Web Document Encoding for Structure-Aware Keyphrase Extraction

Jihyuk Kim
Yonsei University
Seoul, Republic of Korea
jihyukkim@yonsei.ac.kr

Young-In Song
Naver Corp
Gyeonggi-do, Republic of Korea
song.youngin@navercorp.com

Seung-won Hwang
Seoul National University
Seoul, Republic of Korea
seungwonh@snu.ac.kr

ABSTRACT

We study keyphrase extraction (**KPE**) from Web documents. Our key contribution is encoding Web documents to leverage **structure**, such as title or anchors, by building a graph of words representing both (a) position-based proximity and (b) structural relations. We evaluate KPE performance on real-world search engine NAVER and human-annotated KPE benchmarks, and ours outperforms state-of-the-arts in both tasks.

CCS CONCEPTS

• **Information systems** → **Information extraction**.

KEYWORDS

keyphrase extraction, search query extraction, structured Web document, graph convolutional network

ACM Reference Format:
Jihyuk Kim, Young-In Song, and Seung-won Hwang. 2021. Web Document Encoding for Structure-Aware Keyphrase Extraction. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '21), July 11–15, 2021, Virtual Event, Canada.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3404835.3463067

1 INTRODUCTION

In Web search, keyphrase extraction (**KPE**) from documents can contribute to enhancing (a) document ranking and (b) human readability. To illustrate, early models [12] extract key n-grams in the document as likely queries, to improve ranking for such queries. Alternatively, keyphrases can be highlighted for human readability.

For the **first task**, KPE is supervised by click queries and aims to improve ranking. This task can be intrinsically evaluated by the quality of predicted queries, and extrinsically by the quality of ranking, after the Web document is augmented with keyphrases. We evaluate this task on real-life search engine NAVER in Korean.

For the **second task**, KPE is supervised by human-annotated keyphrases from OpenKP [32], on randomly sampled English documents from the Web. This dataset enables a visual representation of a document by including information such as location, font size, and HTML structure, but does not include search queries and relevance assessments, and as such, this task is only intrinsically evaluated.

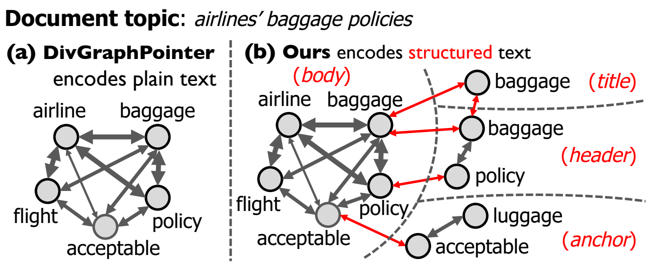

Figure 1: An illustrative example document describing airlines' baggage policies. The thickness of black edges indicates a higher degree of position-based term proximity. The field structure enriches the representation, through inter-field connection, shown in red edges.

Our proposed model is built upon the recently proposed KPE model, DivGraphPointer [28], treating document as plain text and building a word graph, where edges between words (nodes) are represented using position-based proximity. DivGraphPointer is a state-of-the-art KPE for **plain text**, outperforming sequential encoding models such as CopyRNN [21].

Our distinction is proposing an encoder for **Web documents** which takes into account useful structures such as *title*, *body*, and *anchor*. Figure 1 contrasts DivGraphPointer considering only term proximity on a plain text (black edges), with ours considering field structures of diverse importance, by adding inter-field relations (red edges). Representing structure in encoding Web documents is essential, as each field contributes uniquely to KPE: *title* includes high-precision matches, *body* includes high-recall words, and *anchor*, linking from an external source document, connects to "absent" words that may not exist in the document. For example, in Figure 1(b), title captures the high-precision word "baggage" and anchor adds an absent word "luggage" to representation, contributing to answering future queries with vocabulary mismatches.

Our empirical results validate that our model effectively exploits structure information in Web documents, outperforming both baselines with and without consideration of structures.

2 RELATED WORK

This section overviews existing work on KPE and presents our distinction. Initial KPE models focus on sequential encoding of a document [33], which can be implemented by LSTM [14] or GRU [5]. Recently, neural graph-based models [11, 25, 28], that extend traditional graph-based keyword ranking [2, 8, 22], outperformed sequential models, by using co-occurrence or proximity between words as edge weights for modeling relations between them. Our baseline DivGraphPointer [28] using Graph Convolutional Network (GCN) [18] falls into this category, focusing on encoding plain text.

For representing structures in documents, modeling internal and external structures of the text are essential sources of prediction signals. Internal structures considered for KPE include section title [4] as a representation of topics, syntactic structures of documents such as POS-tags [34], discourse relations [15], or dependency parse tree [29], or visual features in Web documents such as font size or location of text chunks [32]. External structures include citation network [3, 9, 26] of scientific papers, hyperlinks connecting documents with anchor text [6, 20], or similar documents that can expand keyword vocabulary and contexts [30, 31].

Our distinction: We propose a graph-based encoder with structure-awareness, representing both external and internal structures, to significantly outperform KPE state-of-the-arts.

3 APPROACH

Given a Web document $\mathbf{X}$, keyphrase prediction task aims to predict gold keyphrase $\mathbf{Y}$ where $\mathbf{Y}$ should sufficiently contain the topical information of $\mathbf{X}$ with only a few words. We leverage structure information in Web documents: $\mathbf{X}$ consists of multiple field contents $\{\mathbf{X}^u\}_{u \in F}$ where F is a set of fields. In §3.1, we first present our baseline, treating $\mathbf{X}$ as plain text that entirely belongs to a field u (*i.e.*, $F = \{u\}$ and $\mathbf{X} = \mathbf{X}^u$), then, in §3.2, present our distinctions.

3.1 Baseline: Plain Text

Targeting plain texts that consist of a single field, *i.e.*, $\mathbf{X} = \mathbf{X}^u$, this section focuses on modeling intra-field relation, *i.e.*, relation between words within a field, describing Graph Convolutional Network (GCN) [18] for KPE, used in DivGraphPointer [28].

A fully connected graph for $\mathbf{X}^u$ is first constructed: Nodes are words and edges reflect the relatedness between the words, using position-based proximity between the words. Given the graph, the GCN layer contextualizes word representations, considering the relatedness between words. To iteratively propagate contexts, starting from uncontextualized word representations obtained from a word embedding matrix, we stack L GCN layers. We denote the word representations for each layer by $\mathbf{H}^u_l \in \mathbb{R}^{N_u \times D}$, where $l \in [1, L]$, N_u, and D denote the index of the contextualization layer, the number of unique words in $\mathbf{X}^u$, and the number of features for each node representation, respectively.

To model intra-field relation, two adjacency matrices, $\overleftarrow{\mathbf{A}}^u$ and $\overrightarrow{\mathbf{A}}^u \in \mathbb{R}^{N_u \times N_u}$ for forward and backward direction respectively, are obtained using position-based proximity:

$$\overleftarrow{\mathbf{A}}^u_{ij} = \sum_{p^u_i \in \mathcal{P}(x^u_i)} \sum_{p^u_j \in \mathcal{P}(x^u_j)} \max((p^u_i - p^u_j)^{-1}, 0) \quad (1)$$

$$\overrightarrow{\mathbf{A}}^u_{ij} = \sum_{p^u_i \in \mathcal{P}(x^u_i)} \sum_{p^u_j \in \mathcal{P}(x^u_j)} \max((p^u_j - p^u_i)^{-1}, 0), \quad (2)$$

where $p^u_{[i/j]} \in \mathcal{P}(x^u_{[i/j]})$ is a position offset of a unique word $x^u_{[i/j]}$ in $\mathbf{X}^u$. Note, a word that appears multiple times sums up its proximity through multiple occurrences, and is naturally emphasized. To stabilize training, following the convention of [18], we normalize $\mathbf{A}^u \in \{\overleftarrow{\mathbf{A}}^u, \overrightarrow{\mathbf{A}}^u\}$ to $\hat{\mathbf{A}}^u$ with eigenvalues close to 1, from which intra-field context vectors $\mathbf{C}^{\text{intra}}_{l,u} \in \mathbb{R}^{N_u \times D}$ can be obtained using graph convolution [18, 28]:

$$\mathbf{C}^{\text{intra}}_{l,u} = \hat{\overleftarrow{\mathbf{A}}}^u \mathbf{H}^u_l \overleftarrow{\mathbf{W}}^u_l + \hat{\overrightarrow{\mathbf{A}}}^u \mathbf{H}^u_l \overrightarrow{\mathbf{W}}^u_l + \mathbf{H}^u_l \mathbf{W}^u_l, \quad (3)$$

where $\overleftarrow{\mathbf{W}}^u_l, \overrightarrow{\mathbf{W}}^u_l, \mathbf{W}^u_l \in \mathbb{R}^{D \times D}$ are learnable matrices.

3.2 Proposed: Inter-field Relation

Our proposed approach decomposes $\mathbf{X}$ into multiple field contents, $\{\mathbf{X}^u\}_{u \in F}$ where F is a set of fields. Our goal in leveraging structure information is to model different language characteristics between fields, and to model inter-field relation, enabling a complementary interaction between the contents of different fields. For example, a short title with high precision often consists of a few topic-indicative words while a long body with high recall additionally includes functional words (*e.g.*, conjunctions or pronouns). Thus, by modeling the interaction between the two fields, we can achieve a well-balanced trade-off between precision and recall.

To tackle the heterogeneity of language characteristics, the baseline that models intra-field relation can be extended with dedicated matrices $\overleftarrow{\mathbf{W}}^u_l, \overrightarrow{\mathbf{W}}^u_l, \mathbf{W}^u_l$ for each field $u \in F$.

To model inter-field relation, for each field $s \in F/\{u\}$, we use words that appear in both u and s in common, as mutual context, to propagate context from s to u. Specifically, we build an adjacency matrix between the two fields $\mathbf{M}^{s \to u} \in \mathbb{R}^{N_u \times N_s}$ where for each word in field u and s, $\forall x^u_i \in \mathbf{X}^u, \forall x^s_k \in \mathbf{X}^s$, $\mathbf{M}^{s \to u}_{ik} = 1$ if $x^u_i = x^s_k$, and 0 otherwise. Similar to normalizing $\mathbf{A}^u$ (§3.1), we normalize $\mathbf{M}^{s \to u}$ to $\hat{\mathbf{M}}^{s \to u} = \frac{\mathbf{M}^{s \to u}_{ik}}{\sum_{\forall s' \in F/\{u\}, \forall k' \in [1, N_{s'}]} \mathbb{I}(x^u_i = x^{s'}_{k'})}$ [1], from which we can obtain inter-field context vectors $\mathbf{C}^{\text{inter}}_{l,u} \in \mathbb{R}^{N_u \times D}$ by

$$\mathbf{C}^{\text{inter}}_{l,u} = \sum_{\forall s \in F/\{u\}} \hat{\mathbf{M}}^{s \to u} \mathbf{H}^s_l \mathbf{W}^{s \to u}_l, \quad (4)$$

where $\mathbf{W}^{s \to u}_l \in \mathbb{R}^{D \times D}$ is a learnable matrix for feature transformation from field s to u.

After aggregating intra- and inter-field contexts by Eq (3, 4), node representations are updated using residual addition between the previous node representations and the aggregated contexts [7]:

$$\mathbf{H}^u_{l+1} = \mathbf{H}^u_l + \left(\mathbf{C}^{\text{intra}}_{l,u} + \mathbf{C}^{\text{inter}}_{l,u}\right) \otimes \sigma\left(\mathbf{G}^{\text{intra}}_{l,u} + \mathbf{G}^{\text{inter}}_{l,u}\right), \quad (5)$$

where $\mathbf{G}^{\text{intra}}_{l,u}$ and $\mathbf{G}^{\text{inter}}_{l,u}$ are obtained in the same way to Eq (3, 4) respectively but with different parameters, and $\otimes$, σ denote element-wise product and sigmoid function respectively. Such residual connection enables effective gradient back-propagation with deep layers [7, 13]. For baseline, $\mathbf{H}^u_l$ is updated without $\mathbf{C}^{\text{inter}}_{l,u}$ and $\mathbf{G}^{\text{inter}}_{l,u}$.

Finally, contextualized word vectors in the last GCN layer for all fields, $\{\mathbf{H}^u_L\}_{u \in F}$, are fed into the decoder to extract keyphrases.

3.3 Shared: Vanilla Decoder

Given contextualized word vectors from encoder, $\{\mathbf{H}^u_L\}_{u \in F}$, the goal of decoder is to extract each keyword, y_t, in the gold keyphrase $\mathbf{Y} = [y_1, \ldots, y_T]$ within the context of the given document. As our focus in this paper is encoding, we do not optimize for the decoder,

[1] We normalize $\mathbf{M}^{s \to u}$ row-by-row, since each row has different numbers of non-zero entries, unlike $\overleftarrow{\mathbf{A}}^u$ and $\overrightarrow{\mathbf{A}}^u$ which are dense matrices without any zero entries.

and share a simple decoder [28] in common: A single layer GRU [5] decoder sequentially copies keywords from the given document.

$$\mathbf{s}_t = \mathrm{GRU}(\mathbf{y}_{t-1}, \mathbf{s}_{t-1}) \tag{6}$$

$$p(y_t|y_1, \dots, y_{t-1}, \mathbf{X}) = \phi(\mathbf{s}_t, \{\mathbf{H}_L^u\}_{u\in F}), \tag{7}$$

where $\mathbf{s}_t \in \mathbb{R}^D$ is a hidden state of the decoder at t-th decoding step, $\mathbf{y}_{t-1}$ is a word vector for previously extracted keyword, and ϕ is a function computing **copy weights**, a distribution of copy probability for words in $\mathbf{X}$, in place of y_t.

To leverage structure information in Web documents, such computation should consider field-specific frequency, for which we make the following changes to vanilla decoder. First, word appearance in some field can be learned to be emphasized, *e.g.*, words in the anchor field are often more likely to be keywords than words in the body field [6, 20]. Note that, for each field u, we use dedicated parameters $\overleftarrow{\mathbf{W}}_l^u, \overrightarrow{\mathbf{W}}_l^u, \mathbf{W}_l^u$ for graph convolution (§3.2). Thus, we can inject field-type information into $\mathbf{C}_{l,u}^{\text{intra}}$ and accordingly into $\mathbf{H}_L^u$ (using Eq (3) and Eq (5) respectively), by which field importance can be considered. Specifically, to compute copy scores $\{z_t^{u,i}\}_{i=1}^{N_u}$ for words in a field u, we employ attention layer [1] with $\mathbf{H}_L^u \in \mathbb{R}^{N_u \times D}$:

$$z_t^{u,i} = \mathbf{v}^\top \tanh(\mathbf{Q}[\mathbf{s}_t; \mathbf{h}_i^u] + \mathbf{b}), \tag{8}$$

where $\mathbf{h}_i^u$ (the i-th row vector in $\mathbf{H}_L^u$) is the word vector for word x_i in the field u, infused with field type information, $[;]$ denotes the concatenation of the two input vectors, and $\mathbf{v} \in \mathbb{R}^D, \mathbf{Q} \in \mathbb{R}^{D\times 2D}, \mathbf{b} \in \mathbb{R}^D$ are learnable parameters.

Second, since overlapping terms that appear in different fields in common are highly likely to be a keyword (*e.g.*, in Figure 1(b), "baggage" in body, title and header), the decoder should emphasize those terms. Specifically, for each unique word in the given document, denoted by $\tilde{x}_j$, we accumulate the copy scores of $\tilde{x}_j$ across fields, producing a probability ϕ_j for the word $\tilde{x}_j$ as follows:

$$\phi_j(\mathbf{s}_t, \{\mathbf{H}_L^u\}_{u\in F}) = \frac{\sum_{\forall u \in F | x_i^u = \tilde{x}_j} \exp(z_t^{u,i})}{\sum_{\forall u' \in F} \sum_{k=1}^{N_{u'}} \exp(z_t^{u',k})}. \tag{9}$$

Finally, following the convention of [28], we use beam search to find word sequences with maximum probability during inference, where beam size is empirically tuned to 200. In addition, during training and inference, we add a special symbol at the end of keyphrases, for predicting the end of sequence. Parameters are optimized to maximize the likelihood of ground-truth keyphrases using cross entropy loss as objective.

4 EXPERIMENT

4.1 Dataset

As illustrated in §1, we have two tasks, represented by the following two datasets.

Query Prediction and Document Ranking: For the first task, we sampled 1.3M real-world Web documents and corresponding click queries using NAVER search engine. To denoise irrelevant clicks, we removed queries that cannot be extracted from the fields in the given document. We randomly sampled 15k documents for evaluation. Documents without click queries were excluded, finally producing 11k documents. We use 5k documents as a test set and the others as a validation set. As most of the documents and click queries are written in Korean, we denote this constructed dataset as KoWeb, short for **Ko**rean **Web** documents.

For structures in Web documents, we use title, body, and anchor text. Body is divided into body as a whole, body with large font sizes, body placed in borders, and h1 headers.

Human-annotated Keyphrases: For the second task, we evaluate models on human-annotated keyphrases using public benchmarks in English, OpenKP [32]. 134K, 6K, and 6K documents are used for train, validation, and test evaluation respectively.

Using visual features on text segments given in the dataset, we structuralized documents into body text as a whole, large font body, and h1 headers. Titles are not provided in this dataset.

Statistics for both KoWeb and OpenKP are presented in Table 1.

Statistics	KoWeb	OpenKP
# of document	1.3M	148K
# of unique CQ/Kph	2.9M	99.6K
Average # of CQ/Kph per document	2.03	1.8
Average document length	263.4	900.4
Average CQ/Kph length	3.13	2.0

Table 1: Statistics on KoWeb and OpenKP. CQ and Kph are short for click query and keyphrase respectively.

4.2 Implementation Detail

For efficient training, we share word embedding matrices of encoders for all fields as well as for decoder. We used top-100k and top-50k frequent words as vocabularies of both encoder and decoder, for KoWeb and OpenKP respectively. We train models by stochastic gradient descent method using Adam optimizer [17] with learning rate 1e-4, l_2 constraint [23] of 1, and batch size 64. We apply early stopping based on perplexity on validation set. We search the best value for D between $\{128, 256\}$ and L among $\{3, 4, 5\}$ based on validation performance. The best value for D was 128 and the best value for L was 3 and 4 for KoWeb and OpenKP respectively. Our implementation was based on Pytorch [24] and OpenNMT [19].

4.3 Baselines

To validate the potential of leveraging structure information in Web documents, we compare our model with models that do not use structure information such as CopyRNN [21] and GraphPointer (proposed in DivGraphPointer [28]). CopyRNN represents a document as a word sequence, and encodes the sequence using GRU [5]. GraphPointer represents a document as a fully connected graph by using proximity between words as edge weights, and then GCN [18] is used to encode the graph (§3.1). For the two baselines, documents are given as concatenation of title and body for KoWeb dataset[2]. For OpenKP, we only use body, as the titles are not provided.

To validate the effectiveness of our design choice on structured Web documents, we also use TGNet [4] as a baseline that uses a simple structure consisting of title and body. Since titles are not provided in OpenKP, we use h1 headers instead.

Recall from §3.3 that all implementations share our vanilla decoder for fairness. Therefore, we distinctively denote the models

[2]For fair comparisons with baselines, we filtered out click queries that cannot be extracted by a combination of body and title fields, for both training and evaluation.

by their encoders: SeqEnc, GraphEnc, TGEnc, and MFGraphEnc (short for Multi-Field Graph Encoder) correspond to encoders used in CopyRNN, GraphPointer, TGNet, and our encoder respectively.

4.4 KoWEB

We first evaluate models supervised by click query using KoWeb.

4.4.1 Intrinsic: Query Prediction. For evaluation metrics, we use macro-averaged precision, recall, and F1 scores, used as standard performance metrics on keyphrase prediction task. Given the top-k predictions by models, we first count the number of correct predictions (denoted by hit@k), then precision and recall are computed as hit@k divided by k and hit@k divided by the number of target click queries respectively. We report F1@1/3/5 in Table 2.

Model	F1@1	F1@3	F1@5
SeqEnc	0.178	0.164	0.135
TGEnc	0.176	0.167	0.137
GraphEnc	0.185	0.157	0.128
MFGraphEnc (ours)	**0.219**	**0.181**	**0.146**

Table 2: KoWeb results: our proposed model outperforms all baselines with statistical significance[3].

SeqEnc shows comparable performances with TGEnc, indicating that titles in Web documents are noisy and thus do not provide effective guidance for encoding. Meanwhile, by using headers and anchors to complement noisy title, ours outperforms all baselines.

4.4.2 Extrinsic: Ranking. We perform extrinsic evaluation using document retrieval task [10]. For evaluation, we constructed a new dataset as follows: First, we randomly sampled 1k search queries from NAVER search log and removed 104 queries which have vague search intents. Then, top-100 ranks from NAVER search for each query were crawled as search candidates.

To reduce the significant time cost necessary to perform keyphrase extractions repeatedly, we sampled documents again from the candidate pool by selecting (1) about 10 documents randomly from low rank area of a search result (lower than top-10 ranks) as potentially irrelevant (or low relevance) candidates and (2) top-3 ranked documents in the search result as potentially-relevant ones. Given the document candidates for each query, we asked three expert annotators to assign relevance scores ranging from 1 to 5 between the query and the documents. For disagreed judgements, a senior assessor was requested to decide the labels. As a result, we collect about 10k unique relevance judged documents for 896 queries. To get realistic document frequency statistics for each term, we used additional 1M documents which were also randomly sampled from NAVER search engine. Note that any document used in ranking experiments is not included in KoWeb, where extraction models were trained, to evaluate generalization ability.

Given the dataset, we extracted keyphrases from each document by using keyphrase extraction models trained on KoWeb. The extracted keyphrases were attached to the document as a separate field to the existing fields in the documents. For document retrieval, we used BM25f [27] as a ranking function, one of the standard ranking approaches to consider field structure of Web documents.

[3]at $p < 0.05$ using Student's paired t-test.

Index	nDCG@1	@3	@5
Document	0.587	0.650	0.695
+ SeqEnc	0.629	0.658	0.705
+ TGEnc	0.621	0.661	0.701
+ GraphEnc	0.631	0.665	0.708
+ MFGraphEnc (ours)	**0.649**	**0.682**	**0.724**

Table 3: nDCG@1/3/5 on retrieval task (KoWeb): before/after document is expanded with extracted keyphrases[4].

80% of search queries in the dataset was used to tune BM25f parameters and 20% for evaluation. BM25f parameters were re-optimized again whenever a keyphrase extraction method was changed in the experiments. We report nDCG@k [16], using the ranked list of retrieved documents and relevance labels, in Table 3.

Overall, indexing documents with predicted click queries improves retrieval performances, validating the effectiveness of click queries as pseudo keyphrases on document retrieval. Among keyphrase extraction models, our proposed model using field structures achieves the largest performance increases.

4.5 OpenKP

Model	F1@1	F1@3	F1@5
SeqEnc	0.212	0.235	0.209
TGEnc	0.226	0.257	0.225
GraphEnc	0.207	0.229	0.200
BLING-KPE	0.244	0.277	0.198
MFGraphEnc (ours)	**0.273**	**0.284**	**0.244**

Table 4: OpenKP results: ours proposed model outperforms all baselines with statistical significance[5].

In addition to KoWeb, we also validate the effectiveness of our proposed model using a public/English dataset, OpenKP [32]. We add a new baseline BLING-KPE [32] that uses visual features of the document as privileged information in this set[6]. We report F1 scores of top-1/3/5 predictions as performance metrics, in Table 4.

In contrast to results on KoWeb dataset, TGEnc outperforms SeqEnc on F1 measures. In this dataset, headers were used as title for TGEnc, which contain 44.8% of ground-truth keyphrases (higher than 38.5% of title in KoWeb), which explains relatively higher performance. Meanwhile, our proposed model outperforms all baselines on all metrics.

5 CONCLUSION

We propose to encode structured Web documents consisting of heterogeneous fields, by leveraging both intra-/inter-field relations, integrated as a graph model, for KPE. In our experiments, our method outperforms state-of-the-arts in both real-life search logs and public benchmarks for KPE.

ACKNOWLEDGMENTS

This work was supported by NAVER-SQR program from NAVER corporation, and IITP funded by MSIT (No. 2017-0-01779, XAI).

[4]We tried varying k for the top-k keyphrases and empirically found $k = 1$ to strike the best balance regarding precision-recall trade-off.

[5]at $p < 0.001$ using Student's paired t-test.

[6]For BLING-KPE, we present performances that were reported in the MSMARCO leaderboard (https://microsoft.github.io/msmarco/). For fair comparisons, we report performances of BLING-KPE without pre-training on Web-scale click queries.

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *ICLR*. http://arxiv.org/abs/1409.0473

[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*. Asian Federation of Natural Language Processing, Nagoya, Japan, 543–551. https://www.aclweb.org/anthology/I13-1062

[3] Cornelia Caragea, Florin Bulgarov, Andreea Godea, and Sujatha Das Gollapalli. 2014. Citation-enhanced keyphrase extraction from research papers: A supervised approach. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 1435–1446.

[4] Wang Chen, Yifan Gao, Jiani Zhang, Irwin King, and Michael R Lyu. 2019. Title-Guided Encoding for Keyphrase Generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 6268–6275.

[5] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, 1724–1734. https://doi.org/10.3115/v1/D14-1179

[6] Van Dang and Bruce W Croft. 2010. Query reformulation using anchor text. In *Proceedings of the third ACM international conference on Web search and data mining*. 41–50.

[7] Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *International conference on machine learning*. 933–941.

[8] Corina Florescu and Cornelia Caragea. 2017. A position-biased pagerank algorithm for keyphrase extraction. In *Thirty-first AAAI conference on artificial intelligence*.

[9] Sujatha Das Gollapalli and Cornelia Caragea. 2014. Extracting Keyphrases from Research Papers Using Citation Networks.. In *AAAI*, Vol. 14. Citeseer, 1629–1635.

[10] Carl Gutwin, Gordon Paynter, Ian Witten, Craig Nevill-Manning, and Eibe Frank. 1999. Improving browsing in digital libraries with keyphrase indexes. *Decision Support Systems* 27, 1-2 (1999), 81–104.

[11] Fred.X Han, Di Niu, Kunfeng Lai, Weidong Guo, Yancheng He, and Yu Xu. 2019. Inferring Search Queries from Web Documents via a Graph-Augmented Sequence to Attention Network. In *The World Wide Web Conference* (San Francisco, CA, USA) *(WWW '19)*. Association for Computing Machinery, New York, NY, USA, 2792–2798. https://doi.org/10.1145/3308558.3313746

[12] Kazi Saidul Hasan and Vincent Ng. 2014. Automatic keyphrase extraction: A survey of the state of the art. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 1262–1273.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

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

[15] Tatsuya Ishigaki, Hidetaka Kamigaito, Hiroya Takamura, and Manabu Okumura. 2019. Discourse-Aware Hierarchical Attention Network for Extractive Single-Document Summarization. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*. INCOMA Ltd., Varna, Bulgaria, 497–506. https://doi.org/10.26615/978-954-452-056-4_059

[16] Kalervo Järvelin and Jaana Kekäläinen. 2017. IR Evaluation Methods for Retrieving Highly Relevant Documents. *SIGIR Forum* 51, 2 (Aug. 2017), 243–250. https://doi.org/10.1145/3130348.3130374

[17] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1412.6980

[18] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net. https://openreview.net/forum?id=SJU4ayYgl

[19] Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. OpenNMT: Open-Source Toolkit for Neural Machine Translation. In *Proc. ACL*. https://doi.org/10.18653/v1/P17-4012

[20] Reiner Kraft and Jason Zien. 2004. Mining anchor text for query refinement. In *Proceedings of the 13th international conference on World Wide Web*. 666–674.

[21] 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)*. Association for Computational Linguistics, Vancouver, Canada, 582–592. https://doi.org/10.18653/v1/P17-1054

[22] Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*. 404–411.

[23] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the Difficulty of Training Recurrent Neural Networks. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28* (Atlanta, GA, USA) *(ICML'13)*. JMLR.org, III–1310–III–1318.

[24] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc., 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[25] Animesh Prasad and Min-Yen Kan. 2019. Glocal: Incorporating Global Information in Local Convolution for Keyphrase Extraction. 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)*. Association for Computational Linguistics, Minneapolis, Minnesota, 1837–1846. https://doi.org/10.18653/v1/N19-1182

[26] Vahed Qazvinian, Dragomir Radev, and Arzucan Özgür. 2010. Citation summarization through keyphrase extraction. In *Proceedings of the 23rd international conference on computational linguistics (COLING 2010)*. 895–903.

[27] Stephen Robertson, Hugo Zaragoza, and Michael Taylor. 2004. Simple BM25 Extension to Multiple Weighted Fields. In *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management* (Washington, D.C., USA) *(CIKM '04)*. Association for Computing Machinery, New York, NY, USA, 42–49. https://doi.org/10.1145/1031171.1031181

[28] Zhiqing Sun, Jian Tang, Pan Du, Zhi-Hong Deng, and Jian-Yun Nie. 2019. Divgraphpointer: A graph pointer network for extracting diverse keyphrases. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 755–764.

[29] Paul Tarau and Eduardo Blanco. 2019. Dependency-based Text Graphs for Keyphrase and Summary Extraction with Applications to Interactive Content Retrieval. *arXiv preprint arXiv:1909.09742* (2019).

[30] Xiaojun Wan and Jianguo Xiao. 2008. CollabRank: Towards a Collaborative Approach to Single-Document Keyphrase Extraction. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*. Coling 2008 Organizing Committee, Manchester, UK, 969–976. https://www.aclweb.org/anthology/C08-1122

[31] Xiaojun Wan and Jianguo Xiao. 2008. Single Document Keyphrase Extraction Using Neighborhood Knowledge.. In *AAAI*, Vol. 8. 855–860.

[32] Lee Xiong, Chuan Hu, Chenyan Xiong, Daniel Campos, and Arnold Overwijk. 2019. Open Domain Web Keyphrase Extraction Beyond Language Modeling. 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)*. Association for Computational Linguistics, Hong Kong, China, 5175–5184. https://doi.org/10.18653/v1/D19-1521

[33] Qi Zhang, Yang Wang, Yeyun Gong, and Xuanjing Huang. 2016. Keyphrase Extraction Using Deep Recurrent Neural Networks on Twitter. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, 836–845. https://doi.org/10.18653/v1/D16-1080

[34] Jing Zhao and Yuxiang Zhang. 2019. Incorporating Linguistic Constraints into Keyphrase Generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Florence, Italy, 5224–5233. https://doi.org/10.18653/v1/P19-1515