Dependency Graphs for Summarization and Keyphrase Extraction

We present a real-time long document summarization and key-phrase extraction algorithm that utilizes a unified dependency graph.

YIFAN GUO

University of Texas at Austin

DAVID BROCK

Dallas College

ALICIA LIN

University of North Texas

TAM DOAN

University of North Texas

ALI KHAN

University of North Texas

PAUL TARAU

University of North Texas

We introduce a graph-based summarization and keyphrase extraction system that uses dependency trees as inputs for building a document graph. The document graph is built by connecting nodes containing lemmas and sentence identifiers after redirecting dependency links to emphasize semantically important entities. After applying a ranking algorithm to the document graph, we extract the highest ranked sentences as the summary. At the same time, the highest ranked lemmas are aggregated into keyphrases using their context in the dependency graph. Our algorithm specializes in handling long documents, including scientific, technical, legal, and medical documents.

CCS CONCEPTS • Real-time system architecture • Graph theory • Information extraction

Additional Keywords and Phrases: Keyphrase extraction, Unified dependency graph, Long document summarization

1 INTRODUCTION

Neural network-based models have been successful at both extractive and abstractive summarization on short documents. However, they require preprocessing such as chunking [8, 30] and encoding [33] for success on complex and lengthy documents, for which summarization has more practical uses. Without preprocessing, most deep learning models are only suitable for specific types of documents and have run times that increase significantly as the lengths of documents processed increase. For example, even though transformers [31] achieve high performance on multi-domain short documents, they have limited scalability on longer documents due to their quadratic computational and memory complexities [12]. On the other hand, even though some models [3] achieve high performance on long documents, they require splitting of a document into subsections in order to process each subsection separately, an approach that will not work for real-time applications on lengthy documents that have not been pre-processed.

In order to effectively handle complex and lengthy documents, including scientific papers and legal documents, in real time, we use a text graph-based approach. To provide additional information about a document, we enable both summarization and keyphrase extraction through the creation of a unified text graph that contains both sentence identifiers and lemmas.

Our main contributions include:

- an algorithmic model simultaneously supporting keyphrase extraction and summarization, with capability for real-time application and a focus on long documents.
- experiments on five publicly available keyphrase extraction datasets demonstrating that our system achieves higher performance than other algorithmic models and comparable performance to most neural network-based models.
- experiments on two publicly available summarization datasets demonstrating that our model achieves higher ROUGE-L score than most other models, both neural network-based and algorithmic.

2 RELATED WORK

Our system relies on a text graph built from dependency links that integrates words and sentences in one graph, resulting in a unified algorithm that enables both keyphrase extraction and summarization. Mihalcea and Radev [23] provide a comprehensive overview of graph-based natural language processing. The approach utilized in TextRank [24] is highly representative of the method taken by many graph-based systems: it extracts keyphrases using word co-occurrence relations controlled by the distance between words and computes sentence similarity as content overlap, giving weights to the links that refine the original PageRank algorithm [29]. Our main innovations with respect to graph-based systems such as TextRank, besides the use of dependency-tree based graphs, are the addition of SVO relations extracted from the text as well as relations extracted from WordNet. In contrast to neural network-based systems, our model, along with several others [9, 34, 37] belongs to the category of graph-based algorithmic models.

2.1 Keyphrase Extraction

To circumvent the requirement for extensive training, algorithmic extractive models that require no training have been proposed. Among them, Wan and Xiao [38] utilize information in nearby documents to enhance keyphrase extraction on a target document. Grineva et al. [11] filter out noise information and effectively process multi-theme documents by partitioning each graph into thematically cohesive groups of terms. Several other unsupervised models focus on the topics of the documents. Among those, Liu et al. [18] construct a Topical PageRank (TPR) on a word graph to measure word importance with respect to different topics. Bougouin et al. [2] cluster keyphrases into topics and assign a significance score to each topic using a graph-based ranking model. Zhang et al. [45] treat topics in documents as heterogeneous relations between words and construct a multi-relational word network.

Compared to most other algorithmic models, our model has the advantage of enabling both summarization and keyphrase extraction by using unified dependency graphs that contain both lemmas and sentence identifiers. Compared to neural network-based models, our model has the advantage of being able to work in real-time on document types that are not often included in training sets.

2.2 Summarization

A popular approach to summarization is to use a hybrid method that can be divided into two major steps. In the first step, fragments are extracted from the original text. In the second step, a summary is formulated based on those fragments, usually using neural network-based methods. Among the models taking this approach, Galanis and Androutsopoulos [10] compress sentences by removing words, Chen and Bansal [6] first select salient sentences and then rewrite them abstractively, and Lebanoff et al. [16] attempt to summarize by both compressing single sentences and fusing pairs through ranking sentence singletons and pairs together. Bae et al. [1] combine extractive and abstractive summarization and maximize ROUGE scores through a training procedure that globally optimizes summary-level ROUGE metrics. Mendes et al. [21] do not require length constraints typical to extractive summarization due to dynamically determining the length of the output summary based on gold-standard summaries observed during training. Xu and Durrett [42] use a sentence extraction model with a compression classifier that controls the deletion of syntax-derived compression options for each sentence. Among all these models, skillful extraction of fragments in the first step is necessary for high performance, considering that the second step is then built upon this extraction. While our system works without an external second step component, it could have an extended use as the first step of the two-step method for summarization.

3 OVERVIEW OF THE DOCTALK SYSTEM

We start with an overview of our system, including the main tools and functions of each module. Figure 1 summarizes the architecture of our system.

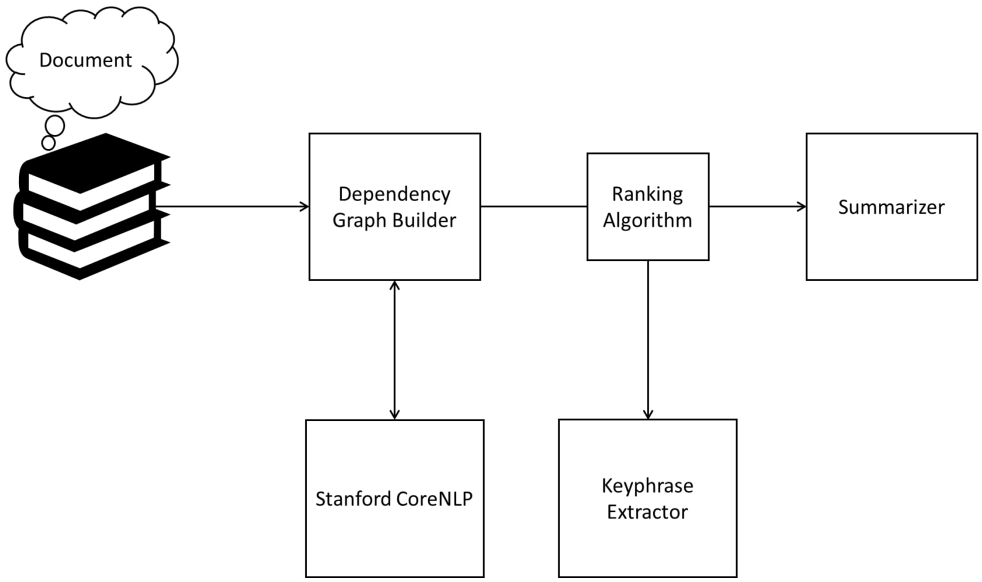

Figure 1: System Architecture

After a document is selected, we feed it to the dependency graph builder, which utilizes Stanford CoreNLP to extract dependency trees that it aggregates together into a document graph along the lines of Tarau and Blanco [36]. Using the information extracted from Stanford CoreNLP, we derive several types of edges and incorporate them into a graph representation of the text using Python's NetworkX library.

Besides dependency links between lemmas of the words in the text, the text graph also connects lemmas to the sentences in which they occur. By selecting the most salient sentences and keyphrases through PageRank or other centrality metrics, our document processor simultaneously supports summarization and keyphrase extraction.

Utilizing our text graph, we have also built an interactive query-answering module that returns extractive summaries specialized to the content of the queries (not discussed in this paper). In fact, the system presented in this paper is exposed as a web app in which a user uploads a document, views a summary and a set of keyphrases, and then interacts with a dialogue agent that supports in-depth exploration of the document in real time.

3.1 Creating the Text Graph

In order to represent the main idea of a document, we focus on the entities present in the document and their influence on each other by extracting *SVO* (short for subject-verb-object) edges and *has_instance* edges.

Each *SVO* edge connects two nodes that represent a subject and an object, respectively, with a verb, all in lemmatized form. An example would be *senate + have + power* in Figure 2.

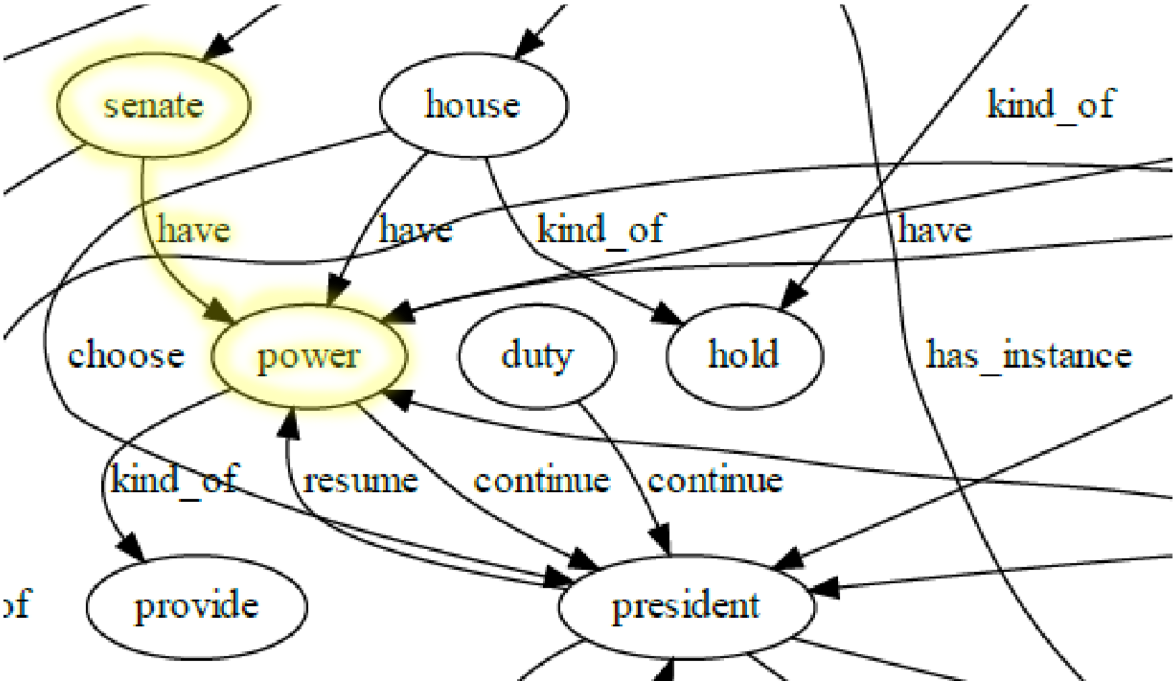

Figure 2: Part of the dependency graph for the Constitution of the United States

With the intuition that nouns are more effective for both summarization and keyphrase extraction, we orient the text graph around entities (nouns) by recognizing nodes that are named entities and identifying them with *has_instance* edges. A *has_instance* edge connects a node representing an entity to another node representing the category of the entity. An example is *organization + has_instance + senate* in Figure 2. Through the inclusion of *has_instance* edges, we add weight to nodes representing named entities, increasing their influence over the keyphrase extraction and summarization process.

3.2 Expanding the Text Graph using Syntactic Relations

To incorporate relational edges that are not syntactically inferable from the dependency trees provided by Stanford CoreNLP, we extract WordNet relations between lemmas and include them as edges in the text graph. For every token that is not a stop word, we utilize the *NLTK* toolkit's WordNet package to find its *synonyms* (words with similar meaning to the token), *hypernyms* (words that represent a broader category the token falls under), and *meronyms* (words that represent a part of the token but are used to refer to the whole). If the related word is in the same sentence as the original token, we organize them as follows:

synonym relations into the form of *original token + is_like + synonym*

hypernym relations into the form of *original token + kind_of + hypernym*

meronym relations into the form of *original token + part_of + meronym*.

Some of these relations are shown in Figure 2. As a result of expanding the text graph with edges reflecting semantic relations between words, the text graph now contains both entity-oriented edges and relational edges.

3.3 Summarizer

Once the text graph is generated, a ranking algorithm from NetworkX is applied to its nodes. Although many different ranking algorithms could be used, PageRank is used in this paper. However, this choice tends to over-prioritize long sentences, since the ranking value of sentences with more words, and thus more connections, will be higher. To prevent short and medium-sized sentences from being overlooked, a post-ranking normalization (1) is applied to the ranking value, which reduces the rank of sentences whose length (L) is much longer than the document's average sentence length (L_{avg}). After normalization, the highest ranked sentence nodes are used for the summary.

$$ranking\_value = \frac{ranking\_value}{1+|L-L_{avg}|+L} \quad (1)$$

3.4 Keyphrase Extractor

As nodes in the dependency graph have links pointed to them from their dependents, those that occur both as nouns and verbs in the text are usually the highest ranked due to having the most connections to other nodes. With the intuition that keyphrases should be entities central to a document (and therefore occur mostly as nouns), we only count a lemma as a potential keyphrase if more than half of its occurrences in the text are tagged as nouns.

Moreover, we prioritize compounds, with the intuition that compounds reveal more detailed information than single words. After the first round of lemma filtering, we collect the compound nodes adjacent to each lemma (connected to each original lemma through an *as_in* edge). If the ranking value of a compound node is sufficiently high compared to that of the original lemma, we choose the compound, instead of the single-word lemma, to be a keyphrase.

4 EMPIRICAL EVALUATION

We evaluate our system's summarization and keyphrase generation capabilities separately using the ROUGE metric [17]. The pyrouge package is used for evaluation.

4.1 Keyphrase Extraction

Similarly to Ye and Wang [43] and Chen et al. [4], we test our model on Krapivin [15], NUS [28], Inspec [13], SemEval-2010 Task 5 [14], and KP20K [22].

The DocTalk system's performance is compared with previous works, as shown in Table 1. The performance of previous works is obtained from Ye and Wang [43].

Table 1: Result of keyphrase extraction with metrics F1@5 and F1@10. The best results in each category are bold.

Model	Krapivin		Inspec		SemEval		NUS		KP20K	
	F1@5	F1@10	F1@5	F1@10	F1@5	F1@10	F1@5	F1@10	F1@5	F1@10
Algorithm										
DocTalk (this work)	**0.247**	**0.252**	**0.373**	**0.466**	**0.180**	**0.266**	**0.289**	**0.294**	**0.272**	**0.283**
TF-IDF	0.113	0.143	0.223	0.304	0.120	0.184	0.139	0.181	0.087	0.113
TextRank [24]	0.173	0.147	0.229	0.275	0.172	0.181	0.195	0.190	0.151	0.132
SingleRank [38]	0.096	0.137	0.214	0.297	0.132	0.169	0.145	0.169	0.099	0.124
ExpandRank [38]	0.096	0.136	0.211	0.295	0.135	0.163	0.137	0.162	-	-
Neural										
Wu et al. [40]	-	-	0.260	-	0.329	-	**0.412**	-	0.351	-

Model	Krapivin		Inspec		SemEval		NUS		KP20K	
	F1@5	F1@10	F1@5	F1@10	F1@5	F1@10	F1@5	F1@10	F1@5	F1@10
CopyRNN-GATER [44]	-	-	-	-	**0.366**	**0.340**	0.374	0.304	**0.401**	**0.324**
ExHiRD-h [5]	0.286	-	0.253	-	0.284	-	-	-	0.311	-
SIFRank [35]	-	-	0.291	**0.388**	0.226	0.329	-	-	-	-
SGG [46]	0.288	0.253	**0.306**	0.359	0.338	0.336	0.363	**0.358**	-	-
Seq2Seq-Copy [43]	0.274	0.207	0.269	0.234	0.278	0.226	0.345	0.282	-	-
Maui [20]	0.243	0.208	0.041	0.033	0.045	0.039	0.249	0.261	0.223	0.204
KEA [39]	0.120	0.131	0.109	0.129	0.027	0.027	0.068	0.081	-	-
RNN [22]	0.135	0.088	0.085	0.064	0.157	0.124	0.169	0.127	-	-
CopyRNN [22]	**0.311**	**0.266**	0.278	0.342	0.293	0.304	0.334	0.326	0.306	0.273

Across the five datasets, our system maintains high performance on keyphrase extraction from scientific documents, surpassing all unsupervised models presented, including TF-IDF and TextRank, on all datasets.

Compared to most supervised models, our model's performance is fairly close. Compared to the supervised model CopyRNN, our model's performance is slightly lower on Krapivin, SemEval, and NUS, comparable on KP20K, and higher on Inspec. In general, our model's performance is comparable to that of supervised models that are not state-of-the-art models specifically trained for datasets containing scientific documents.

4.2 Summarization

Due to our system's focus on scientific documents, we test the system on two summarization datasets consisting of research papers: arXiv and PubMed [8].

The average document length of PubMed and arXiv is significantly longer than that of other datasets. As a result, some neural network-based models, such as Pilault et al. [30], would require pre-processing to circumvent complexity due to the size of these long documents.

Our model was run on the test sets of both datasets, without pre-processing the documents. Our system's performance and the performance of previous models on the PubMed dataset and arXiv dataset are shown in Table 2. The test statistics for other models are obtained from Xiao and Carenini [41] and Pilault et al. [30].

Table 2: Result of summarization on PubMed and arXiv datasets. The best results in each category are bold.

Model	arXiv			PubMed		
	R-1	R-2	R-L	R-1	R-2	R-L
Algorithmic						
SumBasic [37]	29.47	6.95	26.30	37.15	11.36	33.43
LSA [34]	29.91	7.42	25.67	33.89	9.93	29.70
LexRank [9]	33.85	**10.73**	28.99	**39.19**	**13.89**	34.59
DocTalk	**38.19**	10.12	**40.54**	38.43	11.39	**41.16**
Neural						
Attn-Seq2Seq [26]	29.30	6.00	25.56	31.55	8.52	27.38
Cheng & Lapata [7]	42.24	15.97	27.88	43.89	18.53	30.17
Pntr-Gen-Seq2Seq [32]	32.06	9.04	25.16	35.86	10.22	29.69
SummaRuNNer [25]	42.81	16.52	28.23	43.89	18.78	30.36
Discourse [8]	35.80	11.05	31.80	38.93	15.37	35.21

Model	arXiv			PubMed		
	R-1	R-2	R-L	R-1	R-2	R-L
attentive context [41]	43.58	17.37	29.30	44.81	19.74	31.48
concat [41]	43.62	17.36	29.14	44.85	19.70	31.43
Sent-PTR [30]	42.32	15.63	38.06	43.30	17.92	39.47
TLM-I [30]	39.65	12.15	35.76	37.06	11.69	34.27
TLM-I+E (G,M) [30]	41.62	14.69	38.03	42.13	16.27	39.21
LoBART(4k)+MCS [19]	**48.79**	**20.55**	**43.31**	**48.06**	**20.96**	**43.56**
Nguyen & Luu & Lu & Quan [27]	44.53	19.22	40.61	45.99	20.49	41.25
Oracle						
Oracle [41]	53.88	23.05	34.90	55.05	27.48	38.66

On the PubMed dataset and arXiv dataset, our model's ROUGE-1 and ROUGE-2 scores are comparable to other unsupervised algorithmic models and slightly lower than recent neural-network based models. However, our ROUGE-L score is significantly higher than all other algorithmic models, indicating high sentence level structure similarity. Compared to the state-of-the-art, LoBART(4k)+MCS [19], our ROUGE-L score differs by less than 3.

5 CONCLUSION

By building a unified dependency graph containing sentence identifiers and lemmas, our graph-based approach covers both summarization and keyphrase extraction. Since our system is graph based, it does not require training, is real time, and is adaptive to many different types of documents, including research papers and technical, medical, and legal documents. We have shown competitive performance of our implementation on several publicly available datasets for summarization and keyphrase extraction.

For future work, we intend to explore the utilization of transformer language models in combination with our system, since work such as Pilault et al. [30] has demonstrated high performance using this method. In particular, we plan to first build a variant of our system that retrieves information from long documents and creates good sentence-level structure for summarization, as we did with the PubMed and arXiv datasets. Then, we would abstractively create a summary from the extracted information using a neural network-based approach, while preserving the sentence-level coherence established in the extraction step.

REFERENCES

[1] Sanghwan Bae, Taeuk Kim, Jihoon Kim, and Sang-goo Lee. 2019. Summary level training of sentence rewriting for abstractive summarization. In Proceedings of the 2nd Workshop on New Frontiers in Summarization, pages 10–20, Hong Kong, China. Association for Computational Linguistics.

[2] Adrien Bougouin, Florian Boudin, and Béatrice Daille. 2013. TopicRank: Graph-based topic ranking for keyphrase extraction. In Proceedings of the Sixth International Joint Conference on Natural Language Processing, pages 543–551, Nagoya, Japan. Asian Federation of Natural Language Processing.

[3] Asli Celikyilmaz, Antoine Bosselut, Xiaodong He, and Yejin Choi. 2018. Deep communicating agents for abstractive summarization. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pages 1662–1675, New Orleans, Louisiana. Association for Computational Linguistics.

[4] Jun Chen, Xiaoming Zhang, Yu Wu, Zhao Yan, and Zhoujun Li. 2018. Keyphrase generation with correlation constraints. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4057–4066, Brussels, Belgium. Association for Computational Linguistics.

[5] Wang Chen, Hou Pong Chan, Piji Li, and Irwin King. 2020. Exclusive hierarchical decoding for deep keyphrase generation. *arXiv preprint arXiv:2004.08511.*

[6] Yen-Chun Chen and Mohit Bansal. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 675–686, Melbourne, Australia. Association for Computational Linguistics.

[7] Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 484–494, Berlin, Germany. Association for Computational Linguistics.

[8] Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. 2018. A discourse-aware attention model for abstractive summarization of long documents. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers), pages 615–621, New Orleans, Louisiana. Association for Computational Linguistics.

[9] Gunes Erkan and Dragomir R. Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. Journal of Artificial Intelligence Research, 22:457–479.

[10] Dimitrios Galanis and Ion Androutsopoulos. 2010. An extractive supervised two-stage method for sentence compression. In Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, pages 885–893.

[11] Maria Grineva, Maxim Grinev, and Dmitry Lizorkin. 2009. Extracting key terms from noisy and multitheme documents. In Proceedings of the 18th international conference on World wide web, pages 661–670.

[12] Luyang Huang, Shuyang Cao, Nikolaus Parulian, Heng Ji, and Lu Wang. 2021. Efficient attentions for long document summarization.

[13] Anette Hulth. 2003. Improved automatic keyword extraction given more linguistic knowledge. In Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing, pages 216–223.

[14] Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. 2010. Semeval-2010 task 5: Automatic keyphrase extraction from scientific articles. In Proceedings of the 5th International Workshop on Semantic Evaluation, pages 21–26.

[15] Mikalai Krapivin, Aliaksandr Autaeu, and Maurizio Marchese. 2009. Large dataset for keyphrases extraction.

[16] Logan Lebanoff, Kaiqiang Song, Franck Dernoncourt, Doo Soon Kim, Seokhwan Kim, Walter Chang, and Fei Liu. 2019. Scoring sentence singletons and pairs for abstractive summarization. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 2175–2189, Florence, Italy. Association for Computational Linguistics.

[17] Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In Text Summarization Branches Out, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

[18] Zhiyuan Liu,Wenyi Huang, Yabin Zheng, and Maosong Sun. 2010. Automatic keyphrase extraction via topic decomposition. In Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 366–376, Cambridge, MA. Association for Computational Linguistics.

[19] Potsawee Manakul and Mark JF Gales. 2021. Long- span summarization via local attention and content selection. *arXiv preprint arXiv:2105.03801.*

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

[21] Afonso Mendes, Shashi Narayan, Sebastião Miranda, Zita Marinho, André F. T. Martins, and Shay B. Cohen. 2019. Jointly extracting and compressing documents with summary state representations. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 3955–3966, Minneapolis, Minnesota Association for Computational Linguistics.

[22] Rui Meng, Sanqiang Zhao, Shuguang Han, Daqing He, Peter Brusilovsky, and Yu Chi. 2017. Deep keyphrase generation. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 582–592, Vancouver, Canada. Association for Computational Linguistics.

[23] Rada Mihalcea and Dragomir Radev. 2011. Graph based natural language processing and information retrieval. Cambridge University Press.

[24] Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into text. In Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing, pages 404–411, Barcelona, Spain. Association for Computational Linguistics.

[25] Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, AAAI'17, page 3075–3081. AAAI Press.

[26] Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çaglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using

sequence-to sequence RNNs and beyond. In Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning, pages 280–290, Berlin, Germany. Association for Computational Linguistics.

[27] Thong Nguyen, Anh Tuan Luu, Truc Lu, and Tho Quan. 2021. Enriching and controlling global semantics for text summarization. *arXiv preprint arXiv:2109.10616.*

[28] Thuy Dung Nguyen and Min-Yen Kan. 2007. Keyphrase extraction in scientific publications. In Proceedings of the 10th International Conference on Asian Digital Libraries: Looking Back 10 Years and Forging New Frontiers, ICADL'07, page 317–326, Berlin, Heidelberg. Springer-Verlag.

[29] Lawrence Page and Sergey Brin. 1998. The anatomy of a large-scale hypertextual web search engine. Computer Networks and ISDN Systems, 30:107—-117.

[30] Jonathan Pilault, Raymond Li, Sandeep Subramanian, and Chris Pal. 2020. On extractive and abstractive neural document summarization with transformer language models. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 9308–9319, Online. Association for Computational Linguistics.

[31] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683.*

[32] Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

[33] Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

[34] Josef Steinberger, Karel Jezek, et al. 2004. Using latent semantic analysis in text summarization and summary evaluation. Proc. ISIM, 4:93–100.

[35] Yi Sun, Hangping Qiu, Yu Zheng, Zhongwei Wang, and Chaoran Zhang. 2020. Sifrank: A new baseline for unsupervised keyphrase extraction based on pre-trained language model. IEEE Access, 8:10896– 10906.

[36] Paul Tarau and Eduardo Blanco. 2020. Interactive Text Graph Mining with a Prolog-Based Dialog Engine. Theory and Practice of Logic Programming, pages 1–20.

[37] Lucy Vanderwende, Hisami Suzuki, Chris Brockett, and Ani Nenkova. 2007. Beyond sumbasic: Task-focused summarization with sentence simplification and lexical expansion. Information Processing & Management, 43(6):1606–1618.

[38] Xiaojun Wan and Jianguo Xiao. 2008. Single document keyphrase extraction using neighborhood knowledge. In AAAI, volume 8, pages 855–860.

[39] Ian H. Witten, Gordon W. Paynter, Eibe Frank, Carl Gutwin, and Craig G. Nevill-Manning. 1999. KEA: practical automatic keyphrase extraction. CoRR, cs.DL/9902007.

[40] Huanqin Wu, Baijiaxin Ma, Wei Liu, Tao Chen, and Dan Nie. 2022. Fast and constrained absent keyphrase generation by prompt-based learning.

[41] Wen Xiao and Giuseppe Carenini. 2019. Extractive summarization of long documents by combining global and local context. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 3011–3021, Hong Kong, China. Association for Computational Linguistics.

[42] Jiacheng Xu and Greg Durrett. 2019. Neural extractive text summarization with syntactic compression. *arXiv preprint arXiv:1902.00863.*

[43] Hai Ye and Lu Wang. 2018. Semi-supervised learning for neural keyphrase generation. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4142–4153, Brussels, Belgium. Association for Computational Linguistics.

[44] Jiacheng Ye, Ruijian Cai, Tao Gui, and Qi Zhang. 2021. Heterogeneous graph neural networks for keyphrase generation. arXiv preprint *arXiv:2109.04703.*

[45] Fan Zhang, Lian'en Huang, and Bo Peng. 2013. WordTopic-MultiRank: A new method for automatic keyphrase extraction. In Proceedings of the Sixth International Joint Conference on Natural Language Processing, pages 10–18, Nagoya, Japan. Asian Federation of Natural Language Processing.

[46] Jing Zhao, Junwei Bao, Yifan Wang, Youzheng Wu, Xiaodong He, and Bowen Zhou. 2021. Sgg: learning to select, guide, and generate for keyphrase generation. *arXiv preprint arXiv:2105.02544.*