DOMAIN INDEPENDENT KEY TERM EXTRACTION FROM SPOKEN CONTENT BASED ON CONTEXT AND TERM LOCATION INFORMATION IN THE UTTERANCES

Hsien-Chin Lin, Chi-Yu Yang, Hung-Yi Lee, Lin-shan Lee

Graduate Institute of Communication Engineering
National Taiwan University

{eehsienchin, yangchiyi10, tlkagkb93901106}@gmail.com, lslee@gate.sinica.edu.tw

ABSTRACT

This paper proposes a domain independent approach for extracting key terms from spoken content based on context and term location information, or the sentence structures. Once it is trained with data of enough different domains, it is able to extract key terms in other unseen domains. This is obviously very attractive because of the unlimited number of domains over the Internet. Its performance degrades only very slightly with recognition errors, so very useful for spoken content. The basic idea here is that the sentence structures or context and term location information are in general domain independent, and remain essentially unchanged with recognition errors. For example, the fact that the key term for the sentence "The subject of this article is primarily about neural networks" is "neural networks" can be extended to any other unseen term other than "neural networks" in any other unseen domain, and this is more or less preserved under recognition errors. In the experiments a model trained with data for five different domains can extract key terms from data in the sixth unseen domain with very good performance.

Index Terms: **key term extraction, long-short term-memory (LSTM), reinforcement learning, domain independent, spoken content**

1. INTRODUCTION

Key term extraction tries to identify the important terms (words or phrases) from a document (in text or spoken) which carry the main subjects or concepts described by the documents. It is very helpful in many applications, for example, in document classification and clustering [1–3], documents indexing and retrieval [4, 5], and it helps the user browse and navigate across huge quantities of text [6]. For spoken content such as voice mails, meeting recordings, broadcast programs and so on, key term extraction is much more important than for text content, because it is difficult to show the spoken content on the screen, and difficult for the users to browse across spoken content. Also, many of the classified, clustered or retrieved spoken documents may not be desired for the user because of recognition errors. These spoken documents can be discriminated with the key terms [7].

Most frequently used key term extraction approaches can be either unsupervised or supervised [8–11]. The unsupervised approaches usually consider key term extraction as a ranking problem by simply ranking the terms in the document using some statistical features (e.g. tf-idf [12, 13]), semantic features (e.g. topic entropy [14]) or similar, or plus some re-ranking approach (e.g. graph-based algorithm [15–20]), and then selecting those terms with the highest ranking as the key terms. These approaches are usually good; but with relatively low accuracy.

On the other hand, supervised approaches very often considers key term extraction as a sequence classification problem [21–26]. The key terms are simply the labels of the classes for the document. So this approach is to classify every document to the correct classes labeled by the key terms. These approaches usually have higher accuracy, but inevitably limited by the training data. For example, key terms not existing in the training data cannot be obtained, because a classification model cannot classify something to an unseen class. This issue is important. Practically, because very naturally we need to extract key terms from documents (text or spoken) in unseen domain, while there exist essentially unlimited number of domains over the Internet.

In this paper, we present a novel approach for supervised key term extraction from both text or spoken content based on the context and term location information in the sentences or utterances, or the sentence or utterance structures. This means the model learns to identify key terms from sentence structures, which is independent of domain, so the model should be domain independent. For example, if the machine was trained that for the sentence "The subject of this article is primarily about neural networks" the key term is "neural networks", then it may learn that any term following "... this article is ... about" has higher probability to be a key term than other words in the sentence, whether the model has seen that term or that domain before or not. So this approach not only achieves the higher accuracy for supervised approaches, but solves the problem of unseen domains for supervised approaches. We also show this approach is equally useful for text and spoken content when applied on documents with and without recognition errors. This is because the context and term location information remain essentially unchanged with recognition errors.

2. PROPOSED APPROACH

The proposed approach is shown in Figure 1. The basic model in the left half will be presented in section 2.1, while the reinforcement learning in the right half which can be used to improve the performance will be discussed in section 2.2. In section 2.3 we will describe the way we test the model for spoken content by simulating the speech recognition errors.

2.1. Basic model

The basic model is in the left part of Figure 1. The recurrent neural networks (RNN) are very useful in capturing information from sequential inputs. Long short-term memory (LSTM) [27,28] networks can be considered as an extension of RNN. Here we use LSTM in the basic model to learn the relationships between sentence structures including context and term locations and the key terms in long sequences of words.

In Figure 1, $X = [x_1, x_2, ..., x_N]$ is the input sentence with N words, where x_i is the i-th word. $V = [v_1, v_2, ..., v_N]$ is the feature vector sequence of the input X, where v_i is the feature vector of x_i which is the concatenation of some word data and the word

Table 1. Basic information for the data set of the six domains considered

Domain	biology	cooking	travel	robotics	crypto	DIY
number of documents	13196	15404	19279	2771	10432	25918
vocabulary size	38257	24313	32072	17160	26792	32106
number of key terms	678	736	1645	231	392	734

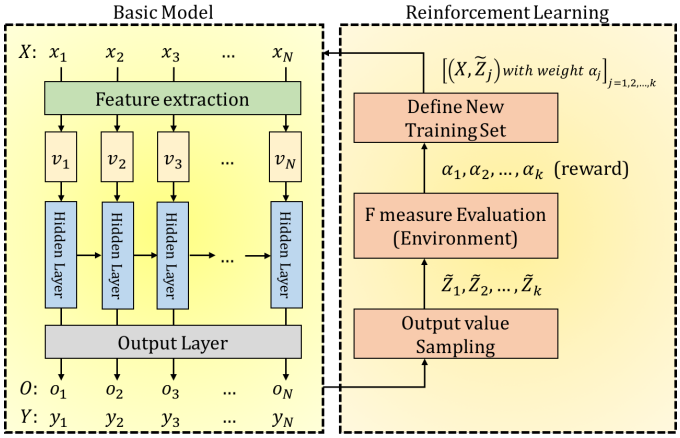

Fig. 1. The proposed model, basic model on the left and reinforcement learning on the right.

embeddings. The target $Y = [y_1, y_2, ..., y_N]$ has the same length as the input sentence, $y_i = 1$ if x_i is a key term, otherwise it equals to 0. In other words, the target is the prediction of whether each term position in the input sentence corresponds to a key term or not. The output $O = [o_1, o_2, ..., o_N]$ is the output of the model. During training, the model tries to have O as close to Y as possible. In the inference stage, we will set a threshold th, such that the word x_i is taken as a key term if $o_i > th$.

2.2. Reinforcement learning

It is not easy to train the basic model using conventional training mechanisms. The target contains much more 0's than 1's because most of the words are not key terms. As a result the conventional accuracy is not a good objective function for the training here. Instead we should be focused on how many correct key terms are identified, which means maximizing the F-measure is a better training target. However, F-measure is not differentiable, so it cannot be taken as the loss function. This is why reinforcement learning is used as shown in the right part of Figure 1.

With the basic model trained as mentioned above, for every input word x_i of the input $X_i = [x_1, x_2, ..., x_n]$ we have a output $o_i \in [0, 1]$. We can thus define a random variable z_i for it, which can be simply 1 or 0, but with a probability distribution

$$\begin{aligned} Prob(z_i = 1) &= o_i \\ Prob(z_i = 0) &= 1 - o_i \end{aligned} \tag{1}$$

We can then sample this distribution to obtain a sample value $\widetilde{z}_i$. All these samples for the input sentence $X = [x_1, x_2, ..., x_N]$ form an N-dimensional output sample vector

$$\widetilde{Z} = [\widetilde{z}_1, \widetilde{z}_2, ...\widetilde{z}_N]. \tag{2}$$

In this way we can obtain for each input sentence X a total of k output sample vectors $\widetilde{Z}_1, \widetilde{Z}_2, ..., \widetilde{Z}_k$, each $\widetilde{Z}_j$ is in the form of Eq. 2. For each pair of input X and output $\widetilde{Z}_j$ we can evaluate the F-measure α_j. The F-measure evaluation corresponds to the environment and the result α_j to the reward for the reinforcement learning. This defines a new training pair $(X, \widetilde{Z}_j)$ whose weight is α_j. All those training pairs are then used to re-train the basic model. Note that for each input X there are k output sample vectors $\widetilde{Z}_1, \widetilde{Z}_2, ..., \widetilde{Z}_k$ whose weights are $\alpha_1, \alpha_2, ..., \alpha_k$. So we have a total of k training pairs $(X, \widetilde{Z}_j)_{j=1,2,...,k}$ each with weight α_j. Therefore training pairs with higher F-measures will be weighted higher, and the model will learn more from them. In this way, the model naturally maximizes not only the accuracy but also the F-measure.

2.3. Speech recognition error simulation

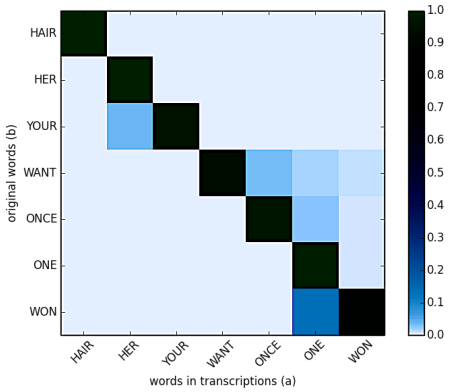

Fig. 2. A partial list of the confusion matrix obtained from Librispeech

The above approach is in general equally applicable to both text context and transcriptions of spoken content. Because it is not easy to find spoken context with labeled key terms of quality enough for training and testing, here we try to simulate transcriptions of spoken content by generating recognition errors, although there is a gap between the simulation and the real utterance. As the result, this way would provide us an indicator to let us know the results of our model applied on the spoken content. We model the recognition process as a transformation $F : X \longmapsto X'$, where $X = [x_1, x_2, ..., x_N]$ is the original correct word sequence for a spoken document and its corresponding recognition transcription is $X' = [x'_1, x'_2, ..., x'_N]$.

We can model the recognition process for a given recognizer with a confusion matrix between any two words in the vocabulary. By aligning the original text and the transcription from the recognizer, we can evaluate the probability that a word b is recognized as another word a

$$P(a|b) = \frac{count(x' = a; x = b)}{\sum_m count(x' = m, x = b)} \tag{3}$$

where a, b, m are all words, and $count(x' = a; x = b)$ is count of word b being recognized as word a. The summation is over all words m in the vocabulary. Deletion and insertion can be equally handled by defined a "blank" as an extra word in the vocabulary. Note that this matrix is not symmetric. This matrix can be trained for any specific recognizer using its input/output data.

A partial list of some examples for the above confusion matrix trained on Librispeech [29] is in Figure 2, where the WER is 5%. Seven words are shown in Figure 2, clearly divided into two groups, $[hair, her, your]$ and $[want, once, one, won]$. The words in the same group have higher probabilities to be recognized as one another. For example, when a word "her" exists in the transcription,

Table 2. The precision, recall and F-measure for DIY domain and F-measures for all other five domains when the key terms were extracted from an unseen domain with a model trained by the data sets for other five domains. All data are completely correct with out recognition errors. Row (a) for oracle, (b)(c) for unsupervised baselines, (d) for supervised baselines, and (e)(f)(g) for proposed approach.

Test Domain			DIY			biology	cooking	travel	robotics	crypto
Models			Precision	Recall	F-measure	F-measure				
(a) Oracle			0.794	0.593	0.679	0.359	0.672	0.578	0.651	0.684
Baseline	(b) Tf-idf sorting		0.176	0.271	0.213	0.095	0.249	0.206	0.193	0.224
	(c) Text Rank		0.181	0.213	0.195	0.084	0.242	0.154	0.151	0.178
	(d) Classification LSTM		0.283	0.282	**0.282**	**0.208**	0.221	**0.243**	0.180	0.222
Proposed	Basic	(e) embedding only	0.229	0.243	0.235	0.102	0.224	0.211	0.185	0.237
		(f) plus word data	0.241	0.245	0.242	0.103	0.285	**0.229**	0.202	0.252
	(g) Basic (f) plus RL		0.242	0.312	**0.272**	**0.113**	**0.318**	0.215	**0.246**	**0.255**

very often it is "her" in the original utterance. If it is incorrectly recognized, it is more likely to be "your" while less likely to be "hair". For a word "once" in an utterance to be recognized, very often it is correctly transcribed. If it is incorrectly recognized, it is more likely to be transcribed as "one" but less likely to be transcribed as "won" or "want". This means the confusion matrix jointly reflects the functions of acoustic and language models. For example, "her" is phonetically closer to "hair" but linguistically closer to "your".

3. EXPERIMENTS

In this section we describe the details of the experiments. The experimental setup, baselines and results are respectively in sections 3.1, 3.2 and 3.3.

3.1. Experimental setup

3.1.1. Data sets

The data set used in this word was parsed from StackExchange[1], a website for Q&A community, where users post their questions and answers plus some key terms for the questions and answers. The data set contains six domains: biology, cooking, travel, robotics, crypto and DIY. The basic information including number of documents, vocabulary size and number of key terms for each of the data set is listed in Table 1. Every document here has a title and the content, labeled with one or more key terms.

Below we choose one domain to be the test set and the others to be the training set in order to evaluate the ability of the proposed approach to extract the key terms in an unseen domain not existing in the training set. This process was repeated for each domain taking as the test set.

3.1.2. Preprocessing

The documents are very noisy, for example, with Html tags, URLs, non-english words, symbols, or equations. We filtered out these noisy parts and transformed all words to lowercase.

3.1.3. Feature extraction

The feature representing every word consisted of two parts. The word data and the word embedding. The former of a word contained the term frequency (tf), the inverse document frequency (idf), tf-idf, the word counts in the domain, and the position of the word in the sentence. The word embedding was obtained via Word2vec [30] which compressed each word into a vector of 200 dimensions. These two parts are then concatenated to represent each word.

3.2. Baselines

The baseline used in the experiments included both unsupervised and supervised approaches as introduced below.

[1] https://stackexchange.com/

3.2.1. Unsupervised baselines

There are two unsupervised baselines used here. The first was **tf-idf sorting**. All words of a document were sorted based on the tf-idf values, and the top-N words on the list were regarded as key terms. The second was **TextRank** [16], which performed random walk over a graph of words constructed based on the relationship among words, very similar to PageRank, where the co-occurrence features were used to estimate the relationship between two words.

3.2.2. Supervised baselines

Classification LSTM [26] was taken as the supervised baseline in this work. This method used a sequential classification model which classified each input document as a sequence to its class labeled by a key term using LSTMs. So the test data had to be in the same domain as the training set. The machine cannot classify a document to an unseen class.

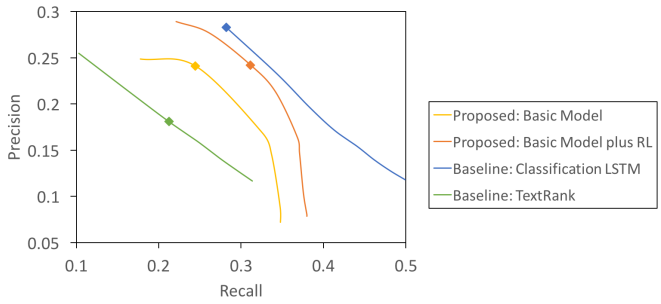

Fig. 3. Precision-Recall curve for the DIY domain with a model trained with the other five domains. No recognition errors. Those data listed in Table 2 are also marked.

3.3. Experimental results

3.3.1. Without recognition errors

The proposed model was trained with five domains of the corpus, then tested with the sixth domain. So the test domain was unseen to the model. When the test set included only reference text without recognition errors, the results for all the six domains taken as tested domain are listed in Table 2. The first domain (DIY) in the first column also has the precision and recall rates shown in the Table, with the precision-recall curves also plotted in Figure 3. The oracle results in row (a) were obtained by simply picking up those correct key terms appearing in the document. The recall is not 1.0 for Oracle because the reference key terms may not appear in the documents. The precision is not 1.0 for Oracle because the precision was set to zero for those documents for which no key term was

Table 3. F-measure for supervised classification LSTM (part (A)) and proposed approach (part (B)) under different WERs for the six domains

Model	WER	DIY		biology		cooking		travel		robotics		crypto	
(A) classification LSTM (Baseline, row (d) of Table 2)	0%	**0.282**	-	**0.208**	-	0.221	-	**0.243**	-	0.180	-	0.222	-
	5%	0.203	-28%	0.098	-53%	0.161	-27%	0.135	-44%	0.135	-25%	0.130	-41%
	10%	0.201	-29%	0.094	-55%	0.155	-30%	0.126	-48%	0.134	-25%	0.130	-41%
	20%	0.203	-28%	0.092	-56%	0.151	-32%	0.127	-47%	0.133	-26%	0.123	-45%
	30%	0.123	-56%	0.091	-56%	0.156	-29%	0.119	-51%	0.133	-26%	0.126	-43%
(B) Basic (f) plus RL (row (g) of Table 2)	0%	0.272	-	0.113	-	**0.318**	-	0.215	-	**0.246**	-	**0.255**	-
	5%	0.272	-0%	0.113	-0%	0.315	-0.9%	0.214	-0.5%	0.239	-2.8%	0.255	-0%
	10%	0.271	-0.4%	0.113	-0%	0.315	-0.9%	0.214	-0.5%	0.236	-4.1%	0.254	-0.4%
	20%	0.267	-1.8%	0.112	-0.9%	0.311	-2.2%	0.213	-0.9%	0.235	-4.5%	0.254	-0.4%
	30%	**0.267**	-1.8%	**0.111**	-1.7%	**0.292**	-8.2%	**0.198**	-7.9%	**0.233**	-5.3%	**0.244**	-4.3%

obtained because all reference key terms do not appear in the documents. Rows (b)(c) are for the unsupervised baselines, and row (d) for the supervised baseline trained with completely in-domain data. Rows (e)(f) are for the proposed basic model on the left of Figure 1, with word embedding only or plus the word data used in the word features. Row (g) is then for row (f) plus reinforcement learning.

In Table 2, from rows (b)(c), we can see the unsupervised baselines had lower precision. By comparing rows (d) with (b)(c), we can see the supervised classification LSTM was significantly better than the unsupervised baselines in three out of the six domains. By comparing rows (e)(f) we see the word data are certainly useful in our basic model. Comparing rows (f)(g), we can find out the reinforcement learning can improve the performance in most cases, and the improvement was significant for at least three out of six domains (DIY, cooking, robotics). By comparing rows (f)(g) with (b)(c), we find out that the proposed approach outperformed unsupervised baselines in all domains. From rows (d)(f)(g), we see that the proposed models performed better than the supervised classification LSTM for three out of the six domains (cooking, robotics, crypto), even though the proposed models didn't see the test domain data but the supervised classification LSTM did. Even for the other three domains (biology, travel, DIY), the proposed models for unseen domains are not too far from the supervised classification LSTM trained with the in-domain data for two of the tree domains (travel, DIY). In any case the proposed models didn't see the in-domain data but the supervised classification LSTM did. These verified the proposed approach was able to extract key terms in unseen domain very well. These results also verified that in the proposed approach the key terms are extracted based on the context information or sentence structures sentence by sentence individually without considering the whole document, and that is reasonably good.

From the precision-recall curves for the DIY domain in Figure 3, we see the supervised classification LSTM performed the best but was trained with completely in-domain data. The proposed basic model plus reinforcement learning is only slightly worse than the classification LSTM but very close, however it didn't see any test domain data before, and was trained with the out-domain data only. The basic model or unsupervised TextRank is obviously worse.

3.3.2. With recognition errors

For spoken content being transcribed by speech recognition, we tried to perform the experiments on the same test data set but with simulated recognition errors. We trained a confusion matrix $P(a|b)$ as explained in section 2.3 using Liberispeech whose WER is 5%. In this case we evaluated the probability that each word b was incorrectly transcribed, some higher (e.g. 8%) and some lower (e.g. 1%) while the overall WER was 5%. In each case those 1% or 8% of incorrectly transcribed words b were randomly assigned to other words a based on $P(a|b)$ as obtained. We also simulated the case for other WER's such as $\beta \cdot 5\%$ where β is a real number. In this case those words incorrectly recognized with probability 8% were assumed to be incorrectly recognized with probability $\beta \cdot 8\%$ and randomly assigned to other words based on the same confusion matrix $P(a|b)$ and so on. The results are in the Table 3 for all the six domains for word error rate (WER) ranging from 5% to 30%, comparing the proposed basic model plus reinforcement learning (row (g) in Table 2) with the supervised classification LSTM (row (d) of Table 2).

From part (A) of Table 3 for the supervised classification LSTM, we can see that with recognition errors at WER ranging from 5% to 30%, the performance was seriously degraded by 25% to 60% in all domains in all cases, and the degradation was more serious with higher WER. On the other hand, in part (B) for the proposed approach, the performance was only very slightly degraded with recognition errors (by 0-8.2%). These are also shown in Figure 4 (a)(b) for two domains, DIY and cooking. This is reasonable, because the proposed approach extracted the key terms based on the context information or sentence structures, which remains essentially unchanged with recognition errors as long as the WER is not too high.

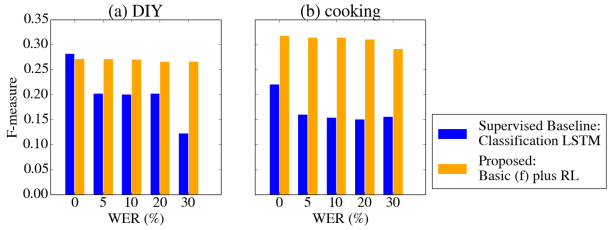

Fig. 4. F-measures under different WERs for (a) DIY and (b) cooking domain

4. CONCLUSION

A novel domain independent approach to extract key terms from spoken content based on context and term location information in the utterances is proposed in this paper. In other words, once trained with data of enough different domains, it can extract key terms in other unseen domains. This is extremely attractive because there exist unlimited number of domains over the Internet. It is also shown that the performance of this approach degrades only very slightly with speech recognition error, this is because sentence structure or context and term location information are in general domain independent, and remain essentially unchanged with recognition errors, therefore, specially useful for key term extraction from spoken content.

5. REFERENCES

[1] Khaled M Hammouda, Diego N Matute, and Mohamed S Kamel. Corephrase: Keyphrase extraction for document clustering. In *MLDM*, volume 2005, pages 265–274. Springer, 2005.

[2] Juhyun Han, Taehwan Kim, and Joongmin Choi. Web document clustering by using automatic keyphrase extraction. In *Proceedings of the 2007 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology-Workshops*, pages 56–59. IEEE Computer Society, 2007.

[3] Aytuğ Onan, Serdar Korukoğlu, and Hasan Bulut. Ensemble of keyword extraction methods and classifiers in text classification. *Expert Systems with Applications*, 57:232–247, 2016.

[4] Steve Jones and Mark S Staveley. Phrasier: a system for interactive document retrieval using keyphrases. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 160–167. ACM, 1999.

[5] Joel L. Fagan. Automatic p h r a s e indexing for document retrieval: An examination of syntactic and non-syntactic methods. *SIGIR Forum*, 51(2):51–61, August 2017.

[6] Felice Ferrara, Nirmala Pudota, and Carlo Tasso. A keyphrase-based paper recommender system. In *IRCDL*, pages 14–25. Springer, 2011.

[7] Lin-shan Lee and Berlin Chen. Spoken document understanding and organization. *IEEE Signal Processing Magazine*, 22(5):42–60, 2005.

[8] Olena Medelyan, Eibe Frank, and Ian H Witten. Human-competitive tagging using automatic keyphrase extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3-Volume 3*, pages 1318–1327. Association for Computational Linguistics, 2009.

[9] Vicky Min-How Lim, Siew Fan Wong, and Tong Ming Lim. Automatic keyphrase extraction techniques: A review. In *Computers & Informatics (ISCI), 2013 IEEE Symposium on*, pages 196–200. IEEE, 2013.

[10] Kazi Saidul Hasan and Vincent Ng. Automatic keyphrase extraction: A survey of the state of the art. In *ACL (1)*, pages 1262–1273, 2014.

[11] Zakariae Alami Merrouni, Bouchra Frikh, and Brahim Ouhbi. Automatic keyphrase extraction: An overview of the state of the art. In *Information Science and Technology (CiSt), 2016 4th IEEE International Colloquium on*, pages 306–313. IEEE, 2016.

[12] Kuo Zhang, Hui Xu, Jie Tang, and Juanzi Li. Keyword extraction using support vector machine. *Advances in Web-Age Information Management*, pages 85–96, 2006.

[13] Zhiyuan Liu, Peng Li, Yabin Zheng, and Maosong Sun. Clustering to find exemplar terms for keyphrase extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 257–266. Association for Computational Linguistics, 2009.

[14] Yun-Nung Chen, Yu Huang, Sheng-Yi Kong, and Lin-Shan Lee. Automatic key term extraction from spoken course lectures using branching entropy and prosodic/semantic features. In *Spoken Language Technology Workshop (SLT), 2010 IEEE*, pages 265–270. IEEE, 2010.

[15] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[16] Rada Mihalcea and Paul Tarau. Textrank: Bringing order into text. In *EMNLP*, volume 4, pages 404–411, 2004.

[17] Adrien Bougouin, Florian Boudin, and Béatrice Daille. Topicrank: Graph-based topic ranking for keyphrase extraction. In *International Joint Conference on Natural Language Processing (IJCNLP)*, pages 543–551, 2013.

[18] Soheil Danesh, Tamara Sumner, and James H Martin. Sgrank: Combining statistical and graphical methods to improve the state of the art in unsupervised keyphrase extraction. In * *SEM@ NAACL-HLT*, pages 117–126, 2015.

[19] Juan Martinez-Romo, Lourdes Araujo, and Andres Duque Fernandez. Semgraph: Extracting keyphrases following a novel semantic graph-based approach. *Journal of the Association for Information Science and Technology*, 67(1):71–82, 2016.

[20] Javad Rafiei-Asl and Ahmad Nickabadi. Tsake: A topical and structural automatic keyphrase extractor. *Applied Soft Computing*, 58:620–630, 2017.

[21] Eibe Frank, Gordon W Paynter, Ian H Witten, Carl Gutwin, and Craig G Nevill-Manning. Domain-specific keyphrase extraction. In *16th International Joint Conference on Artificial Intelligence (IJCAI 99)*, volume 2, pages 668–673. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999.

[22] Peter D Turney. Learning algorithms for keyphrase extraction. *Information retrieval*, 2(4):303–336, 2000.

[23] Peter D Turney. Learning to extract keyphrases from text. *arXiv preprint cs/0212013*, 2002.

[24] Kamal Sarkar, Mita Nasipuri, and Suranjan Ghose. A new approach to keyphrase extraction using neural networks. *arXiv preprint arXiv:1004.3274*, 2010.

[25] Rui Wang, Wei Liu, and Chris McDonald. Corpus-independent generic keyphrase extraction using word embedding vectors. In *Software Engineering Research Conference*, volume 39, 2014.

[26] Sheng syun Shen and Hung yi Lee. Neural attention models for sequence classification: Analysis and application to key term extraction and dialogue act detection. In *INTERSPEECH*, 2016.

[27] Felix A Gers and E Schmidhuber. Lstm recurrent networks learn simple context-free and context-sensitive languages. *IEEE Transactions on Neural Networks*, 12(6):1333–1340, 2001.

[28] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[29] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 5206–5210. IEEE, 2015.

[30] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.