MultPAX: Keyphrase Extraction Using Language Models and Knowledge Graphs

Hamada M. Zahera(✉), Daniel Vollmers, Mohamed Ahmed Sherif, and Axel-Cyrille Ngonga Ngomo

DICE Group, Department of Computer Science, Paderborn University, Paderborn, Germany
{hamada.zahera,daniel.vollmers,mohamed.sherif, axel.ngonga}@uni-paderborn.de

Abstract. Keyphrase extraction aims to identify a small set of phrases that best describe the content of text. The automatic generation of keyphrases has become essential for many natural language applications such as text categorization, indexing, and summarization. In this paper, we propose MultPAX, a multitask framework for extracting *present* and *absent* keyphrases using pre-trained language models and knowledge graphs. In particular, our framework contains three components: first, MultPAX identifies present keyphrases from an input document. Then, MultPAX links with external knowledge graphs to get more relevant phrases. Finally, MultPAX ranks the extracted phrases based on their semantic relatedness to the input document and return top-k phrases as a final output. We conducted several experiments on four benchmark datasets to evaluate the performance of MultPAX against different state-of-the-art baselines. The evaluation results demonstrate that our approach significantly outperforms the state-of-the-art baselines, with a significance t-test $p < 0.041$. Our source code and datasets are public available at https://github.com/dice-group/MultPAX.

Keywords: Present keyphrase extraction · Absent keyphrase generation · Knowledge graph · Pre-trained language models

1 Introduction

Keyphrase extraction is the process of extracting a small set of phrases that best describe a document. This process has been leveraged for several downstream applications, including text summarizing, organizing, and indexing [16]. In the literature, keyphrase extraction is divided into two sub-tasks: (i) detecting present keyphrases (PKE) that appear in a document, and (ii) generating absent keyphrases (AKG) that do not appear in the original document, but are essential for downstream applications (e.g., text summarization, indexing). Table 1 shows an example of extracting present and absent keyphrases from an input text.

Existing works mostly focus on extracting *present* keyphrases from an input text, including supervised learning (e.g., sequence labelling [22]), and unsupervised learning (e.g., TextRank [17], YAKE [4]). By contrast, generating *absent*

U. Sattler et al. (Eds.): ISWC 2022, LNCS 13489, pp. 303–318, 2022.
https://doi.org/10.1007/978-3-031-19433-7_18

Table 1. Example of present and absent keyphrase extraction from `Inspec` dataset. The predicted present keyphrases are in *italic, and the absent ones are highlighted in gray*

Input Text	*"This paper shows the importance that management plays in the protection of information and in the planning to handle a security breach when a theft of information happens. Recent thefts of information that have hit major companies have caused concern. These thefts were caused by companies' inability to determine risks associated with the protection of their data, and these companies lack of planning to properly manage a security breach when it occurs." quoted from [20]*
Ground-truth Keyphrases	security breach, risk analysis, management issue, theft of information
Predicted Keyphrases	*security breach, theft of information*, security management, security risk, data management

keyphrases (i.e., keyphrases that do not appear in a text) is a challenging task. According to the statistical study by [31], some benchmarking datasets (e.g., Inspec [9]) are missing up to 37.7% of *absent* keyphrases. To cope with this challenge, few studies have been proposed. For example, [15] employed a supervised sequence-to-sequence model with a *copy mechanism*, which allows copying important words directly from a source text, rather than decoding new words. However, this approach requires large-scale labelled data for training the model efficiently. In addition, the copy mechanism only generates one word at each time step and does not consider any dependencies between the selected words [33]. Another line of work aims to utilize external sources of knowledge to generate absent keyphrases. For example, [24] constructs a phrase bank consisting of all keyphrases in a text corpus. The authors assumed that absent keyphrases in one document might be found in other relevant documents. However, this approach requires creating a domain-specific phrase bank to generate absent keyphrases.

In this paper, we aim not only to *extract* present keyphrases from an input document, but also to *generate* absent keyphrases that are relevant and do not appear in the document. We reduce the effort required to develop a keyphrase model by employing pre-computed resources. In particular, we use *pre-trained language models* to extract present keyphrases and *knowledge graphs* (KGs) to generate absent keyphrases. Therefore, we propose an unsupervised multitask framework (dubbed MultPAX) with the following pipeline: i) We tokenize an input document into n-grams phrases and embed both (*document and n-gram phrases*) as low-dimensional vectors into one semantic space. Then, we *extract* the top-k phrases that are close to the document's vector as candidates for present keyphrases. ii) We then *link* the extracted present keyphrases to find additional related terms (e.g., synonyms, hypernyms) from external KGs (e.g., DBpedia, BabelNet). For this purpose, we developed a new version of the MAG framework [18], which is optimized for linking keywords and extracting related terms. iii) Finally, we *rank* all keyphrases (i.e., *present* and *absent*) based on their semantic similarity to the input document. The top-k phrases are returned as the final keyphrases output.

Additionally, we propose an improved metric for evaluating predicted keyphrases based on their *semantic-matching* with ground-truth keyphrases. Existing studies [13,15,32] consider *precision, recall*, and F_1 based on the *exact-matching* between predicted and ground-truth keyphrases, which yields reasonable evaluation for present keyphrases that appear in text. However, in evaluating absent keyphrases, the exact-matching demonstrated an inefficient assessment of words that are semantically similar but are literally different [21]. As an example, assume *"Cryptocurrency"* as a ground-truth keyphrase, and a keyphrase model was able to generated *"Bitcoin"* as a predicted keyphrase. In this case, the exact-matching metric ignores the semantic relatedness between both words and considers them completely unrelated. By means of words embeddings, these words are similar and adjacent to each other in the embedding space. In this regard, we propose using an embedding-based F_1-score to evaluate keyphrases extraction in a more accurate semantic way.

To evaluate the performance of MultPAX, we conducted several experiments on four benchmark datasets, where we study the performance of our system against different approaches. The evaluation results show that our approach significantly outperforms the state-of-the-art baselines with a significance t-test $p < 0.041$ and F_1-score up to 0.535. The main contributions in this paper can be summarized as follows:

- We propose an *unsupervised* multitask framework that not only extracts present keyphrases, but also generate absent ones.
- To the best of our knowledge, our approach is the first attempt that leverages existing *knowledge graphs* for keyphrases generation without the need to create keyphrases vocabularies or phrase banks.
- We introduce an *embedding-based F_1* evaluation that considers semantic similarity between generated and ground-truth keyphrases rather than the existing *exact-matching*.
- We carried out several experiments on four benchmark datasets. The evaluation results showed that our approach proved to be more accurate compared with state-of-the-art baselines.

2 Related Work

In this section, we give an overview of the related approaches in *unsupervised keyphrase extraction* and *absent keyphrase generation*.

2.1 Unsupervised Keyphrase Extraction

Several approaches have recently been developed for extracting keyphrases in unsupervised setting without the need for annotated data. For example, statistical approaches such as TF-IDF and YAKE [4] compute statistical features (e.g., word frequencies and co-occurrences) to find important words as candidates for present keyphrases. Moreover, graph-based approaches like TextRank [17] construct a graph representation of text, where words are represented as nodes and their co-occurrences as edges. Thereafter, a node ranking algorithm (e.g., PageRank) is used to sort words, and return top-k words as candidate keyphrases. [3] proposed TopicRank, a graph-based approach similar to TextRank. In the first step, candidate phrases are clustered into topics and then ranked based on with their importance in the document.

Recent studies have demonstrated that embedding-based models can achieve significant results in extracting keyphrases. For example, EmbedRank [2] approach uses part-of-speech tags to extracts potential keyphrases from an input document. Then, EmbedRank uses a pre-trained embedding model to represent both phrases and an input document as low-dimensional vectors. Candidate keyphrases are then ranked based on their Cosine similarity scores to the document's embedding vector. Although pre-trained language models have shown promising performances for extracting present keyphrases, they have failed to generate absent keyphrases from their lexical corpus. Furthermore, [13] pointed out that embedding-based models ignore local information in a document. Accordingly, they developed a jointly-trained model to incorporate global and local context of a document. In the global view, their approach represented candidate keyphrases and the input document as low-dimensional vectors into one semantic space. After that, the similarity between each candidate keyphrase and the document is computed. In terms of the local context, the authors built a graph structure based on the document context, where nodes represent phrases and edges represent similarities between them. Finally, the output keyphrases are ranked based on this global and local information.

2.2 Absent Keyphrase Extraction

Many previous approaches have relied on *sequence-to-sequence* models—with encoder-decoder architecture—to generate absent keyphrases [6]. By doing so, sequence-to-sequence models are able to decode not only keyphrases that appear in source text, but also those that may be absent, i.e., the ones that are not explicitly mentioned in the text. However, additional mechanisms need to be integrated to improve the generation of absent keyphrases. For example, [31] applied a *Graph Neural Network* (GNN) to capture knowledge from related references in scholarly publications. A *neural topic model* is employed in [28] to expand the context of the decoding component to generate more absent keyphrases.

It is noteworthy that [32] achieved significant results in extracting keyphrases by dividing this task into two sub-tasks: present keyphrase extraction and absent keyphrase generation. Furthermore, the authors proposed a multitask approach to *select*, *guide*, and *generate* keyphrases. In the *select* module, the authors used a BiLSTM to predict whether a sentence has a keyphrase or not. Then, a *guider* network is employed to utilize the attention information and memorize the predictions of the *selector*. Finally, this information is fed to a *generator* network to generate absent keyphrases by selecting words from both the source text and a predefined vocabulary. In addition to these fully-supervised approaches, there are also some unsupervised methods that achieved promising results in generating keyphrases without the need for labelled data. [24] observed that many keyphrases absent from an input document appeared in other related documents. Therefore, they constructed a *phrase bank* of all keyphrases in a corpus. Then, they identified present keyphrases in relevant documents as candidates for absent keyphrases for the input document. In addition, they employed present keyphrases as *sliver labels* to train a sequence-to-sequence model. Finally, all keyphrases (both present and absent) were ranked based on their lexical and semantic similarity to an input document.

3 Our Approach

In this section, we present our approach for extracting *present* and *absent* keyphrases. Figure 1 depicts the architecture of our MultPAX framework, including three components: i) *present keyphrase extraction* (PKE), ii) *absent keyphrase generation* (AKG), and iii) *Keyphrases Semantic Matching*.

3.1 Problem Formulation

Let $\mathcal{D}$ be an input document with $|S|$ sentences; each sentence $s \in S$ is a sequence of $|s|$ tokens $\mathcal{T} = \{t_1, t_2, \cdots, t_{|s|}\}$. Our goal is to build a keyphrase model that not only extracts *present keyphrases* $\mathcal{Y}^p = \{y^p_1, y^p_2, \cdots, y^p_{|\mathcal{Y}^p|}\}$ but also generates *absent keyphrases* $\mathcal{Y}^a = \{y^a_1, y^a_2, \cdots, y^a_{|\mathcal{Y}^a|}\}$ that are relevant to $\mathcal{D}$ by leveraging knowledge graphs such as DBpedia [1] and BabelNet [19].

Following previous works [8,22], we divide the task of keyphrase extraction into two sub-tasks: *Present Keyphrase Extraction* (PKE) and *Absent Keyphrase Generation* (AKG). Furthermore, we define the computation of final keyphrases as a *Semantic Matching* task. First, we consider PKE as a *ranking* problem, where candidate phrases are extracted and then ranked based on their similarities to the input document (see Sect. 3.2). Second, we formulate AKE as a *linking* problem to infer relevant information from external knowledge graphs. For this task, we employ an unsupervised *entity-linker* [23] that maps a present keyphrase ($\mathcal{Y}^p$) to its corresponding entity in a knowledge graph (i.e., DBpedia, BabelNet) and then get relevant terms (e.g., from `dct:subject`, `gold:hypernym` properties) as candidates for absent keyphrases. Finally, all keyphrases $\mathcal{Y}^p \cup \mathcal{Y}^a$ are ranked based on their similarities to $\mathcal{D}$, the top-k keyphrases are returned as the final output.

3.2 Present Keyphrase Extraction (PKE)

We employ the Bert language model [7] to extract present keyphrases based on their semantic similarity to a document. The main steps are as follows: (1) We tokenize an input document $\mathcal{D}$ into *n-gram phrases* and annotate each token

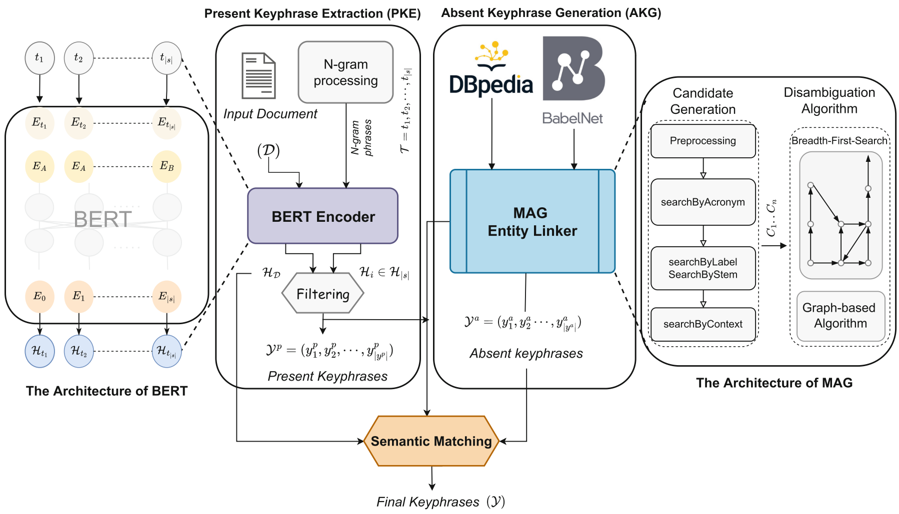

Fig. 1. The architecture of MultPAX framework with three components: *present keyphrase extraction, absent keyphrase generation* and *semantic matching*

with part-of-speech tags (e.g., ADJ: adjectives, NOUN: nouns, VERB: verbs). (2) Then, we remove stop words and keep noun phrases that consist of zero or more adjectives followed by one or multiple nouns [27]. (3) We employ the pre-trained language model (`BERT-Encoder`) to encode candidate keyphrases as low-dimensional vectors together with the input document into one embedding space.

A special preprocessing is applied to the input text of the `BERT-Encoder` as follows: a `[CLS]` token is added at the beginning of each sentence, which is then used to obtain the contextualized embeddings vector of a sentence. An additional token `[SEP]` is inserted to mark the end of a sentence. Afterward, the input is tokenized by *WordPiece* tokenizer [25]; each token t_i is associated with three types of embeddings: *token embeddings* (E_{t_i}) which represents the vocabulary index of each token, *segmentation embeddings* that distinguishes between input sentences (E_A or E_B), and *position embeddings* (E_i) to indicate the position of each word. The output of the `BERT-Encoder` is the sentence's representation matrix $\mathcal{H} = [h_0, h_1, \cdots h_{|s|}]$, where h_i denotes the embedding vector of token t_i. Formally, the embedding vector of a sentence s_j is

$$\mathcal{H}_j = \texttt{BERT-Encoder}(\{t_1, t_2, \cdots t_{|s|}\}). \tag{1}$$

Pooling is an essential operation for creating sentence and document embeddings [5]. It is commonly used to aggregate (e.g., mean, max) multiple representations (e.g., sentences) into one embedding vector. To obtain the document's embeddings $\mathcal{H}_\mathcal{D}$, we employ a `MaxPooling` layer on top of all sentences' representations. Formally,

$$\mathcal{H}_\mathcal{D} = \texttt{MaxPooling}(\{\mathcal{H}_1, \mathcal{H}_2, \cdots \mathcal{H}_{|S|}\}). \tag{2}$$

Finally, we use Cosine distance to compute similarities between the embedding vectors of candidate keyphrases $\mathcal{H}_i \in H_{|S|}$ and the document embedding $\mathcal{H}_{\mathcal{D}}$. We select the top-$k$ keyphrases as candidates for present keyphrases.

3.3 Absent Keyphrase Generation (AKG)

To obtain absent keyphrases, we first link all present keyphrases $\mathcal{Y}^p$ to a knowledge graph and get additional surface forms (i.e., strings that could be synonyms or alternative names). We consider the DBpedia knowledge graph since it provides surface forms for a wide range of common entities. For *entity linking*, we follow a similar approach to the MAG framework [18].

MAG extracts entity links using two steps: *candidate generation* and *candidate disambiguation*. In the *candidate generation* step, MAG aims to find candidate links $(C_1, \dots, C_n)$ for pre-marked entities in the search index [18]. To this end, MAG uses acronyms and labels in a knowledge graph to map premarked entity spans from the input text to candidate entities. Furthermore, MAG also relies on the Concise Bounded Description (CBD)[1] of the entities in a knowledge graph by comparing the context of the entity spans in the input document and the CBD of an entity in a knowledge graph [18]. We keep this candidate generation step from MAG and apply it to the extracted present keyphrases from the PKE component. In the *candidate disambiguation* step, MAG generates a local graph using a breadth-first-search method for all candidate entities on a knowledge graph. Then, MAG applies the HITS ranking algorithm [11] to jointly rank the candidate links for all entities in the local graph. HITS ranks the nodes in a directed graph based on incoming and outgoing edges. Authorities are seen as nodes, that carry important information, while hubs are nodes, that point to a large amount of authority nodes. So the authority score of a node n is calculated based on the hub score of the nodes, that have a directed edge to the node n, while the hub score of n is calculated based on the authority score of the nodes which are linked by n [11]. Formally, HITS calculates the authority score a_p for the node p as

$$a_p = \sum_{q:(q,p) \in G} h_q. \tag{3}$$

where h_q is the hub-score for the node q, given that a directed edge from node q to node p exists in the graph G. The hub-score h_p for a node p is calculated as

$$h_p = \sum_{q:(q,p) \in G} a_q. \tag{4}$$

where a_q is the authority-score for a node q, which is linked by node p [11]. a_q and h_p are initialized randomly and updated iteratively until convergence.

In contrast to MAG, we not only link present keyphrases, but also extract related terms for each linked keyphrase from a knowledge graph. Furthermore, we extract top-ranked candidates for each entity and n nodes with the highest

[1] https://www.w3.org/Submission/CBD/.

authority scores in the local graph, since their surface forms could be used as candidates for the absent keyphrases. In our approach, we use BabelNet to find *hypernyms* for the present keyphrases, in addition to the surface forms from DBpedia.

3.4 Keyphrases Semantic Matching

In the last component, we aim to identify top-k relevant keyphrases (*present* and *absent*), we set $k = \{5, 10, 20\}$ in our experiments. We regard this task as a semantic textual similarity [14]. To match similarities between a document $\mathcal{D}$ and candidate keyphrases, we embed them into one semantic space using a pre-trained embedding model. Then we employ Cosine distance to find top-k nearest keyphrases ($\mathcal{H}_i$) to the document's vector $\mathcal{H}_D$ and return as final keyphrase predictions. Formally,

$$\mathrm{Cos}(\mathcal{H}_i, \mathcal{H}_\mathcal{D}) = \frac{\mathcal{H}_i \cdot \mathcal{H}_\mathcal{D}}{||\mathcal{H}_i|| \times ||\mathcal{H}_\mathcal{D}||}. \tag{5}$$

where $\mathcal{H}_i$ donates the embedding vector of candidate keyphrase (*present* y_i^p or *absent* y_i^a), and $\mathcal{H}_\mathcal{D}$ represents the embedding vector of the input document.

4 Experiments

We conducted our experiments to answer the following research questions:

Q_1. *How efficient is our approach in extracting present keyphrases compared to the state-of-the-art approaches?*
Q_2. *Are the existing exact-matching metrics (i.e., Precision, Recall and F_1-score) suitable for evaluating absent keyphrases?*
Q_3. *To what extent does each component in our framework contribute to the overall performance?*

Table 2. Statistics about the datasets (#Doc: number of documents, #Test: size of test set, #Avg. KP: average keyphrase per document, #Ratio%: percentage of absent keyphrase per dataset)

Dataset	#Doc	#Test	Avg. KP	Ratio%
Inspec	2k	500	7.65	37.7%
Krapivin	2.3k	460	3.03	15.3%
SemEval2010	144	100	7.15	11.3%
NUS	211	211	2.71	17.8%

4.1 Experimental Setup

Datasets. In our experiments, we used four benchmark datasets of English documents, namely, *Inspec* [9], *SemEval2010* [10], *Krapivin* [12], and *NUS* [26]. Table 2 provides a statistical overview of each dataset, including the total number of documents (#Doc.), the number of documents in the evaluation set (#Test), average keyphrases per document (Avg. KP) and the ratio of absent keyphrases in each dataset (Ratio%).

Baselines. We compared our approach with the following baselines for extracting keyphrases:

- **TextRank** [17] is an unsupervised approach that constructs a graph representation from a document, where nodes represent phrases and their edges are computed based on lexical similarities. Further, TextRank uses the *PageRank* algorithm to extract present keyphrases.
- **YAKE** [4] is a simple unsupervised method that extracts keywords automatically based on statistical features such as words co-occurrence and frequencies.
- **EmbedRank** [2] is an unsupervised method that employs words embeddings to identify relevant words to a document as candidate keyphrases. Furthermore, EmbedRank utilizes the *Maximum Marginal Relevance* algorithm to increase the diversity of the extracted keyphrases.
- **Supervised-CopyRNN** [15] is a supervised baseline that trains a sequence-to-sequence model with a *copy mechanism* on `KP20K` dataset [15]. We used this approach as a baseline for present keyphrases extraction as well as absent keyphrase generation to compare the performance of copy mechanism.
- **AutoKeyGen** [24] is an unsupervised approach that constructs a *phrase bank* by combining keyphrases from all documents into a corpus. Then, AutoKeyGen considers lexical- and semantic-level similarities for selecting top candidate keyphrases (present and absent) for each input document.

Evaluation Metrics. We evaluated our approach using different metrics: *Precision*, *Recall*, and F_1 scores. The *Precision* is computed as the number of correctly-matched keyphrases over all predicted keyphrases.

Given a list of predicted keyphrases $\mathcal{Y} = (y_1, \dots, y_{|\mathcal{Y}|})$, we select the top-$k$ ranked keyphrases $\mathcal{Y}_{:k} = (y_1, \dots, y_{\min(k,|\mathcal{Y}|)})$ and compare with the top-k ranked keyphrases in the ground-truth set. We set $k = \{5, 10\}$ for present keyphrases and $k = \{10, 20\}$ for absent ones in our experiments. Following previous works [24,30], we use the *Porter Stemmer* from the `NLTK library`[2] `v3.7` to compute exact-matching between the top-k predicted ($\mathcal{Y}_{:k}$) and the ground-truth ($\mathcal{Y}^{gold}$) keyphrases. The precision of the top-k predicted keyphrases is defined as

$$P@k = \frac{|\mathcal{Y}_{:k} \cap \mathcal{Y}^{gold}|}{|\mathcal{Y}_{:k}|}. \tag{6}$$

[2] https://www.nltk.org/index.html.

The *Recall* is calculated as how many correctly-matched keyphrases among all ground-truth keyphrases. Formally, the Recall is defined as

$$R@k = \frac{|\mathcal{Y}_{:k} \cap \mathcal{Y}^{gold}|}{|\mathcal{Y}^{gold}_{:k}|}. \tag{7}$$

and the $F_1@k$-score is defined as the harmonic mean of P@k and R@k

$$F_1@k = 2 \times \frac{P@k \times R@k}{P@k + R@k}. \tag{8}$$

Although the *exact-matching* metric has been used widely in the literature [13], there is still a room for improvement regarding the absent keyphrases evaluation based on semantic similarity. Hence, we propose in Sect. 4.3 a semantic-based matching to evaluate the performance of generated absent keyphrases.

Hyperparameters. We performed a grid search to optimize the hyperparameters of our approach. We found the following values yield the best F_1-scores. In the PKE component, we tokenized the input text into phrases of 2–4 grams. Further, we considered the top-10 ranked phrases as candidates for present keyphrases. The full setup of our experiments is available at the GitHub repository.[3] For the baseline methods, the hyperparameters were set according to their original papers. In the MAG framework, we adapted the extraction of common entities to cover a larger set of entity types. In addition, we set the other hyperparameters values with the standard configuration[4] of the MAG framework.

Table 3. Evaluation results of *present* keyphrases prediction on `Inspec`, `SemEval2010`, `Krapivin`, and `NUS` datasets. $F_1@k$-scores are reported based on **exact-matching** between the predicted and ground-truth keyphrases. Best results are reported in bold

Model	Inspec		SemEval2010		Krapivin		NUS	
	$F_1@5$	$F_1@10$	$F_1@5$	$F_1@10$	$F_1@5$	$F_1@10$	$F_1@5$	$F_1@10$
TextRank	0.263	0.279	0.183	0.181	0.148	0.139	0.187	0.195
YAKE	0.027	0.038	0.050	0.242	0.013	0.020	0.013	0.020
EmbedRank	0.295	0.344	0.108	0.145	0.131	0.138	0.103	0.134
Supervised-CopyRNN	0.292	0.336	0.291	**0.296**	0.302	0.252	0.342	0.317
AutoKeyGen	0.303	**0.345**	0.187	0.240	0.171	0.155	0.218	0.233
MultPAX	**0.371**	0.210	**0.449**	0.255	**0.384**	**0.334**	**0.535**	**0.344**

[3] https://github.com/dice-group/MultPAX.

[4] https://github.com/dice-group/AGDISTIS/blob/master/src/main/resources/config/agdistis.properties.

Table 4. *Absent* keyphrases evaluation (in terms of R@10, R@20). All results are reported based on **exact-matching** between the predicted and ground-truth keyphrases, except the last row shows Recall results based on **semantic-matching**

Model	Inspec		SemEval2010		Krapivin		NUS	
	R@10	R@20	R@10	R@20	R@10	R@20	R@10	R@20
Supervised-CopyRNN	0.051	0.068	0.049	0.057	0.116	0.142	0.078	0.10
`AutoKeyGen`-Bank	0.015	0.017	0.007	0.009	0.031	0.041	0.021	0.026
`AutoKeyGen`-Full	0.017	0.021	0.010	0.011	0.033	0.054	0.024	0.032
MULTPAX$_{\text{exact-Matching}}$	0.079	0.080	–	–	–	–	0.017	0.017
MULTPAX$_{\text{semantic-Matching}}$	0.696	0.584	–	–	–	–	0.608	0.669

4.2 Present Keyphrase Evaluation (Q_1)

To answer Q_1, we evaluated our approach (MULTPAX) vs. different baselines in extracting *present* keyphrases. As shown in Table 3, MULTPAX significantly outperforms all baselines by a large margin on most datasets with a significant *t-test* $p < 0.041$. This is due to, MULTPAX employs semantic similarity between candidate keyphrases and an input document using the state-of-the-art pre-trained language model in semantic textual matching [29]. In contrast, CopyRNN [15] and AutoKeyGen [24] used sequence-to-sequence models to encode an input document as a low-dimensional vector and decode it back into a sequence of predicted keyphrases.

On the other hand, we find that YAKE does not perform well in detecting present keyphrases from short texts (e.g., papers' abstracts). Since YAKE relies on statistical features such as words co-occurrence and frequencies, which are efficiently computed only in long texts (e.g., full papers or news). Remarkably, the embedding-based baseline (EmbedRank) achieves comparable results; however, it fails to generate absent keyphrases. In our approach, we extract present keyphrases from text using contextualized embeddings and semantic matching. These findings answer Q_1; by employing pre-trained language models, we can not only efficiently identify present keyphrases from text without labelled data, but we also outperform the state-of-the-art approach (AutoKeyGen).

4.3 Absent Keyphrase Evaluation (Q_2)

We conduct further experiments to evaluate the performance of our approach against two baselines (namely, CopyRNN and AutoKeyGen) in generating absent keyphrases. Following previous work [24], we use the Recall metric (R@10, R@20) based on *exact-matching* for the performance evaluation as shown in Table 4. Since we used the same experimental setup of CopyRNN and AutoKeyGen approaches, we obtained the evaluation results from their papers [15,24].

Regarding Q_2, we can clearly see that all approaches achieve poor performances when considering exact-matching between predicted and ground-truth keyphrases. For example, if two keyphrases are semantically similar, e.g., *"disaster relief organization"* and *"crisis responses institute"*, these keyphrases will not be considered as a *match* using the existing metrics. Hence, we found that such metrics are unsuitable for evaluating absent keyphrases. We propose an improved evaluation metric based on the *semantic-matching*. Formally, let $\mathcal{Y}^a$ be predicted keyphrases; $\mathcal{Y}^{gold}$ is ground-truth keyphrases. We first embed each keyphrase in $\mathcal{Y}^a$ and $\mathcal{Y}^{gold}$. Then, we use Cosine distance to compute similarities between the embedding of each keyphrase in $\mathcal{Y}$ and $\mathcal{Y}^{gold}$. We set a threshold (> 0.5) for similarities scores to consider semantic-matching between $\mathcal{Y}$ and $\mathcal{Y}^{gold}$. The two last rows in Table 4 show the evaluation results of R@10 and R@20 based on semantic-matching compared to exact-matching in absent keyphrase extraction.

The AutoKeyGen baseline demonstrates competitive performance in generating absent keyphrases on the `NUS` dataset. However, the generated keyphrases by AutoKeyGen are limited to the ones from the phrase bank of each dataset. In contrast, our approach leverages public knowledge graphs (such as DBpedia and BabelNet) to obtain relevant phrases as candidates for absent keyphrases.

Limitation of Our Work. In our experiment, we used the MAG framework to connect present keyphrases to DBpedia knowledge graph (see Sect. 3.3). In the `SemEval2010` and `Krapivin` datasets, we were unable to link present keyphrases, due to the lack of coverage for these keyphrases in the DBpedia knowledge graph. That is the reason for the missing values shown in the last two rows of Table 4 for these datasets. In our future work, we plan to integrate other knowledge graphs (e.g., YAGO and Wikidata) to extend the coverage of entity linking in the MAG framework.

4.4 Ablation Study (Q_3)

To answer Q_3, we analysed the impact of each component of our framework on the overall performance. For this purpose, we set up four variants of our framework. The first variant MultPAX-PKE was dedicated for only extracting present keyphrases, i.e., no absent keyphrase generation and thus no linking with knowledge graphs. We also created two variants of MultPAX with the purpose of evaluating the generation of absent keyphrases, namely MultPAX-$\text{AKE}_{\text{DBpedia}}$ and MultPAX-$\text{AKE}_{\text{BabelNet}}$. Furthermore, we configured the MAG framework to link present keyphrases only with DBpedia in case of MultPAX-$\text{AKE}_{\text{DBpedia}}$, and only with BabelNet for MultPAX-$\text{AKE}_{\text{BabelNet}}$. Finally, we benchmarked the entire framework $\text{MultPAX}_{\text{Full}}$ as our fourth variant.

Table 5 reports the evaluation results of each component in terms of *semantic-matching* F_1@5, and F_1@10 on the `Inspec` dataset, since it contains the highest ratio of absent keyphrases among the benchmark datasets. We can see that the performance of MultPAX-PKE is improved when linking with knowledge graphs, e.g., MultPAX-$\text{AKE}_{\text{DBpedia}}$ outperforms MultPAX-PKE by +0.41 in F_1@10. In addition, we noticed that our approach could retrieve more terms from DBpedia than BabelNet, since DBpedia contains more semantic ontologies (approximately 3.5 millions instances) extracted from Wikipedia information boxes. Finally, our MultPAX-$_{\text{Full}}$ showed an improved performance with

F_1-scores (0.911 in F_1@5, 0.763 in F_1@10) when incorporating both knowledge graphs (i.e., DBpedia and BabelNet) compared with individual variants. These findings conclude that each component of MultPAX contributes to the overall performance of our framework and answers our last research question Q_3.

Table 5. Ablation Study of MultPAX framework on `Inspec` dataset. F_1@K-scores are reported based on **semantic-matching** between the predicted and ground-truth keyphrases

MultPAX-variant	F_1@5	F_1@10
MultPAX-PKE	0.892	0.686
MultPAX-AKE$_{\text{BabelNet}}$	0.907	0.701
MultPAX-AKE$_{\text{DBpedia}}$	0.911	0.727
MultPAX$_{\text{Full}}$	**0.911**	**0.763**

5 Conclusion

This paper presents MultPAX framework, a multitask approach for extracting present and absent keyphrases, including three components: i) Present Keyphrase Extraction, ii) Absent Keyphrases Generation, and iii) Keyphrases Semantic Matching. In our approach, we employ a pre-trained language model (Bert) and knowledge graphs (DBpedia and BabelNet) in keyphrase extraction. Our experiments showed that pre-trained language models are capable of efficiently extracting present keyphrases. Furthermore, knowledge graphs proved to be valuable resources for generating keyphrases that are absent, especially in short text. In our future work, we plan to apply a bootstrapped approach for keyphrase extraction from DBpedia abstracts to find more relevant terms. In particular, we intend to apply MultPAX recursively on the abstracts of DBpedia entities. In addition, we will experiment with other knowledge graphs (e.g., YAGO and Wikidata) to extend the coverage of entity link in the MAG framework.

Supplemental Material Statement. We implemented our framework in `Python 3.7`, the source code and how-to-run instructions can be found at the GitHub repository.[5] Furthermore, we used the benchmarking datasets available in the Dropbox drive.[6] For the baseline models, we used `OpenNMT library`[7], which enables us to benchmark different state-of-the-art baselines in our experiments. In addition, we used the pre-trained embedding of Bert model, namely `all-MiniLM-L6-v2` with 384 embedding dimension from the `huggingface`

[5] https://github.com/dice-group/MultPAX.
[6] https://www.dropbox.com/s/aluvkblymjs7i3r/MULTPAX-Datasets.zip?dl=0.
[7] https://github.com/memray/OpenNMT-kpg-release.

`library`[8] v4.16. For the hardware requirements, we used a computing server with 256 GB memory and `Xeon(R) CPU E5-2630 v4` with 2.20 GHz to run our experiments.

Acknowledgments. This work has been supported by the German Federal Ministry for Economic Affairs and Climate Action (BMWK) within the projects RAKI (grant no 01MD19012B) and SPEAKER (grant no 01MK20011U) as well as by the German Federal Ministry of Education and Research (BMBF) within the projects COLIDE (grant no 01I521005D) and EML4U (grant no 01IS19080B). We are also grateful to Diego Moussallem for the valuable discussion on earlier drafts and Pamela Heidi Douglas for editing the manuscript.

References

1. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: DBpedia: a nucleus for a web of open data. In: Aberer, K., et al. (eds.) ASWC/ISWC -2007. LNCS, vol. 4825, pp. 722–735. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-76298-0_52
2. Bennani-Smires, K., Musat, C., Hossmann, A., Baeriswyl, M., Jaggi, M.: Simple unsupervised keyphrase extraction using sentence embeddings. In: Proceedings of the 22nd Conference on Computational Natural Language Learning, pp. 221–229 (2018)
3. Bougouin, A., Boudin, F., Daille, B.: Topicrank: graph-based topic ranking for keyphrase extraction. In: Proceedings of the Sixth International Joint Conference on Natural Language Processing, pp. 543–551 (2013)
4. Campos, R., Mangaravite, V., Pasquali, A., Jorge, A., Nunes, C., Jatowt, A.: Yake! keyword extraction from single documents using multiple local features. Inf. Sci. **509**, 257–289 (2020)
5. Chen, Q., Ling, Z.H., Zhu, X.: Enhancing sentence embedding with generalized pooling. In: Proceedings of the 27th International Conference on Computational Linguistics, pp. 1815–1826 (2018)
6. Chen, W., Gao, Y., Zhang, J., King, I., Lyu, M.R.: Title-guided encoding for keyphrase generation. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 6268–6275 (2019)
7. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
8. Gollapalli, S.D., Li, X.L., Yang, P.: Incorporating expert knowledge into keyphrase extraction. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 31 (2017)
9. Hulth, A.: Improved automatic keyword extraction given more linguistic knowledge. In: Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing, pp. 216–223 (2003)
10. Kim, S.N., Medelyan, O., Kan, M.Y., Baldwin, T.: Semeval-2010 task 5: automatic keyphrase extraction from scientific articles. In: Proceedings of the 5th International Workshop on Semantic Evaluation, pp. 21–26 (2010)

[8] https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2.

11. Kleinberg, J.M.: Authoritative sources in a hyperlinked environment. J. ACM (JACM) **46**(5), 604–632 (1999)
12. Krapivin, M., Autaeu, A., Marchese, M.: Large dataset for keyphrases extraction (2009)
13. Liang, X., Wu, S., Li, M., Li, Z.: Unsupervised keyphrase extraction by jointly modeling local and global context. In: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pp. 155–164 (2021)
14. Majumder, G., Pakray, P., Gelbukh, A., Pinto, D.: Semantic textual similarity methods, tools, and applications: a survey. Comput. Sist. **20**(4), 647–665 (2016)
15. Meng, R., Zhao, S., Han, S., He, D., Brusilovsky, P., Chi, Y.: Deep keyphrase generation. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (vol. 1: Long Papers), pp. 582–592 (2017)
16. Alami Merrouni, Z., Frikh, B., Ouhbi, B.: Automatic keyphrase extraction: a survey and trends. J. Intell. Inf. Syst. **54**(2), 391–424 (2019). https://doi.org/10.1007/s10844-019-00558-9
17. Mihalcea, R., Tarau, P.: Textrank: bringing order into text. In: Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing, pp. 404–411 (2004)
18. Moussallem, D., Usbeck, R., Röder, M., Ngonga Ngomo, A.C.: MAG: a multilingual, knowledge-base agnostic and deterministic entity linking approach. In: K-CAP 2017: Knowledge Capture Conference, p. 8. ACM (2017)
19. Navigli, R., Ponzetto, S.P.: Babelnet: the automatic construction, evaluation and application of a wide-coverage multilingual semantic network. Artif. Intell. **193**, 217–250 (2012)
20. Polstra III, R.M.: A case study on how to manage the theft of information. In: Proceedings of the 2nd Annual Conference on Information Security Curriculum Development, pp. 135–138 (2005)
21. Ray Chowdhury, J., Caragea, C., Caragea, D.: Keyphrase extraction from disaster-related tweets. In: The World Wide Web Conference, pp. 1555–1566 (2019)
22. Sahrawat, D., et al.: Keyphrase extraction as sequence labeling using contextualized embeddings. In: Jose, J.M., et al. (eds.) ECIR 2020. LNCS, vol. 12036, pp. 328–335. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45442-5_41
23. Shen, W., Wang, J., Han, J.: Entity linking with a knowledge base: issues, techniques, and solutions. IEEE Trans. Knowl. Data Eng. **27**(2), 443–460 (2014)
24. Shen, X., Wang, Y., Meng, R., Shang, J.: Unsupervised deep keyphrase generation. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 36, pp. 11303–11311 (2022)
25. Song, X., Salcianu, A., Song, Y., Dopson, D., Zhou, D.: Fast wordpiece tokenization. In: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pp. 2089–2103 (2021)
26. Vijayakumar, A.K., et al.: Diverse beam search: decoding diverse solutions from neural sequence models. arXiv preprint arXiv:1610.02424 (2016)
27. Wan, X., Xiao, J.: Single document keyphrase extraction using neighborhood knowledge. In: AAAI, vol. 8, pp. 855–860 (2008)
28. Wang, Y., Li, J., Chan, H.P., King, I., Lyu, M.R., Shi, S.: Topic-aware neural keyphrase generation for social media language. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pp. 2516–2526 (2019)
29. Xia, T., Wang, Y., Tian, Y., Chang, Y.: Using prior knowledge to guide bert's attention in semantic textual matching tasks. In: Proceedings of the Web Conference 2021, pp. 2466–2475 (2021)

30. Ye, H., Wang, L.: Semi-supervised learning for neural keyphrase generation. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pp. 4142–4153 (2018)
31. Ye, J., Cai, R., Gui, T., Zhang, Q.: Heterogeneous graph neural networks for keyphrase generation. arXiv preprint arXiv:2109.04703 (2021)
32. Zhao, J., Bao, J., Wang, Y., Wu, Y., He, X., Zhou, B.: SGG: learning to select, guide, and generate for keyphrase generation. In: Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 5717–5726 (2021)
33. Zhao, Y., et al.: Deep keyphrase completion. arXiv preprint arXiv:2111.01910 (2021)