next section.

III. Methodology

A. Text Processing

B. Training Phrase Embeddings

1) Skipgram:

2) Continuous Bag of Words:

C. Candidate Identification

D. Candidate Scoring and Ranking

IV. Experiments

A. Embedding Model Selection

For implementing *Key2Vec*, a good quality keyphrase embedding model is needed that can be used for constructing semantically aware representations of phrases, sentences and textual content that forms as a basis for calculating similarities. As explained in Section III-D, we are mainly interested in capturing three kinds of similarities *phrase-phrase*, *sentence-sentence* and *phrase-sentence*, respectively. Although, we don't use phrase-phrase similarity, yet it is fundamental towards understanding the quality of the trained models and also acts as a building block for the other types of similarity calculations. In order to evaluate the models for the above criterias, we needed an evaluation dataset. Since, we could not find an appropriate one, we created three datasets from an exsiting dataset[2], originally developed for evaluating document representations that capture document similarities [18].

The first dataset consists of 106 triplets, which is a subset of manually curated triplets provided in the original dataset. It consists of three phrases in a triplet for evaluating *phrase-phrase* similarity, where the first phrase is semantically closer to the second phrase than it is closer to the third phrase. For example, *Deep learning* is closer to *Machine learning* than *Computer network* or *September* is closer to *October* than *June*. The second dataset consists of 6247 triplets, which is also a subset of automatically generated triplets shared in the original dataset. The triplets are phrases that are titles of Wikipedia articles and are also used for evaluating *phrase-phrase* similarity. The first phrase is supposed to be closer to the second than the third, on the basis, that the first and second phrases are titles of articles belonging to the same category in Wikipedia, unlike the third phrase, which is the title of an article from Wikipedia that belongs to a different category. Contrary to the original dataset that uses the full content of the Wikipedia articles mapped to these triplets for evaluating document similarities, we only use the title phrases.

The third dataset consists of 6,353 triplets that we derive from the first two datasets and original articles automatically collected from Wikipedia. The triplets are *phrase-sentence* combinations intended for evaluating *phrase-sentence* and *sentence-sentence* similarities. We combine the triplets from first two datasets and collect all the Wikipedia articles mapped to them using a crawler. Each part of the triplet consists of a combination of a phrase, which is the title of a Wikipedia article, and the first sentence of the article that mentions that phrase. The first phrase is supposed to be semantically more similar to the sentence associated with it than the sentence associated with the second or the third phrase. For example, *Deep learning* is closer to *"Deep learning (also known as deep structured learning, hierarchical learning or deep machine learning) is a class of machine learning algorithms that: use a cascade of many layers of nonlinear processing units for feature extraction and transformation"*, than *"A computer network or data network is a telecommunications network which allows nodes to share resources"*, which is a sentence about *computer network*. Also, the first sentence is closer to the second sentence (*Machine learning is the subfield of computer science that, according to Arthur Samuel in 1959, gives "computers the ability to learn without being explicitly programmed."*) than the third sentence.

Additionally, we automatically collect full content of 17,326 Wikipedia articles mapped to the titles of the triplets provided in the original dataset, which we use for training different configurations of keyphrase embedding models that are further used for carrying out the similarity evaluation tasks. The full articles are also used for showing the effectiveness of keyphrase embeddings in capturing semantic similarities between text documents (Section V). The entire evaluation dataset is publicly shared[3].

We process the text of the Wikipedia articles as explained in Section III-A and prepare the dataset for training keyphrase embeddings. In order to select the model configurations that would best capture the underlying similarities between different textual units, we train keyphrase embedding models using *skipgram* and *continuous bag of words* schemes as implemented in *Word2Vec*[4] and *Fasttext*[5] toolkits. The vocabulary size of all our models is 3,000,664 unique keyphrases. In this work, we use *negative sampling* for all the schemes, with the number of negative samples fixed to 5. We also fix the size of the *context window* to 5 and number of *epochs* to 10. For producing the vector representations of larger blocks of text, like sentences, we average out the vectors of individual keyphrases extracted from it. The accuracies of the trained

[2]http://cs.stanford.edu/ quocle/triplets-data.tar.gz

[3]www.example.com

[4]https://radimrehurek.com/gensim/models/word2vec.html

[5]https://github.com/facebookresearch/fastText

models for different dimensions (10 - 1000) on three different similarity tasks *phrase-phrase*, *sentence-sentence* and *phrase-sentence* are reported in Tables II, III and IV, respectively.

TABLE II: Accuracies of keyphrase embedding models for phrase-phrase similarity task.

Model	**Dataset**	**10**	**25**	**50**	**75**	**100**	**500**	**1000**
Word2Vec Skipgram	manual triples	72.65%	84.60%	83.52%	83.52%	87.86%	87.86%	84.60%
	auto triples	62.48%	64.98%	64.89%	64.60%	64.98%	66.29%	64.77%
Word2Vec CBOW	*manual triples*	65.04%	80.26%	83.52%	81.34%	80.26%	81.34%	79.17%
	auto triples	60.70%	62.10%	60.11%	61.50%	61.42%	61.08%	61.50%
Fasttext Skipgram	*manual triples*	79.17%	88.95%	92.17%	94.39%	90.04%	86.95%	90.04%
	auto triples	64.85%	67.05%	67.65%	67.18%	68.28%	64.12%	67.90%
Fasttext CBOW	*manual triples*	73.73%	82.43%	88.95%	85.69%	90.04%	86.95%	83.52%
	auto triples	62.14%	65.32%	64.89%	65.19%	64.68%	64.12%	63.41%

TABLE III: Accuracies of keyphrase embedding models for sentence-sentence similarity tasks.

Model	**10**	**25**	**50**	**75**	**100**	**500**	**1000**
Word2Vec Skipgram	63.00%	65.63%	66.56%	65.74%	66.36%	66.25%	65.98%
Word2Vec CBOW	60.52%	62.69%	63.78%	63.93%	63.85%	64.03%	63.79%
Fasttext Skipgram	65.59%	69.05%	70.04%	70.51%	70.94%	70.95%	71.03%
Fasttext CBOW	63.65%	66.67%	67.79%	67.92%	67.77%	68.01%	67.40%

TABLE IV: Accuracies of keyphrase embedding models for phrase-sentence similarity task.

Model	**10**	**25**	**50**	**75**	**100**	**500**	**1000**
Word2Vec Skipgram	73.07%	76.40%	76.75%	76.98%	76.96%	76.75%	76.58%
Word2Vec CBOW	61.27%	64.67%	66.37%	67.37%	68.58%	68.44%	69.11%
Fasttext Skipgram	82.89%	90.08%	93.25%	94.36%	94.71%	96.18%	96.27%
Fasttext CBOW	78.05%	85.89%	89.73%	91.01%	92.22%	93.99%	93.92%

The models that we train in this section is not used directly for the downstream process of ranked keyword extraction. We perform these training and evaluations in order to narrow down to an optimal set of parameter configurations and scheme for training the main keyphrase embedding model that we use for implementing *Key2Vec*. We are interested to try out many other tools and configurations and study the effects on the quality of the keyphrase embeddings, as a part of our future work. We believe that the dataset developed and shared in this paper would allow us and the community at large for carrying out such studies. From the table we can clearly see that the models trained using *Fasttext* performs better than the models trained using *Word2Vec* on all the three tasks. After analyzing the accuracies of the models, we decided to use 100 dimensional keyphrase embeddings trained using *skipgram* method and *negative sampling*, as implemented in the *Fasttext* toolkit. In the future sections these configurations should be assumed for the underlying keyphrase embedding model for *Key2Vec*.

B. Training Keyphrase Embedding Model for Key2Vec

TABLE V: Top 5 similar keyphrases to a given keyphrase as produced by the keyphrase embedding model trained on the arxiv dataset using Fasttext (SkipGram).

Phrase	**Top 5 Similar Phrases**
convolutional_neural_network	cnn, feature_representations, deep_convolutional_neural_network, deep_neural_network, scene_recognition
dark_matter	dm, dark_matter_particle, non-baryonic_dark_matter, dark_energy, self-interacting_dark_matter
natural_language_processing	nlp, language_processing, machine_translation, named_entity_recognition, sense_disambiguation
rnn	blstm, long_short-term_memory, lstms, handwritten_documents, recurrent_neural_network, lstm
svm	support_vector_machine, support_vector_machines, random_forest, svms, naive_bayes

Since this work deals with the domain of scientific articles we train our keyphrase embedding model to be used for implementing *Key2Vec* on a collection of more than million scientific documents. We collect 1,147,000 scientific abstracts related to different areas from arxiv.org[6]. For collecting data we use the API provided by arxiv.org that allows bulk access of the articles uploaded to their portal. A distribution of scientific abstracts from different topics present in our dataset is shown in Fig 2. We also add the scientific documents present in the benchmark datasets (Sections IV-C2 and IV-C3). After processing the text of 1,149,244 scientific documents as mentioned in Section III-A, we train our model using the configurations chosen in the previous section. Table V. shows top 5 similar keyphrases for five different keyphrases as produced by the trained keyphrase embedding model. We use this model as the underlying keyphrase embedding model for implementing *Key2Vec*. Next, we evaluate the performance of *Key2Vec* on two different benchmark datasets and compare the results against some state-of-the-art systems known to perform well on these datasets.

C. Evaluating Ranked Keyword Extraction

We evaluate *Key2Vec* by comparing it with the state-of-the-art systems, on two datasets: *Inspec* dataset of ACM abstracts [28], representing short documents, and *SemEval 2010* Task 5 dataset [8], representing longer documents. We don't implement any existing system for comparison, as we have found from our experience that it is often hard to get the best results by replicating other systems when the original implementation is not present. We directly take the best results for the benchmark datasets as reported in the literature and evaluate our system using the same metrics. In this section, we explain the evaluation metrics used, and present statistics

[6]http://arxiv.org

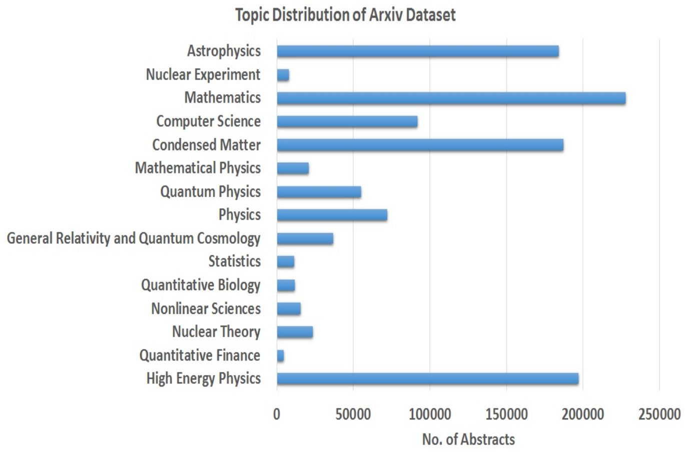

Fig. 2: Frequency distribution of topics in the arxiv dataset.

and results for each of the benchmark datasets. Both the datasets are publicly available[7]. The datasets have training, development and testing data. We only use the test data for all our evaluations as used by all the other systems. Some general statistics for test data provided in both the corpus is presented in Table VI.

TABLE VI: General Statistics for Inspec and SemEval 2010 benchmark datasets.

Corpus Statistics	**Inspec**	**SemEval 2010**
Type	Abstract	Full Article
No. of Documents	500	100
Avg No. of Unigram Tokens [29]	136.3	5179.6
Total No. of Annotated Keywords		3003
Avg No. of Annotated Keywords		30.03
Total No. of Candidates for Key2Vec	6100	47159
Avg No. of Candidates for Key2Vec	12.2	471.59
Total No. of Matches	3562	958
Accuracy	72.50%	31.90%

1) Evaluation Metrics: The ranked keywords are evaluated using exact match evaluation metric as used in SemEval 2010 Task 5 [8]. We match the keywords in the annotated documents in the benchmark datasets with those generated by *Key2Vec*, and calculate micro-averaged precision, recall and F-score ($\beta = 1$) as given in equations 1, 2 and 3, respectively. In the evaluation, we check the performance over the top 5, 10 and 15 candidates returned by *Key2Vec*. While comparing the performance of *Key2Vec* with other systems for the Inspec dataset, we only use F1@10 as the data for other measures are not available. In case of SemEval 2010 dataset, we directly compare our system's performance with the best system in the competition. Next, we present the evaluation results for both the benchmark datasets.

$$precision = \frac{correctly\ matched\ keywords}{total\ no.\ of\ extracted\ keywords} \tag{1}$$

$$recall = \frac{correctly\ matched\ keywords}{total\ no.\ of\ assigned\ keywords} \tag{2}$$

[7]https://github.com/snkim/AutomaticKeyphraseExtraction

$$F1 = \frac{2 \times precision \times recall}{precision + recall} \tag{3}$$

2) Inspec : The Inspec dataset [28] is composed of 2000 abstracts of scientific articles divided into sets of 1000, 500, and 500, as training, validation and test datasets respectively. Each document has two lists of keywords assigned by humans - *controlled*, which are assigned by the authors, and *uncontrolled*, which are freely assigned by the readers. The controlled keywords are mostly abstractive, whereas the uncontrolled ones are mostly extractive [20].

TABLE VII: My caption

Inspec Uncontrolled	**Key2Vec (actual match)**	**Key2Vec (fuzzy match)**	**SGRank**	**TFIDF**
Avg Precision@5			NA	
Avg Recall@5			NA	
Avg F1@5			29.16%	
Avg Precision@10			NA	
Avg Recall@10			NA	
Avg F1@10			33.95%	
Avg Precision@15			NA	
Avg Recall@15			NA	
Avg F1@15			33.66%	

TABLE VIII: My caption

Inspec Controlled + Uncontrolled	**Key2Vec (actual match)**	**Key2Vec (fuzzy match)**	**SGRank**	**TFIDF**
Avg Precision@5				
Avg Recall@5				
Avg F1@5				
Avg Precision@10				
Avg Recall@10				
Avg F1@10				
Avg Precision@15				
Avg Recall@15				
Avg F1@15				

3) SemEval 2010 : The Semeval dataset was used in the Semeval 2010 keyphrase extraction shared task (Kim et al., 2010). To the best of our knowledge this shared task is the largest recent comparison of keyphrase extraction algorithms and an algorithms performance on this dataset is a relatively good indication of where it stands compared to others in the field. The Semeval dataset consists of 284 full length ACM articles divided into a test set of size 100, training set of size 144 and trial set of size 40 which we used as the development set for parameter tuning. Each article has two sets of human assigned keyphrases: the author-assigned and reader-assigned ones. The gold standard used in our experiments is the combined set of author and reader assigned keyphrases which is the same as was done in the Semeval shared task. The table below provides a statistical overview of this datasets documents.

Table showing Semeval 2010 dataset statistics

We have compared our algorithm with KPMiner and TextRank using only the 100 documents in the test set. The following diagram shows the average precision and recall achieved by each algorithm. As was done in the Semeval task,

TABLE IX: My caption

SemEval 2010 Combined	Key2Vec (actual match)	Key2Vec (fuzzy match)	HUMB	TFIDF
Avg Precision@5	41%		39.00%	
Avg Recall@5	14.37%		13.30%	
Avg F1@5	21.28%		19.80%	
Avg Precision@10	35.29%		32.00%	
Avg Recall@10	24.67%		21.80%	
Avg F1@10	29.04%		26.00%	
Avg Precision@15	34.39%		27.20%	
Avg Recall@15	32.48%		27.80%	
Avg F1@15	33.41%		27.50%	

TABLE X: My caption

SemEval 2010 Combined	Key2Vec (actual match)	Key2Vec (fuzzy match)	HUMB	TFIDF
Avg Precision@5	38.20%		30.40%	
Avg Recall@5	15.97%		12.60%	
Avg F1@5	22.52%		17.80%	
Avg Precision@10	31.68%		24.80%	
Avg Recall@10	26.25%		20.60%	
Avg F1@10	28.71%		22.50%	
Avg Precision@15	34.92%		12.10%	
Avg Recall@15	34.39%		47.00%	
Avg F1@15	34.65%		19.30%	

comparisons are done between once stemmed human assigned keyphrases and ranked candidates returned by each algorithm.

TABLE XI: My caption

SemEval 2010 Combined (Raw Text)	title	title + abstract	title + introduction	title + abstract + introduction+ conclusion	full article
Avg Precision@5	41.40%	40.60%	41.20%	41%	41%
Avg Recall@5	14.56%	14.24%	14.48%	14.40%	14.37%
Avg F1@5	21.54%	21.09%	21.43%	21.32%	21.28%
Avg Precision@10	35.29%	35.09%	35.09%	34.99%	35.29%
Avg Recall@10	24.66%	24.51%	24.51%	24.45%	24.67%
Avg F1@10	29.04%	28.86%	28.86%	28.79%	29.04%
Avg Precision@15	34.40%	34.53%	34.68%	34.53%	34.39%
Avg Recall@15	32.45%	32.60%	32.75%	32.60%	32.48%
Avg F1@15	33.40%	33.54%	33.69%	33.54%	33.41%

TABLE XII: My caption

SemEval 2010 Combined (Preprocessed)	title	title + abstract	title + introduction	title + abstract + introduction+ conclusion	full article
Avg Precision@5	43.80%	43.80%	43.80%	44%	43.80%
Avg Recall@5	15.08%	15.08%	15.08%	15.18%	15.08%
Avg F1@5	22.44%	22.44%	22.44%	22.57%	22.44%
Avg Precision@10	37.39%	37.49%	37.19%	36.99%	37.19%
Avg Recall@10	25.86%	25.95%	25.76%	25.59%	25.77%
Avg F1@10	30.57%	30.67%	30.44%	30.25%	30.44%
Avg Precision@15	37.96%	37.61%	37.61%	37.69%	37.77%
Avg Recall@15	35.62%	35.34%	35.35%	35.37%	35.50%
Avg F1@15	36.75%	36.44%	36.44%	36.49%	36.60%

V. Discussion

VI. Future Work and Conclusion

References

[1] C. Gutwin, G. Paynter, I. Witten, C. Nevill-Manning, and E. Frank, "Improving browsing in digital libraries with keyphrase indexes," *Decision Support Systems*, vol. 27, no. 1, pp. 81–104, 1999.

[2] M. Litvak and M. Last, "Graph-based keyword extraction for single-document summarization," in *Proceedings of the workshop on Multi-source Multilingual Information Extraction and Summarization*. Association for Computational Linguistics, 2008, pp. 17–24.

[3] K. M. Hammouda, D. N. Matute, and M. S. Kamel, "Corephrase: Keyphrase extraction for document clustering," in *International Workshop on Machine Learning and Data Mining in Pattern Recognition*. Springer, 2005, pp. 265–274.

[4] J. Gulla, H. Borch, and J. Ingvaldsen, "Ontology learning for search applications," *On the Move to Meaningful Internet Systems 2007: CoopIS, DOA, ODBASE, GADA, and IS*, pp. 1050–1062, 2007.

[5] A. Hulth and B. B. Megyesi, "A study on automatically extracted keywords in text categorization," in *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2006, pp. 537–544.

[6] W. Wu, B. Zhang, and M. Ostendorf, "Automatic generation of personalized annotation tags for twitter users," in *Human language technologies: The 2010 annual conference of the North American chapter of the association for computational linguistics*. Association for Computational Linguistics, 2010, pp. 689–692.

[7] C. Collins, F. B. Viegas, and M. Wattenberg, "Parallel tag clouds to explore and analyze faceted text corpora," in *Visual Analytics Science and Technology, 2009. VAST 2009. IEEE Symposium on*. IEEE, 2009, pp. 91–98.

[8] S. N. Kim, O. Medelyan, M.-Y. Kan, and T. Baldwin, "Semeval-2010 task 5: Automatic keyphrase extraction from scientific articles," in *Proceedings of the 5th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, 2010, pp. 21–26.

[9] I. Augenstein, M. Das, S. Riedel, L. Vikraman, and A. McCallum, "Semeval 2017 task 10: Scienceie-extracting keyphrases and relations from scientific publications," *arXiv preprint arXiv:1704.02853*, 2017.

[10] Z. Liu, W. Huang, Y. Zheng, and M. Sun, "Automatic keyphrase extraction via topic decomposition," in *Proceedings of the 2010 conference on empirical methods in natural language processing*. Association for Computational Linguistics, 2010, pp. 366–376.

[11] K. S. Hasan and V. Ng, "Automatic keyphrase extraction: A survey of the state of the art." in *ACL (1)*, 2014, pp. 1262–1273.

[12] ——, "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, 2010, pp. 365–373.

[13] M. Baroni, G. Dinu, and G. Kruszewski, "Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors." in *ACL (1)*, 2014, pp. 238–247.

[14] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research*, vol. 12, no. Aug, pp. 2493–2537, 2011.

[15] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[16] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation." in *EMNLP*, vol. 14, 2014, pp. 1532–1543.

[17] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *arXiv preprint arXiv:1607.04606*, 2016.

[18] A. M. Dai, C. Olah, and Q. V. Le, "Document embedding with paragraph vectors," *arXiv preprint arXiv:1507.07998*, 2015.

[19] Z. Liu, P. Li, Y. Zheng, and M. Sun, "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, 2009, pp. 257–266.

[20] R. Wang, W. Liu, and C. McDonald, "Using word embeddings to enhance keyword identification for scientific publications," in *Australasian Database Conference*. Springer, 2015, pp. 257–268.

[21] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," *arXiv preprint arXiv:1607.01759*, 2016.

[22] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *Journal of machine learning research*, vol. 3, no. Feb, pp. 1137–1155, 2003.

[23] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 160–167.

[24] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[25] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, 2014, pp. 1188–1196.

[26] R. Mihalcea and P. Tarau, "Textrank: Bringing order into texts." Association for Computational Linguistics, 2004.

[27] S. Danesh, T. Sumner, and J. H. Martin, "Sgrank: Combining statistical and graphical methods to improve the state of the art in unsupervised keyphrase extraction," *Lexical and Computational Semantics (* SEM 2015)*, p. 117, 2015.

[28] A. Hulth, "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, 2003, pp. 216–223.

[29] A. Bougouin, F. Boudin, and B. Daille, "Topicrank: Graph-based topic ranking for keyphrase extraction," in *International Joint Conference on Natural Language Processing (IJCNLP)*, 2013, pp. 543–551.