Journal of Computers Vol. 30 No. 2, 2019, pp. 135-149
doi:10.3966/199115992019043002012

Automatic Tag Recommendation Approach with Keyphrase Extraction and Word Embedding Techniques

Taechawat Konkaew, Sukumal Kitisin*

Department of Computer Science, Kasetsart University, Bangkok, Thailand
{taechawat.k, sukumal.i}@ku.th

Received 15 December 2018; Revised 15 February 2019; Accepted 15 February 2019

Abstract. The number of online videos made available has been increasing rapidly. The most common way to find interesting videos is to search for the relevant tags. However, a large number of video clips may not have tags which cover in every aspect their contents, therefore searching for the relevant videos may not be very effective. Recent development in topic modeling research shows good performance of the word embedding model [1-2]. Our work proposed an automatic video tag recommendation approach from video transcript using a hybrid of unsupervised keyphrase extraction and word embedding model considering the semantic similarity between keyphrase candidates. Our experiments were performed on TED Talk videos dataset. The results show improvements over a variety existing approach. The proposed approach can help video owners identify a set of tags presumably having good coverage of the video contents. Our work could be applied in many existing video sharing platforms like YouTube, Twitch, etc.

Keywords: automatic tag recommendation, keyphrase extraction, natural language processing, word embedding

1 Introduction

Video sharing platform is a system which allows users to distribute their videos. Retrieving videos in the platforms depend on several properties which consist of video length, popularity, number of subscribers, and relevant tags. The common way to retrieve interesting videos is querying for relevant words/phrases which related to the videos, not only for title or description but also for metadata. At present, many video sharing platforms do not provide automatically generated video tags. Often, many video clips were left untagged. It is difficult for video consumers to be able to search for relevant video clips, especially when video clips are just published online. It takes time until those videos were watched and tagged and then indexed. Then, there is a chance that these videos would come up in the later search. In general, video tags are manually provided by the video owner. Creating appropriate tags for the video is not an easy task. Each video owner has a different view when deciding words/phrases to tags. Generally, tags can be words/phrases which significantly related to the video; for example, the name of the products, the main character names, the video topics, etc. In addition, some videos have inappropriate tags because the video owners try to make their videos easy to obtain by giving popular tags that do not relate to the content directly.

In order to be able to automatically tagging a video, one could extract video tags out of the video's features such as its title, metadata and transcript. The video tag recommender system should produce a set of tags that associated to the content and metadata of the video as much as possible.

Many researches on the video domain; for example, Guo and Gurrin from Dublin City University did short user-generated videos classification from audio features in 2012. They defined seven types of audio characteristics consist of speaking sounds, positive sounds such as clapping, laughing, etc., background music, outdoor rural noise, indoor noise, and quiet sound. They classified the videos corresponding to the

* Corresponding Author

above sound types using support vector machine (SVM) algorithm [3]. Bhuiyan, et al. from Dhaka demonstrated the retrieving of YouTube videos by analyze sentiments of user comments in 2017. The search results from YouTube comes back with irrelevant and low-quality videos in the higher order than the relevant and high-quality ones because those videos have larger view counts and larger likes. Hence, the video comments, metadata, and number of likes and dislikes are analyzed and applied to improve the relevance of the search result [4]. Another work by Chen, et al. from National Central University, Taiwan, is to classify YouTube videos into six emotion categories: - anger, fear, happiness, sadness, surprise, and disgust. They constructed an emotion dictionary which includes emotion-related terms for each category. Then they trained the voting ensemble classifier which is a combination of logistic regression, SVM, and naïve bayes and use the model to predict video categories [5].

Our motivation is inspired by YouTube's caption generation. Basically, generating the video caption on YouTube will be processed within 24 hours after the video is uploaded. Nowadays, YouTube's speech recognition framework is very effective and can capture words precisely for English. Video transcript is then made available within a day. Hence, using video caption be propose a technique in automatically generating tag recommendation for video owners.

2 Background

Automatic video tag recommendation can be done with Natural Language Processing (NLP) techniques which include keyphrase extraction and topic modeling techniques. Generally, researches on keyphrase extraction were performed on scientific publication data. Extracted keyphrase are mostly technical terms on scientific words/phrases which are available in scientific term database. Hence, the result of keyphrase extraction for scientific publication data yields high accuracy rate. However, video clip contents are generally cover much broader domains. Keyphrase extraction alone is not sufficient to recommend video tags. Thus, topic modeling techniques are combined to measure the semantic relation between tags in each video. The techniques include Word Embedding and Paragraph Vector representation.

2.1 Keyphrase Extraction

Keyphrase extraction techniques can be of supervised unsupervised approaches. For supervised approaches, classifier was created from training documents to classify keyphrases considering many features categorized into internal-collection features and external features such as KEA [6], Maui [7], CeKE [8]. For unsupervised approaches such as TextRank, RAKE, TopicRank, PositionRank, MultipartiteRank. The approaches can extract keyphrase from individual documents. Keyphrases from each document can be extracted. Keyphrases and keywords refer to key terms which provide a succinct topic in which a document is related [9]. These extracted keyphrases and keywords can be used to categorize documents and search for them efficiently. The keyphrase extraction performance is affected by many parameters, including the length of a document, document structure, topic changing, and topic correlation [10]. The first one, the length of a document, affects the performance of the keyphrase extractor. Most of the keyphrase extraction methods prioritize keyphrase candidates which are a set of nouns, compound nouns, and compound adjectives than others. Thus, the longer the length of a document the greater number of extracted keyphrases increase. Having large amount of keyphrase candidates, filtering process would be more difficult and time-taking. The second one, the document structure, contributes to the effectiveness of keyphrase extraction. Most researches on keyphrase extraction were done with scientific publication data which are structured documents containing abstract, introduction, related works, and so on. Such document has a solid, well-defined structure. Keyphrases appeared in some sections may be far more significant than those appeared in other sections. Meanwhile, unstructured documents like movie reviews news, one cannot tell the location of keyphrases which are significantly represent the main point of the articles, especially when review articles which authors may discuss on both positive and negative opinions. Some keyphrase extractors use keyphrase position offset in order to determine keyphrases significance of the document. Unstructured document made it difficult for such approaches.

The third one, the topic changing, which generally found in the unstructured documents. The keyphrase extractor does not know when the discussion topics change. For example, a news article contains a conversation between interviewees depending on the flow of conversation. The topic changing

can occur from time to time making it more difficult for the keyphrase extractor to be able to extract the main points. The last one is the correlation among documents. Typically, scientific publications in the same or related fields would share similar keyphrases. their contents related to each other in the same domain that affects to keyphrases are related to each other as well. On the other hand, multi-domain dataset like video clips (text context) belongs to various categories. Within a category, a video content may not relate to the others. It is difficult for extractors to accurately extract keyphrase from uncorrelated keyphrase candidates.

Even though, researchers have developed many keyphrase extraction techniques, they are still confronted with several limitations due to the computer does not actually understand the human natural language structure. Many approaches choose the important keywords/keyphrases from additional properties consisting of word frequency, phrase length, phrase prominence, phrase popularity, etc. Some unsupervised approaches, extract keyphrase based on word frequency resulting in extracting common words/phrases instead of the significant words/phrases. Supervised approaches use term frequency-inverse document frequency (TF-IDF) that represents how much of a word on a document is significant among those appeared in the training corpus. The word can often occur in a document but rarely occur in other documents in the training set. Hence, the performance of unsupervised approaches achieves lower accuracy than supervised approaches.

The phrase length is another considered factor in keyphrase extraction. It depends on characteristics of the dataset domain. Long phrase length is appropriate in some domain whereas only a single word is appropriate for other domains. In 2017 [11], Wu and et, al. stated that scientific keyphrase regularly should have more than one word. They extracted keyphrase candidates and then filtered those which match their filtering condition. Single words are generally filter out except specific named entity. However, video tags characteristic is different. On YouTube, the length of video tags typically can range from a single word up to three words.

Considering whether keyphrase candidates are prominent or not, is similar to TF-IDF concept. The actual concept is to find words/phrases which rarely or often appear in a document but they can represent the main subject of the document efficiently.

Extracted keyphrases may not be good candidates. The phrase popularity, the external features, can be taken into consideration. There are many external resources such as Wikipedia. Many researches done using Twitter dataset make use of extracted phrases from English Wikipedia articles retrieved on January 30, 2010. Score of each extracted phrase was computed from the probability that the phrase appears in Wikipedia articles as anchor texts divided by the number of the phrase appearing in all articles on Wikipedia. The score is called keyphraseness values. In total, the extracted phrases are greater than 4.1 million phrases [12-14]. Next, we will describe how keyphrase extraction generally work.

Keyphrase extraction is performed in two steps: (1) candidate selection: extracting set of words/phrases as candidates, (2) candidate filtering: identifying whether the candidate is a keyphrase or not using supervised/unsupervised techniques [9-10].

Candidate selection is a step defining word sequences with their particular grammatical context based on linguistic rules of natural language. Typically, most of keyphrase extraction approaches allow selecting certain types of part-of-speech such as nouns, adjective as keyphrase candidates. Most keyphrase candidates are usually compound nouns and compound adjectives. Some keyphrase extraction approaches allow selecting chunks of compound noun/adjective connecting with preposition or conjunction as well. Depending on the length of a document which affects to the number of keyphrase candidates, the selection conditions are specified to ignore spurious words/phrases and keep the number of keyphrase candidates as minimum as possible. To avoid the overgeneration of keyphrase candidates that will affect the result accuracy, there are optional keyphrase candidate normalization consisting of case folding, pruning (stop words removal), and stemming techniques.

Candidate filtering step is the challenging step of keyphrase extraction. As we mentioned before about natural language understanding limitation, it is very tough for filtering spurious keyphrase candidates out from the result. Many researchers in this field proposed several techniques to address this issue including supervised and unsupervised approaches.

In supervised keyphrase extraction, each supervised approach is a technique which combines machine learning algorithms include naïve bayes, decision trees, SVM, etc., to train a classifier with training documents. Supervised approaches require an assigned document dataset, it means each keyphrase candidate is marked as keyphrase-class or non-keyphrase-class, the technique is called a binary

classification. Various supervised keyphrase extraction approaches consider different features either internal-collection features or external features or both [10] to train the classifier. Internal-collection features consist of three feature types: statistical features, structural features, and syntactic features. The statistical features are the features that were converted into numerical feature values based on the training documents such as TF-IDF. Other statistical features are word/phrase distance and within-collection keyphraseness. The word/phrase distance is determined by the number of words which locate before the first occurrence phrase divided by the number of words in the document. Within-collection keyphraseness differs from the Wikipedia keyphraseness as described before. It searches in the other training documents whether the keyphrase candidates appears as other document keyphrase or not. The assumption is that words/phrases frequently indexed as a keyphrase is probably more likely to be keyphrases in the unseen documents. Statistical features are popular among supervised approaches. Structural features represent the different parts which keyphrases are placed in a document.

Syntactic features encode the grammatical context of each word by tagging part-of-speech attribute. Another syntactic feature is morphological suffix sequence. Each word has embedded suffix. For example, a word “slowness”, “-ness” is encoded to be a feature value for the word “slowness”. According to the experiment results from [15], syntactic features are not apparently useful for keyphrase extraction comparing with others.

In addition to Wikipedia external feature which is the famous external-based feature used in keyphrase extraction, search engine external feature can be applied. If a keyphrase candidate is shown on a query logs of a search engine, it is potentially be a keyphrase.

Basically, supervised approaches achieved higher keyphrase extraction accuracy than unsupervised approaches. One of the key factors is having a good amount of training corpus. Unfortunately, assigned keyphrases are not always available in many domains. In supervised approaches, the classifier determines each keyphrase candidate independently, it means the classifier cannot capture the semantic similarity among them.

Examples of supervised keyphrase extraction approaches are Keyphrase Extraction Algorithm (KEA), Maui, and Citation-Enhanced Keyphrase Extraction (CeKE).

Keyphrase Extraction Algorithm (KEA) was proposed by Frank, et al in 1999. A classifier is trained using naïve bayes from training documents with TF-IDF statistical feature and phrase distance. The classifier analyzes orthography properties such as line break, capitalization, punctuation, and so on.

Maui was developed by Medelyan, et al in 2009, an extension approach from KEA which attaches many additional features such as node degree which quantifies the semantic relation between keyphrase candidates, and spread which is the distance between first and last occurrences of a keyphrase in a document. A bagging ensemble classifier is used to predict keyphrases in unseen documents. The experiment was done on CiteULike dataset.

Citation-Enhanced Keyphrase Extraction (CeKE) was developed by Caragea, et al in 2014. The approach embedded citation network information as features and use it simultaneously with traditional supervised features. A classifier was trained with these features using naïve bayes algorithm. The experiments were performed on two machine learning conference publications which are World Wide Web (WWW) and Knowledge Discovery and Data Mining (KDD) dataset.

Unsupervised keyphrase extraction approaches are separated into graph-based, topic-based, simultaneous learning, and language modeling approaches [10].

Graph-based approaches which offer a strong extraction performance results are TextRank [16] and RAKE [17]. TextRank, a successful graph-based ranking was proposed by Mihalcea and Tarau in 2004. The word graph is constructed from texts and edges were computed by co-occurrence counts between words as the ranking of keyphrase candidates. TextRank can extract the important sentences from the given text effectively. Rapid Automatic Keyword Extraction (RAKE) was developed by Rose, et al. in 2010, RAKE is another graph-based approach adapting from TextRank. Each node of the graph represents contiguous words of each keyphrase candidate without any duplication. After the graph is constructed, a score of each word is calculated by

$$S(w_i) = \frac{\deg(w_i)}{freg(w_i)} \quad \textbf{(1)}$$

where $\deg(w_i)$ is the count of adjacent words of the word w_i including itself and $freq(w_i)$ is the frequency count of word w_i. After that, each word has its own score. Scoring each keyphrase candidate is defined as the sum of its adjacent word scores. The extracted keyphrases by RAKE usually contain multiple words concatenated to each other by conjunction or preposition.

TextRank takes more computing time than that of RAKE due to its complexity, but it yields better accuracy than RAKE. However, both of them are not able to capture semantic between keyphrase candidates. Many approaches were explored recently consist of TopicRank [18], PositionRank [19], MultipartiteRank [20].

TopicRank was developed by Bougouin, et al. in 2013. It groups keyphrase candidates that probably belonging to the same topic using hierarchical agglomerative clustering with average linkage. The average linkage is a distance metric between two clusters that is defined as the average distances between every pairs of instances between two clusters. The hierarchical agglomerative clustering works bottom-up having an individual instance then merge them according to the shortest average linkage between the two and so on until the targeted number of the clusters is reached. Then, a complete undirected graph is built and keyphrase candidates are selected from each cluster which are the most representative keyphrases of the cluster.

Another graph-based approach called PositionRank was proposed by Florescu and Caragea in 2017. It leverages additional features that are word position with word frequency, which made significant improvement on keyphrase extraction efficiency. PositionRank is sensitive to long documents. The performance of PositionRank depends on the length of a given document.

Different from TopicRank, MultipartiteRank which was proposed by Boudin in 2018, build a graph representation of keyphrase candidates. The multipartite graph is directed graph instead of undirected graph of TopicRank. Keyphrase candidate nodes connect only to other nodes from different topics and not to the nodes within the same topic. MultipartiteRank yields better keyphrase extraction accuracy than TopicRank.

In topic-based approaches, a keyphrase is probably relevant to one or more topics and the extracted keyphrases should cover all topics of a document. One of the approaches is Topical PageRank (TPR) [21]. TPR ensures that all topics of a document are covered by extracted keyphrases using topic modeling technique; Latent Dirichlet Allocation (LDA). The performance of topic model depends on the amount of the training documents.

In simultaneous learning approaches, an extractor can achieve better performance when keyphrase extraction and text summarization techniques work together. Language modeling leverages phraseness and informativeness features. It trains model with foreground corpus and background corpus. Foreground corpus is the normal collection of considering documents. Background corpus is an online external corpus which contains usual knowledge from various domains; for example, Wikipedia.

In our research we compare graph-based unsupervised approaches with our model. Most of keyphrase extraction approaches still have several obstructions directly affect to their performance [10]. The first one is an overgeneration error which happens when an extractor correctly extracts a candidate as a keyphrase, usually contains one word. It also extracts further candidates containing the keyphrase due to the frequently occur word. But the other keyphrase candidates are not matched with assigned keyphrases because they may be considered redundant terms. The second one is an in-frequency error which happens when an extractor cannot extract a keyphrase candidate appearing only once in a document. The third one is a redundancy error which is similar to overgeneration error except that the extracted keyphrase candidates with the same context. The last problem comes from evaluation protocol. Assigned keyphrases of a document do not contain any word appear in the document. Therefore, there is no way an extractor can extract such keyphrases from the document. Additionally, exact match causes evaluation error; for example, “student” and “students”. One can solve this issue by stemming every keyphrase candidates and assigned keyphrases before evaluating the extractor performance.

2.2 Topic Modeling

Topic modeling approaches are incorporated in keyphrase extraction in order to cover all topics of a document. One of the key techniques in topic modeling is word embedding which proposed by Mikolov et, al from Google Inc. in 2013. Word embedding is a technique for representing words in the numerical distributed vector space. Its goal is similar to the goal of topic modeling, is to dynamically clusters words

into a topic where words in the same topic are related to each other. Existing topic modeling techniques are Latent Semantic Analysis (LSA) [22], Probabilistic Latent Semantic Analysis (PLSA) [23], Latent Dirichlet Allocation (LDA) [24], and Correlated Topic Model (CTM) [25]. Latent semantic analysis (LSA) also called latent semantic indexing (LSI). It discovers words which appear together in the same document and group them together according to their similarity. Then, represents them in the distributed vector. LSA applies singular value decomposition (SVD) to rearrange the data. The method can handle general topics and their synonym but, except polysemy words. Addressing disadvantages of LSA, PLSA uses a probabilistic method instead of SVD to performing the task. PLSA find a probabilistic model with latent topics from the given documents. The model is later use to allocate various semantic of words from a single document to different topics. Hence, PLSA could handle polysemy words. LDA uses Dirichlet distribution to prevent the overfitting problem. The Dirichlet distribution is not actually normal probability distribution but, it is sampling over a probability simplex which is a collection of numbers that sum up to one. LDA showed better performance against LSA and PLSA significantly, although LDA is unable to verify correlation among generated topics. CTM solves the limitation of LDA. It uses the logistic normal distribution to measure the relations between generated topics. Nevertheless, CTM generated a large number of general words within the topics and requires a lot of calculation [26].

Topic modeling approaches require lot of computing time to create word vector from training dataset. The training complexity comes from number of word vocabularies. In English, the number of vocabularies is greater than 700k words. Measuring the similarity among these words in the training corpus, the vocabulary can lead to a very large dimension vector space O(n2) and the computation overhead is very expensive. Additionally, defining the number of topics that should be the most appropriate to the model is also challenging.

Topic modeling evaluation can be done by human or automatic evaluation. Human evaluation is time-consuming and expensive. Therefore, researchers have tried to find an automatic evaluation method using semantic relatedness of words within topics to avoiding the human evaluation. Topic coherence is a measurement to evaluate generated topics from a collection of documents. Each generated topic consists of many words depend on the number of words in the training corpus. The coherence model score is defined based on several those obtained from WordNet, Wikipedia, and Google search engine [27]. Therefore, the higher coherence score a model has the better topic grouping the model can perform.

In text mining, most machine learning algorithms require a fixed-length feature vector as the input. The common and well-known architecture to represent the fixed-length feature vector is bag-of-words (BOW) which has two notable disadvantages that it ignores the order of words and it loses word semantic. Two model architectures were proposed to address the two disadvantages by constructing the distributed vector representation of words, including continuous bag of words (CBOW) and continuous skip-gram. Word embedding can use either one of the architectures to efficiently create vector representation of words from large amounts of unstructured text data.

Word embedding achieved better performance of grouping similar words and speeded up training complexity which computed using artificial neural network. It captured more precise syntactic and semantic relationship of words than existing topic modeling techniques [28]. In addition to the grouping of similar words using neural network, Le and Mikolov tried to improve the model to handle the measurement of document similarity. In [29], Paragraph Vector was proposed as distributed representations of paragraph and document. The Paragraph Vector extends the word embedding by embedded a paragraph id to find the closest document.

3 Our Methodology

For automatic tag recommendation, we proposed an approach which recommends a set of relevant tags on each video using MultipartiteRank keyphrase extractor from video transcript. Additionally, we apply word embedding and Paragraph Vector in our approach. Our video tag recommendation is performed with following steps: (1) keyphrase extractor, (2) candidate score weighting, (3) word embedding and Paragraph Vector model construction (4) Similar Document Selection and Candidate Score Re-ranking (5) Common Candidate Removal.

3.1 Keyphrase Extractor

MultipartiteRank was chosen as our keyphrase extractor model. It is composed of 2 steps: (1) building a graph representation (2) Ranking keyphrase candidate nodes. In the first step, MultipartiteRank identifies keyphrase candidates which are sequences of compound noun and compound adjective and groups them into different topics using hierarchical clustering with average linkage [18].

An example given text from the Hulth-2003 dataset is shown in Fig. 1. The given text contains 91 words without removing duplicate and stop words. The underlined words/phrases are selected keyphrase candidates by the extractor, in total 21 keyphrase candidates. Let's us focus on 5 keyphrase candidates including "collections", "strong collections", "weak collections", "entire new libraries", and "libraries". These keyphrase candidates are clustered into 2 different topics based on stemmed overlapping. The minimum similarity threshold was set to 0.74 as default. We followed the default parameter. The first topic consists of 3 keyphrase candidates which are "collect", "strong collect", and "weak collect". the second topic also consists of 2 keyphrase candidates which are "entir new librari", and "librari".

Twenty years of the literature on acquiring out-of-print materials This article reviews the last two-and-a-half decades of literature on acquiring out-of-print materials to assess recurring issues and identify changing practices. The out-of-print literature is uniform in its assertion that libraries need to acquire o.p. materials to replace worn or damaged copies, to replace missing copies, to duplicate copies of heavily used materials, to fill gaps in collections, to strengthen weak collections, to continue to develop strong collections, and to provide materials for new courses, new programs, and even entire new libraries.

Fig. 1. The example of given text from the Hulth-2003 dataset

After discovering topics, the next step is to build a complete directed multipartite graph which nodes are keyphrase candidates and each edge has computed weight using semantic relation between adjacent nodes. These nodes are connected only if they are clustered into different topics. After the graph is constructed, each edge w_{ij} from node i to node j is initialized as the sum of inverse distance of offset between these two keyphrase candidates in a document by the following equation [20]:

$$w_{ij} = \sum_{p_i \in P(c_i)} \sum_{p_j \in P(c_j)} \frac{1}{|p_i - p_j|} \quad \textbf{(2)}$$

where $P(c_i)$ is the set of the word offset positions of candidate c_i. In Fig. 2, the incoming edge's weight of a node is adjusted by the outgoing weights of other nodes in the same topic connecting to the paired node in the different topic by the following equation [20]:

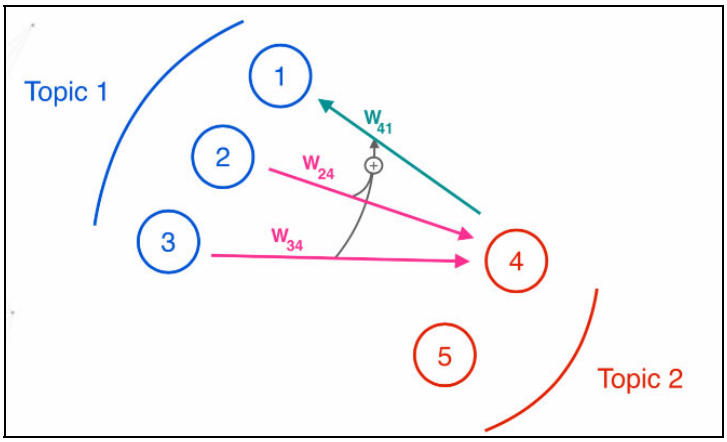

Fig. 2 An example of the weight adjustment; node 1 is promoted by the sum of weights of outgoing edges of node 2 and 3

$$w_{ij} = w_{ij} + \alpha \cdot e^{(\frac{1}{p_i})} \sum_{c_k \in T(c_j) \setminus \{c_j\}} w_{ki} \quad \textbf{(3)}$$

where w_{ij} is the edge weight between nodes c_i and c_j, α is a hyperparameter that controls the strength of the weight adjustment, $T(c_j)$ is the set of candidates belonging to the same topic as candidate c_j, and p_i is the offset position of the first occurrence of candidate c_i. Higher weight directly represents more related semantic between two keyphrase candidates.

Last step in keyphrase extractor is to rank keyphrase candidates with TextRank, a widely used graph-based ranking algorithm. Ranking score $S(c_i)$ can be calculated by [20]:

$$S(c_i) = (1-\lambda) + \lambda \cdot \sum_{c_j \in I(c_i)} \frac{w_{ij} \cdot S(c_j)}{\Sigma_{c_k} \in O(c_j) w_{jk}} \quad \textbf{(4)}$$

Where λ is a damping factor set to 0.85 [16], $I(c_i)$ is the set of predecessors of candidate c_i and $O(c_j)$ is the set of successors of candidate c_i. After that, top N highest ranked score keyphrase candidates are selected as keyphrases.

3.2 Candidate Score Weighting

From the previous step, each keyphrase candidate has its own score. The occurrence of keyphrase candidates play important role in determining keyphrases. For example, those in video title, video description. Keyphrase candidates which occur in these positions should be weighted more than those occur in the transcript. Wu et al. [11] revealed that a named entity recognition (NER) could improve an accuracy of keyphrase extraction. Therefore, named entity especially, person name or speaker received high weighting score to increase the probability in keyphrase selection. The weight is adjusted by:

$$S(c_i) = S(c_i) \cdot \beta \quad \textbf{(5)}$$

Where β is a threshold of the weighting strength and $S(c_i)$ is a score of candidate c_i.

3.3 Word Embedding and Paragraph Vector Model Construction

Our assumptions are that a newly uploaded video with the highest similarity score to another video will probably belongs to the same category of that video and then keyphrase candidates on both videos are related to each other.

A Paragraph Vector (Doc2Vec) model and Word Embedding model (Word2Vec) [28-29] are trained with video training corpus for measuring similarity between a newly uploaded video and each one in the training set and keyphrase candidate of them.

3.4 Similar Document Selection and Candidate Score Re-ranking

In this step, the most similar video was selected from the training set and named "targeted document" using Paragraph Vector model. Then targeted keyphrase candidates were selected by the MultipartiteRank keyphrase extractor. The similarity between words in keyphrase candidates and targeted keyphrase candidates is calculated by word embedding model. Then the keyphrase candidate score $S(c_i)$ is re-ranked by multiplying the similarity score of each word of the candidate with following equation:

$$S(c_i) = \prod_{k=1}^{n} S'(c_i) \cdot sim(w_k) \quad \textbf{(6)}$$

where $S'(c_i)$ is a prior computed score of candidate c_i, n is the number of words of candidate c_i, and $sim(w_k)$ is a similarity score for each word of candidate c_i. word embedding similarity score could be

either negative or positive. If the score is a negative value, it means two words are not related to each other. On other hand, if it is a positive value, they are related to each other.

3.4 Common Candidate Removal (CCR)

The result from previous step still give a large number of keyphrase candidates. Reducing number of keyphrase candidates we then apply common candidate removal technique. A set of common words is defined by counting a frequency of each word in the corpus. In each test document, there are different number of keyphrase candidates. A common candidate is determined by

$$S(c_i) = \frac{freq(c_i)}{n_{dj}} \tag{7}$$

where c_i is a keyphrase candidate, $freq(c_i)$ is the number of candidate c_i appeared in the test documents. and n_{dj} is the number of keyphrase candidates in document j. If $S(c_i)$ is greater than a threshold, c_i will be defined as a common keyphrase candidate therefore, can be removed from keyphrase candidate list.

4 Experiments & Results

Our research objective is automatically recommended video tags for video sharing platforms such as YouTube, Twitch, etc. Our video corpus was collected from TED Talks. Our experiments were performed on three collected datasets and used default parameters from the previous work [20] using [30] to build MultipartiteRank keyphrase extractor and Gensim library to build word embedding and Paragraph Vector in Python 3.6.

4.1 Data Collection

Our proposed method is not a fully unsupervised approach thus, each dataset is split into training set and test set. In a step to building a word embedding model, it is necessary to learn words from the training set. We performed our experiments based on three different datasets consist of:

Hulth-2003 [31], is a collection of paper abstracts in a computer science and information technology. A gold standard keyphrases were assigned by professional indexers. The dataset consists of 1,000 and 500 as training set and test set respectively.

Journal of Statistical Software (JoSS), is journal abstracts about statistical software and algorithms was exported from Scopus including title, abstract and assigned keyphrases on July 4, 2018. The dataset consists of 817 and 91 abstract articles as training set and test set respectively.

TED Talks (TED), a set of videos from successful speakers in various fields, to guide and inspire audiences. The dataset was collected from a TED website which include title, description and video transcript on September 16, 2017, consists of 2,211 and 193 videos as training set and test set respectively. TED videos do not provide with detailed tags, but only come with rough categories with the total of 415 different categories. Each video in the test set was assigned tags by professional indexers.

The number of instances, the number of average tokens per document, and the number of keyphrases per document for the three datasets are shown in Table 1.

Table 1. The statistics for our collection of datasets

	Instances	Avg. tokens/doc	Avg. keyphrases/doc
Hulth-2003	1500 instances	73 tokens	10 keyphrases
JoSS	908 instances	88 tokens	5 keyphrases
TED	2404 instances	892 tokens	10 keyphrases

4.2 Evaluation Metric

Evaluation metric for keyphrase extraction considers each candidate between the extracted keyphrases and the reference keyphrases using exact match and score them using precision (P), recall (R), and F1-Measure (F1). Precision is calculated by the number of correct extracted keyphrases divided by the

number of extracted keyphrases. Recall is calculated by the number of correct extracted keyphrases divided by the number of assigned keyphrases. F1-Measure is the harmonic mean of the precision and recall which is calculated by

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \quad \textbf{(8)}$$

The output keyphrases are removed redundant keyphrase candidate out from the list.

4.1 Experiments & Results

Our experiments were performed on three collected datasets comparing TopicRank, PositionRank, and MultipartiteRank to our approach, with and without common candidate removal step. Baselines were compared to our approaches and results detailed in Table 2.

Table 2. The comparison of precision (P), recall (R), and f1-measure (F1) among our approach and baselines, computed from top 10 extracted keyphrases

	Hulth-2003			JoSS			TED		
	P@10	R@10	F1@10	P@10	R@10	F1@10	P@10	R@10	F1@10
TopicRank	25.89	28.84	27.29	6.48	12.40	8.51	15.75	14.53	15.12
PositionRank	**32.16**	**36.35**	**34.13**	6.26	12.03	8.24	6.42	6.24	6.33
MultipartiteRank	26.94	30.80	28.74	6.81	13.43	9.04	**16.63**	**15.36**	**15.97**
Our Proposed Approach w/o CCR	27.46	31.49	29.34	6.59	13.10	8.77	13.32	12.37	12.83
Our Proposed Approach	31.50	34.58	32.97	**7.03**	**13.89**	**9.34**	12.59	11.65	12.10

Our first experiment was performed by following the proposed steps without adjusting the weight of keyphrase candidates which occur in the crucial positions such as video title, video description, etc. This experiment set a threshold (β) of the weighting strength as 1. On the Hulth-2003 dataset, PositionRank achieved the highest precision, recall, and F1-Measure. Even though PositionRank achieved the best performance unanimously on the dataset, it is also highly sensitive on the TED dataset because TED video transcripts are long and unstructured document. MultipartiteRank yielded the lower evaluation metric results than those of PositionRank. On the Hulth-2003 dataset even though, it was studied to perform better [20] Empirically, it can be explained that most of the abstract articles contain a large number of duplicate keyphrase candidates. PositionRank leverages the occurrence position of keyphrase candidates and keyphrase candidate frequency which makes it work well on this dataset. Another reason is that keyphrase candidates compose of a word with the same root. According to the results from the evaluation mechanism in [20], differs from ours that each keyphrase candidate was stemmed and evaluated using the Mean Average Precision (MAP). Hence, MultipartiteRank results in [20] come out better than those of PositionRank.

On the Journal of Statistical Software dataset, our proposed approach achieved the best performance among all the baseline methods and one without common candidate removal step. Surprisingly, our approaches achieved lower performance than TopicRank and MultipartiteRank on the TED dataset. The result of our approach without common candidate removal step achieved better performance than one with common candidate removal step because tags assign for in a video differ from assigned keyphrases in scientific publication papers. Video tags are usually common words and unlikely long technical keyphrases.

The experiment was repeated using our approach which achieved best result for each dataset with tuning threshold of the weighting strength ranging from 1 to 8. It is observed that the threshold of the weighting parameter affects the performance of our approach. It varies on each dataset.

On the Hulth-2003 dataset, our approach that was performed the weighting keyphrase candidate scores is all of the proposed steps, the result shows that after weighted keyphrase candidates that occurred in the title, the threshold of the weighting parameter which made the method achieved better performance is 2. The highest precision, recall, and F1-measure are 31.74%, 34.89%, and 33.24% respectively. By the way, the results after weighted the keyphrase candidates are still lower than PositionRank.

On the Journal of Statistical Software dataset our approach without the candidate weighting score already achieved highest results among others. Weighting keyphrase candidate scores that occurred in the

title, the results showed that the threshold of the weighting parameter yielding better performance is 3. The highest precision, recall, and F1-Measure are 7.36%, 14.81%, and 9.84% respectively. In the Hulth-2003 and Journal of Statistical Software datasets, the overall F1-Measure is improved for 0.385 percent on average. While in the TED dataset, the result of our approach without common candidate removal step with weighted keyphrase candidates in crucial positions including title, description, and the name of persons and speakers showed that the threshold yielding better performance is 7. The highest precision, recall, and F1-Measure are 17.77%, 16.70%, and 17.22% respectively. Our approach achieved significantly better F1-Measure for 4.39 percent higher. The weighted results detailed in Table 3.

Table 3. The comparison of precision (P), recall (R), and f1-measure (F1) of our best approach from table 2 with weighting keyphrase candidates, computed from top 10 extracted keyphrases

Weighted	Hulth-2003			JoSS			TED		
Threshold	P@10	R@10	F1@10	P@10	R@10	F1@10	P@10	R@10	F1@10
1	31.50	34.58	32.97	7.03	13.89	9.34	13.32	12.37	12.83
2	**31.74**	**34.89**	**33.24**	7.25	14.54	9.68	15.85	14.76	15.29
3	31.66	34.83	33.17	**7.36**	**14.81**	**9.84**	16.63	15.45	16.02
4	31.64	34.81	33.15	7.25	14.54	9.68	17.36	16.20	16.76
5	31.66	34.83	33.17	7.25	14.54	9.68	17.25	16.16	16.69
6	31.66	34.83	33.17	7.25	14.54	9.68	17.31	16.21	16.74
7	31.68	34.85	33.19	7.25	14.54	9.68	**17.77**	**16.70**	**17.22**
8	31.64	34.81	33.15	7.25	14.54	9.68	17.77	16.63	17.19

The MultipartiteRank also used weighted keyphrase candidates. Therefore, the experiments were performed by weighting keyphrase candidates on MultipartiteRank. The results showed that F1-Measure after weighted keyphrase candidates, on the first dataset, the extractor achieved an improvement of F1-Measure for 1.33 percent, however, our method gives better precision, recall, and F1-Measure at all. For the Journal of Statistical Software dataset, weighting keyphrase candidate scores yield worse results. We discerned that the extracted keyphrases were weighted and these keyphrase candidates were re-ordered and spurious keyphrase candidates gain better scores and presumed as keyphrases. On the TED dataset, the extractor achieved significant improvement for F1-Measure, up to 19.03 percent which was 3.06 percent increased as seen in Table 4. The overall F1-Measure of MultipartiteRank is greater than our approach with weighted keyphrase candidates. Because of the TED dataset contains the general words/phrases as assigned tags, the word embedding measures the similarity between keyphrase candidates and the most similar document's keyphrase candidates (candidate score re-ranking step). Similarity scores of general phrases were small. That is the reason why our approaches yield lower precision, recall, and F1-Measure than TopicRank and MultipartiteRank.

Table 4. The comparison of precision (P), recall (R), and f1-measure (F1) of MultipartiteRank approach with weighting keyphrase candidates, computed from top 10 extracted keyphrases

Weighted	Hulth-2003			JoSS			TED		
Threshold	P@10	R@10	F1@10	P@10	R@10	F1@10	P@10	R@10	F1@10
1	27.22	31.13	29.04	**6.81**	**13.43**	**9.04**	16.68	15.42	16.03
2	27.50	31.52	29.38	6.59	12.95	8.74	18.60	17.38	17.97
3	**27.50**	**31.53**	**29.38**	6.70	13.31	8.92	**19.59**	**18.51**	**19.03**
4	27.48	31.51	29.36	6.70	13.31	8.92	19.22	18.23	18.72
5	27.48	31.51	29.36	6.70	13.31	8.92	18.86	17.54	18.34
6	27.48	31.51	29.36	6.70	13.31	8.92	18.55	17.57	18.05
7	27.48	31.51	29.36	6.70	13.31	8.92	18.50	17.53	18.00
8	27.48	31.51	29.36	6.70	13.31	8.92	18.34	17.40	17.86

The TED videos belong to broad range of topic domains. We believed that our approaches would work well within the same domain. Therefore, a set of videos in one category were selected. In our experiment we selected TED videos from science domain. This dataset consists of 481 and 64 videos as training set and test set respectively. Our approach without common candidate removal step is selected. The result showed that our approach achieved better performance than MultipartiteRank for the same domain dataset. Our approach yielded the highest precision, recall, and F1-Measure when the threshold was set as

2 which are 18.44%, 17.38%, and 17.89% respectively whereas MultipartiteRank yielded the highest precision, recall, and F1-Measure when the threshold was set as 3 which are 17.66%, 17.00%, and 17.32% respectively as seen in Table 5.

Table 5. The comparison between our approach and MultipartiteRank on TED Science videos dataset

Weighted Threshold	Our Approach			MultipartiteRank		
	P@10	R@10	F1@10	P@10	R@10	F1@10
1	16.72	15.72	16.20	16.88	16.05	16.45
2	**18.44**	**17.38**	**17.89**	17.03	16.06	16.53
3	17.66	16.69	17.16	**17.66**	**17.00**	**17.32**
4	16.56	15.75	16.15	16.72	16.25	16.48
5	16.09	15.45	15.77	16.09	15.56	15.82
6	15.63	15.08	15.35	15.78	15.28	15.53
7	15.47	15.02	15.24	15.78	15.28	15.53
8	15.16	14.72	14.93	15.63	15.14	15.38

Referring to Table 1 average keyphrases on each dataset are 10, 5, 10 for Hulth-2003, Journal of Statistical Software, and TED dataset correspondingly. Another factor to be consider is number of extracted keyphrases. We explored the number of extracted keyphrase candidates which could make our approach achieves the best result for each dataset. Fig. 3 shows the curve of precision, recall, and F1-Measure for the number of extracted keyphrase candidates on each dataset. Firstly, there is a good probability that small number of extracted keyphrases would give better precision.

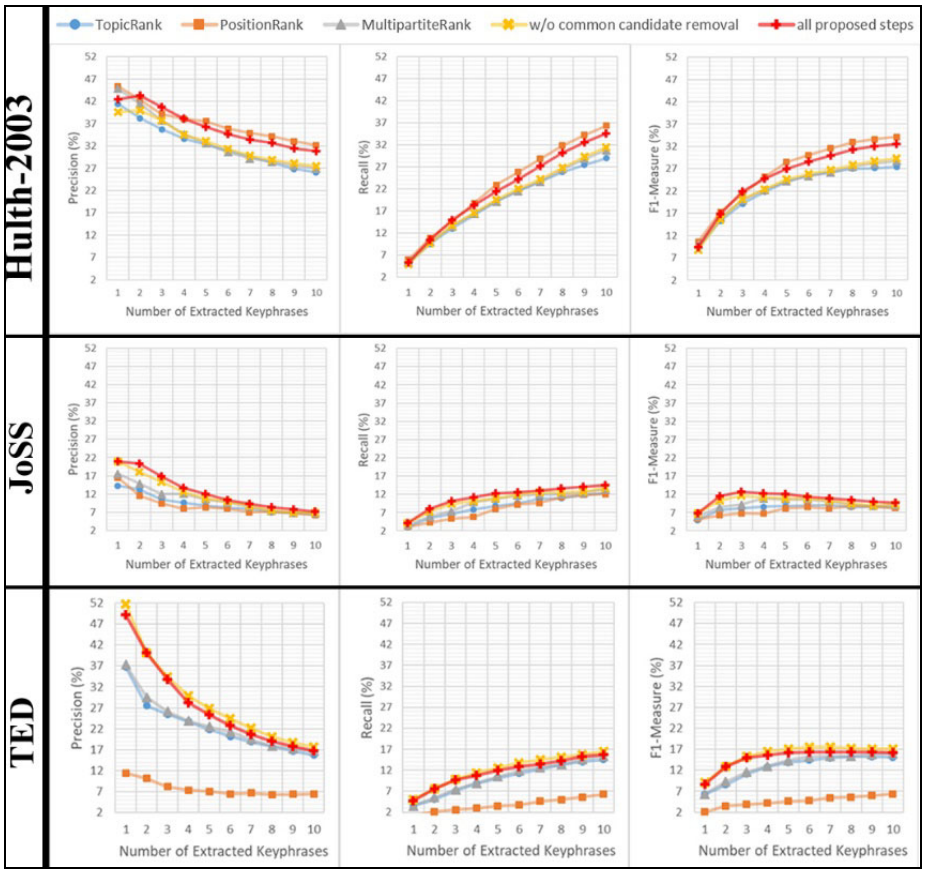

Fig. 3 The curve of precision, recall, and f1-measure for the number of extracted keyphrase candidates on each dataset

Thus, the larger the number of extracted keyphrases the smaller the precision result. The larger number of extracted keyphrases would give higher probability that the keyphrase candidates will match the assigned keyphrases. The larger number extracted keyphrases the larger the recall result. The graph of

F1-Measure is the harmonic mean of precision and recall among the three datasets.

The interesting observation appeared in the Journal of Statistical Software dataset, on the F1-Measure graph. F1-Measures were decreasing continuously when the number of extracted keyphrases is greater than 3. This is contrast to the results of other datasets which F1-Measure increased as the number of extracted keyphrases increased. This is because the average number of assigned keyphrases in the Journal of Statistical Software dataset is 5. If the number of extracted keyphrase is higher than the length of assigned keyphrases, then resulting in lower precision. Having lower precision when calculating F1-Measure the measure then also decreases.

5 Conclusion

We proposed an approach that could automatically recommend a set of tags for video clips using a combination between unsupervised keyphrase extraction and word embedding models. The score of each keyphrase candidate was computed with MultipartiteRank extractor. Then, keyphrase candidates were re-ranked by multiplying the similarity score between the words of the most similar document and the considering document were computed by the word embedding. The results demonstrated our approach performance improvement. The experiments were performed on three different datasets consist of Hulth-2003, Journal of Statistical Software, and TED datasets, and were compared with TopicRank, PositionRank, and MultipartiteRank.

Our approach works well on identifying keyphrases of video clips within the same topic domain therefore, it is suitable for automatic video tagging recommendation engine on online video sharing platforms. Nonetheless, it could perform reasonably well on multi-domain data as seen with the weighted keyphrase candidates of TED results. Our approach will assist and save video owner's time in tagging their newly uploaded videos. Our approach has a limitation in automatically tagging video which does not contain speaker or narrative transcript.

Acknowledgements

We are pleased to acknowledge support from Department of Computer Science and we would like to appreciate Ms. Lanlalit Srisathapornphat and her professional team for indexing the TED videos testing set.

References

[1] Y. Liu, Z. Liu, T.-S. Chua, M. Sun, Topical word embeddings, in: Proc. AAAI, 2015.

[2] D.Q. Nguyen, R. Billingsley, L. Du, M. Johnson, Improving topic models with latent feature word representations, Transactions of the Association for Computational Linguistics 3(2015) 299-313.

[3] J. Guo, C. Gurrin, Short user-generated videos classification using accompanied audio categories, in: Proc. the 2012 ACM International Workshop on Audio and Multimedia Methods for Large-scale Video Analysis, 2012.

[4] H. Bhuiyan, J. Ara, R. Bardhan, M.R. Islam, Retrieving YouTube video by sentiment analysis on user comment, in: Proc. 2017 IEEE International Conference on Signal and Image Processing Applications (ICSIPA), 2017.

[5] Y.-L. Chen, C.-L. Chang, C.-S. Yeh, Emotion classification of YouTube videos, Decision Support Systems, 101(2017) 40-50.

[6] E. Frank, G.W. Paynter, I.H. Witten, C. Gutwin, C.G. Nevill-Manning, Domain-specific keyphrase extraction, in: Proc. 16th International Joint Conference on Artificial Intelligence (IJCAI 99), 1999.

[7] O. Medelyan, E. Frank, I.H. Witten, Human-competitive tagging using automatic keyphrase extraction, in: Proc. the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3-Volume 3, 2009.

[8] C. Caragea, F.A. Bulgarov, A. Godea, S.D. Gollapalli, Citation-enhanced keyphrase extraction from research papers: a

supervised approach, in: Proc. the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014.

[9] S. Siddiqi, A. Sharan, Keyword and keyphrase extraction techniques: a literature review, International Journal of Computer Applications 109(2)(2015) 18-23.

[10] K.S. Hasan, V. Ng, Automatic Keyphrase extraction: a survey of the state of the art, in: Proc. ACL (1), 2014.

[11] J. Wu, S.R. Choudhury, A. Chiatti, C. Liang, C.L. Giles, HESDK: A hybrid approach to extracting scientific domain knowledge entities, in: Proc. 2017 ACM/IEEE Joint Conference on Digital Libraries (JCDL), 2017.

[12] C. Li, A. Sun, J. Weng, Q. He, Exploiting hybrid contexts for tweet segmentation, in: Proc. the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2013.

[13] C. Li, A. Sun, A. Datta, Twevent: segment-based event detection from tweets, in: Proc. the 21st ACM International Conference on Information and Knowledge Management, 2012.

[14] C. Li, J. Weng, Q. He, Y. Yao, A. Datta, A. Sun, B.-S. Lee, TwiNER: named entity recognition in targeted twitter stream, in: Proc. the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2012.

[15] S.N. Kim, M.-Y. Kan, Re-examining automatic keyphrase extraction approaches in scientific articles, in: Proc. the Workshop on Multiword Expressions: Identification, Interpretation, Disambiguation and Applications, 2009.

[16] T. Mihalcea, P. Textrank, Bringing order into texts, in: Proc. the Conference on Empirical Methods in Natural Language Processing, 2004.

[17] S. Rose, D. Engel, N. Cramer, W. Cowley, Text Mining: Applications and Theory, 1st ed., Wiley, New York, 2010.

[18] A. Bougouin, F. Boudin, B. Daille, Topicrank: Graph-based topic ranking for keyphrase extraction, in: Proc. International Joint Conference on Natural Language Processing (IJCNLP), 2013.

[19] C. Florescu, C. Caragea, Positionrank: An unsupervised approach to keyphrase extraction from scholarly documents, in: Proc. the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2017.

[20] F. Boudin, Unsupervised keyphrase extraction with multipartite graphs. <http://arxiv.org/abs/1803.08721>, 2018.

[21] Z. Liu, W. Huang, Y. Zheng, M. Sun, Automatic keyphrase extraction via topic decomposition, in: Proc. the 2010 Conference on Empirical Methods in Natural Language Processing, 2010.

[22] S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer, R. Harshman, Indexing by latent semantic analysis, Journal of the American Society for Information Science 41(6)(1990) 391-407.

[23] T. Hofmann, Probabilistic latent semantic analysis, in: Proc. the Fifteenth Conference on Uncertainty in Artificial Intelligence, 1999.

[24] D.M. Blei, A.Y. Ng, M.I. Jordan, Latent dirichlet allocation, Journal of Machine Learning Research 3(Jan)(2003) 993-1022.

[25] D. M. Blei, J.D. Lafferty, A correlated topic model of science, The Annals of Applied Statistics 1(1)(2007) 17-35.

[26] R. Alghamdi, K. Alfalqi, A survey of topic modeling in text mining, I. J. ACSA 6(1)(2015) 147-153.

[27] D. Newman, J.H. Lau, K. Grieser, T. Baldwin, Automatic evaluation of topic coherence, in: Proc. Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, 2010.

[28] T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: Proc. Advances in Neural Information Processing Systems, 2013.

[29] Q. Le, T. Mikolov, Distributed representations of sentences and documents, in: Proc. International Conference on Machine Learning, 2014.

[30] F. Boudin, pke: an open source python-based keyphrase extraction toolkit, in: Proc. COLING 2016, the 26th International

Conference on Computational Linguistics: System Demonstrations, 2016.

[31] A. Hulth, Improved automatic keyword extraction given more linguistic knowledge, in: Proc. the 2003 Conference on Empirical Methods in Natural Language Processing, 2003.