Inferring Search Queries from Web Documents via a Graph-Augmented Sequence to Attention Network

Fred X. Han
University of Alberta
Edmonton, Canada
xuefei1@ualberta.ca

Di Niu
University of Alberta
Edmonton, Canada
dniu@ualberta.ca

Kunfeng Lai
Tencent
ShenZhen, China
calvinlai@tencent.com

Weidong Guo
Tencent
BeiJing, China
weidongguo@tencent.com

Yancheng He
Tencent
ShenZhen, China
collinhe@tencent.com

Yu Xu
Tencent
ShenZhen, China
henrysxu@tencent.com

ABSTRACT

We study the problem of search query inference from web documents, where a short, comprehensive natural language query is inferred from a long article. Search query generation or inference is of great value to search engines and recommenders in terms of locating potential target users and ranking content. Despite being closely related to other NLP tasks like abstract generation and keyword extraction, we point out that search query inference is, in fact, a new problem, in that the generated natural language query, which consists of a few words, is expected to be comprehensive enough to lead to the click-through of the corresponding document. Therefore, query generation requires an accurate inference of query words, as well as a deeper level of understanding on document semantic structures. Toward this end, we propose a novel generative model called the Graph-augmented Sequence to Attention (G-S2A) network. Adopting an Encoder-Decoder architecture, G-S2A incorporates a sentence-level Graph Convolutional Network (GCN), a keyword-level GCN, as well as a hierarchical recurrent neural network (RNN) into the encoder to generate structural document representations. An attentional Transformer decoder is then applied to combine different types of encoded features to generate a target query. On a query-document dataset from a real-world search engine, our model outperforms several neural generative models on a wide range of metrics.

CCS CONCEPTS

• **Information systems → Information retrieval**; **Query suggestion**;

KEYWORDS

natural language processing; information retrieval; search query generation; query suggestion

This paper is published under the Creative Commons Attribution 4.0 International (CC-BY 4.0) license. Authors reserve their rights to disseminate the work on their personal and corporate Web sites with the appropriate attribution.
WWW '19, May 13–17, 2019, San Francisco, CA, USA
© 2019 IW3C2 (International World Wide Web Conference Committee), published under Creative Commons CC-BY 4.0 License.
ACM ISBN 978-1-4503-6674-8/19/05.
https://doi.org/10.1145/3308558.3313746

ACM Reference Format:
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 *Proceedings of the 2019 World Wide Web Conference (WWW '19), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3308558.3313746

1 INTRODUCTION

Search engines handle a massive amount of natural language queries every second. Although a search query is commonly made up of only a few words, it can reflect the user's intent as well as the theme of the retrieved documents on a conceptual level. In this paper, we consider a reverse problem of web search, which is to infer the natural language query that leads to the click-through of a web document. Automatically reverse-engineering search queries from documents is of great values to personalization in search and news feed apps for a number of reasons. First, a search query usually reveals the conceptual theme of the article retrieved by the query and thus can assist information retrieval and personalized recommendation. Second, we can discover trending topics broadly discussed on the Internet by looking at search queries. Natural language search queries, such as "best SUVs under 50k" and "jobs with least competitions", are more fine-grained than keywords and topics. As a result, concepts represented by these queries provide a unique angle to reveal user interests, as compared to pre-defined topics or key phrases. In the meantime, a query is still less specific than a headline or abstract of the document—a query covers a number of matching articles. Thus, generated queries also serve as excellent labels for fine-grained text classification and clustering.

A natural, conventional method to tag documents with search queries is to check query-document pairs in the click-through logs of a search engine. The first document a user clicks and spends some time reading can be viewed as a positive match to the query, while the remaining documents retrieved by the query may or may not be a match, depending on the search engine. As a result, the positive query-document pairs generated this way would only cover a small portion of all available documents. This fact motivates the need to reverse-engineer the best query for a given document via a generative machine learning model, such that documents can be automatically tagged with the matching search queries, which can in turn enhance personalized search experience and recommender performance.

In this work, we first formally define the problem of search query generation from web documents and show that it is unique as compared to other similar Natural Language Processing (NLP) tasks. We propose a deep learning model called **G**raph-augmented **S**equence to **A**ttention network (G-S2A) to solve this problem. To better capture the most important concepts and semantic attentions within a document, in G-S2A, we extend the existing hierarchical Recurrent Neural Network (RNN) based encoder [38] by novelly incorporating two Graph Convolutional Networks (GCN) [16], namely, a sentence-level GCN that produces a condensed graphical representation of a document, and a keyword-level GCN, which emphasizes on the interactions among critical keywords in a document. Subsequently, an attentional Transformer-based [44] decoder, which is more efficient to train, is adopted to attend over different types of the document representations to generate a natural language query.

We evaluate the proposed generative model on a real-world query-document dataset collected from the Tencent news engine in the QQ mobile browser. Extensive evaluation results suggest that by adopting a number of innovations in the encoding and decoding processes, G-S2A outperforms a number of generative baselines, including the widely popular Seq2Seq model with copying and attention mechanisms and a hierarchical RNN-based generative model [38], on all variants of ROUGE [18] and BLEU [33] scores as well as the Exact Match (EM) ratio.

Content: Heihe City, located in the northwestern part of Heilongjiang Province, has the reputation of the doors of Europe and Asia, the window of China and Russia, and is the first batch of open cities along the border. The only border city in the Heihe Sino-Russian # km border that corresponds to the Russian capital. It is also the largest and highest-profile border city. Famous attractions include Wudalianchi, Sino-Russian National Customs Park, Shaoguan East Film and Television Base, Jinhe Grand Canyon, and Ancient City. Wudalianchi Scenic Area, located in Wudalianchi City, Heihe City, Heilongjiang Province, is composed of the Wushan Lake area of Helong Lake, Heilongjiang Lake, Bailong Lake, Heilongjiang Lake, and is a national AAAAA scenic spot global geological park.

Topics: Heihe city, attractions, parks

Keywords: Wudalianchi, city, world, reputation, Jinhe, national, lake, park, attractions

1-Sentence Summary: Famous attractions include Wudalianchi, Sino-Russian National Customs Park, Shaoguan East Film and Television Base, Jinhe Grand Canyon, and Ancient City.

Query: Tourist attractions worth visiting in Heihe

Figure 1: An example illustrating the differences between topics, keywords, summaries and queries. Note that the keyword "attractions" appears in the content and all outputs, whereas the rest of the query words are inferred.

2 PROBLEM DEFINITION

The goal of search query generation is to reverse-engineer *appropriate* queries from a given piece of text. The term *appropriate* means that the generated query should be similar to what a human user would type into a search engine, and then he/she would click on the input text. By doing this, the user implicitly creates an input-query pair. In this paper, the input is a long document. We also limit the scope of this work to generate a single best query only and leave multi-query generation as future work. We model the search query generation task as a sequence-to-sequence learning problem [40], where given an input text sequence T with n tokens $\{t_1, t_2, ..., t_n\}$, the objective is to maximize the conditional probability of generating query sequence Q with m tokens $\{q_1, q_2, ..., q_m\}$.

We argue that the problem of search query generation is different from existing NLP tasks such as summarization, keyword extraction and topic detection. Fig. 1 is an example from our training data that best illustrates the differences. We use TextRank [29] to extract the keywords and the 1-sentence summary. We manually determine the topics and translate the article content from Chinese to English. We discuss the differences from other tasks separately.

Summarization: We believe search query generation and automatic summarization share the same starting point—both tasks attempt to gain a deeper semantic understanding on the document context. The key difference between them lies in the output length. Summarization consolidates a document into a shorter summary of one or several sentences. Each summary is unique to the document itself and captures its most salient information. In contrast, a query crafted in our problem is much more concise, usually consisting of only a few words. A query does not need to summarize a document accurately, but rather, should match the document on a higher conceptual level. Consider the summary and query in Fig. 1. Here the query does not contain any details about tourist attractions in Heihe, but the document is still a good match to this query because they matched on the concept of "Heihe tourist attractions".

Keyword & key-phrase extraction: The key distinguishing factor here is that query generation relies on semantic understanding while keyword extraction does not. If a word in a search query also appears in the matching document, then it is likely to be a keyword. However, the reverse is not true, as shown in Fig. 1. A document could have many keywords or key-phrases. Yet, simply assembling some of them would not produce an appropriate query. The reason is that a generated query needs to properly and semantically "grasp" the central concept discussed in a document, whereas in keyword & key-phrase extraction we only need to determine whether a word or a phrase is important.

Topic detection: The boundary between topic detection and search query generation lies in their granularities. A traditionally perceived topic is usually predefined, such as "signs of pregnancy", "baby foods", "in-door plants", while a good fraction of queries are very specific, such as "early pregnancy signs", "recipes for three-year-old babies", "best in-door plants for lazy people", which automatically discovers finer-grained topics that people pay more attention to.

In essence, the problem of search query generation is different from yet is related to all previously compared tasks. Therefore, a good model for our problem should first develop a strong semantic understanding of the article context. Then, it should be able to accurately extract critical query terms from the content as well as infer the corresponding higher-level conceptual terms to produce a concise natural language query. In Sec. 4, we design our generative query inference model taking these challenges into consideration.

3 RELATED WORK

In this section, we review the literature in closely related tasks including text summarization, keyword extraction, query-document matching, and topic models.

Summarization: Traditional approaches [7, 29] are extractive based, where salient summary sentences are discovered from a document. Deep extractive models like [4, 30, 34] are able to learn more useful latent semantic features and easily outperform traditional methods. Generative summarization models are attracting increased attention due to their potential in generating diverse summaries. [31] is the first to apply an RNN-based Encoder-Decoder setup with an attention mechanism. [37] further refines this framework by adding a pointer network to permit direct copy from the input. More recently, [41] introduces a hierarchical Encoder-Decoder with a graph attention mechanism to better capture multi-granularity features in a document. Other studies [9, 22, 23] improve summary quality from the perspectives of semantic relations, bottoms-up key-phrase selection and multi-document clustering.

Keyword & Key-phrase Extraction: As another classic NLP task, unsupervised keyword & key-phrase extraction methods like [20, 24, 29, 36] rank all the words in a document and select the top candidates as keywords. Supervised approaches such as [8, 42] initially model this task as a binary classification problem, where key-phrases are distinguished from non-key-phrases. Then, statistical machine learning methods like Naive Bayes [43], Conditional Random Fields (CRFs) [46] and Support Vector Machines (SVMs) [47] have been applied to learn from more useful input features like TF-IDF scores, Part-of-Speech (POS) tags and syntactical functions.

We would also like to point out that many extraction techniques [6, 20, 29] propose that a document should be represented as a graph, in order to discover key dependencies among words. In the query generation problem, if a query word appears in the document, then it is likely to be a keyword of the document. This motivates us to also consider graphical representations in our model.

Query-Doc Matching & Query Generation: Another relevant field is query-document matching, where the best document for a query is selected. Trivial approaches compute an aggregated similarity score between query and document word embeddings. Metrics like cosine and TF-IDF similarities, or Okapi BM25 [35] are often considered. Many deep matching models have also been proposed [14, 15, 21, 25, 32, 39], which are either representation-based that focus on semantic feature extraction, or interaction-based that emphasize pair-wise matching. The MGAN [48] is a more recent framework for query to long document matching. It models a long document as a keyword distance graph, where each vertex is a keyword and edge weights are the inverses of distances between keyword pairs. The only related work on search query generation, to the best of our knowledge, is [45]. It proposes a multi-tasking sequence-to-sequence model that simultaneously perform title compression and query generation on E-commerce product titles.

Topic Models: Conventional methods for topic modelling take either probabilistic, Singular Value Decomposition (SVD) based, or matrix factorization based approaches [1, 2, 13]. Deep learning based methods have also been proposed, [27] applies Deep Belief Nets (DBN) to discover topics from digital media content. [17] applies deep neural networks to bag-of-words document representations and achieve state-of-the-art topic modelling performance. [49] further extends the model to multimodal data. A common characteristic of these approaches is that the generated topic is always a single word or phrase.

4 MODEL

In this section, we present a detailed description of the proposed G-S2A network. Fig. 2 is a complete illustration of the model.

4.1 Hierarchical Bi-directional RNN

To reveal the sequential structure of a document, we first utilize a hierarchical RNN-based encoder [38, 41] to model the document, where the lower-level RNN encodes each sentence word-by-word into a single vector, while the higher-level RNN aggregates all sentence vectors to create a paragraph representation. We believe the hierarchical RNN is an effective approach for learning crucial sequential dependencies among the sentences. In our work, we generate a single vector representation for each sentence S_i using a Bi-directional Gated Recurrent Unit (GRU) [5] network. Next, we join sentence vectors on the first axis to form a 2-D document representation, which is then fed into another sentence-level Bi-GRU to yield sequential sentence embeddings.

4.2 Sentence-Level GCN

In order to capture more complex semantic structures beyond sequential dependencies, we further represent a document as a graph $G_s = \{V_s, E_s\}$, where the vertices V_s correspond to sentences and the edges E_s represent some connections between sentences. Our sentence-level Graph Convolutional Network (GCN) [16] is partially established on top of the word-level Bi-GRU from Sec. 4.1. Specifically, the feature of a vertex input to the GCN is the sentence vector produced by the word-level Bi-GRU.

Each edge weight in our sentence-level document graph reflects the percentage of word overlaps between a pair of sentences, which is determined by dividing the number of unique overlapping words against the total number of unique words in both sentences. The intuition here is that similar sentences have more overlapping words and thus a higher edge weight between them. At this point, we have created a new feature map that captures the pair-wise sentence similarities within a document. Next, a multi-layer GCN learns from such sentence interactions by iteratively convolving on the features of a node and its neighbors. This operation is defined by the following update rule:

$$H^{l+1} = \sigma(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^l W^l), \tag{1}$$

where H^l is the output of the previous GCN layer. For the first layer, H^0 is simply the input vertex features, i.e., the sentence vectors produced by the word-level Bi-GRU. $\hat{A}$ is the adjacency matrix constructed based on edge weights, which is symmetrical. $\hat{D}$ is a diagonal degree matrix where $D_{ii} = \sum_j A_{ij}$. σ is the sigmoid activation function. W^l is the set of trainable weights in layer l. We refer interested readers to [16] for more GCN-related details.

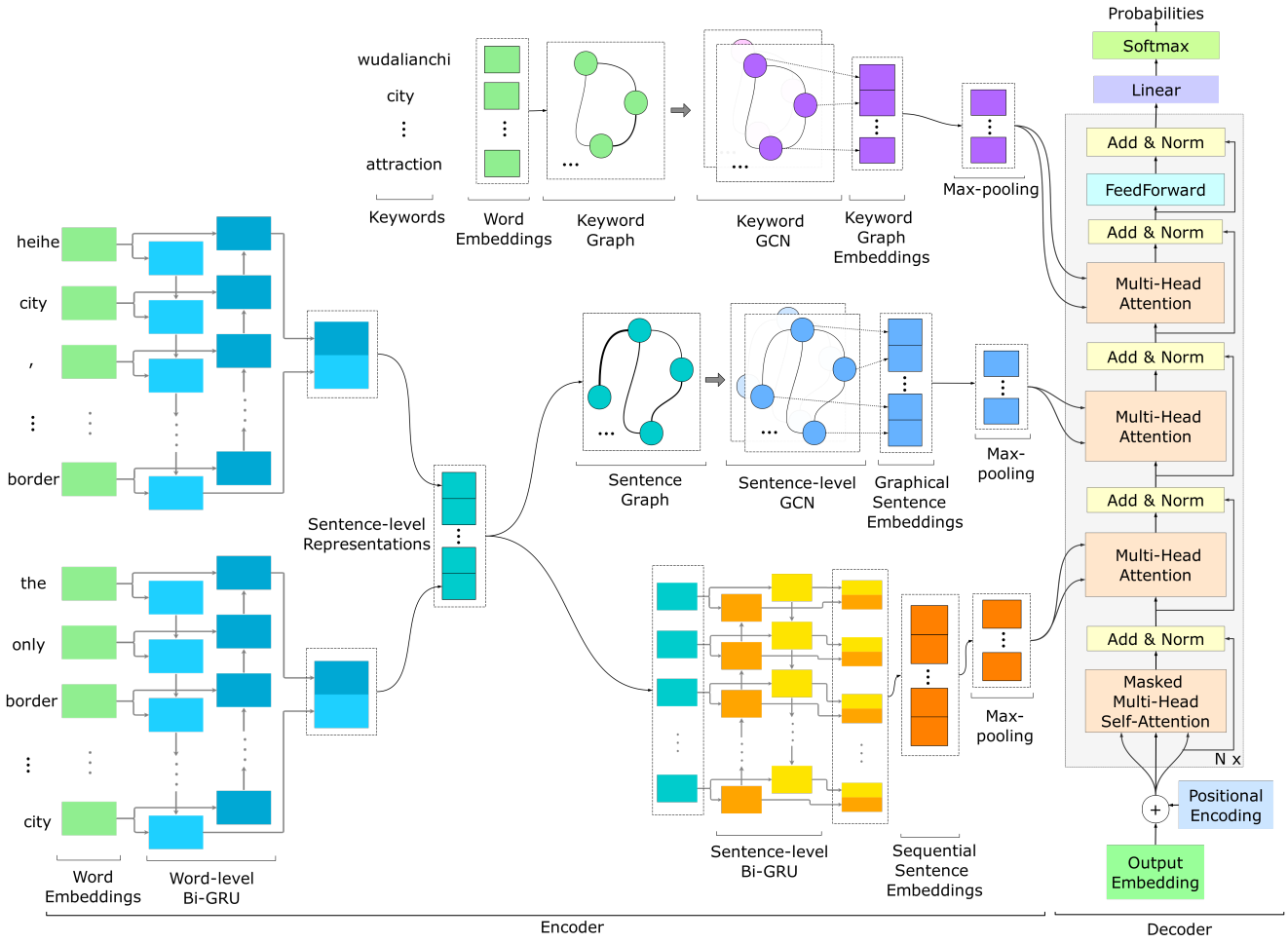

Figure 2: Full view of our proposed model.

4.3 Keyword-Level GCN

As has been mentioned in Sec. 2, if a query word appears in a document, it is usually also a keyword for that document. Therefore, in addition to the sentence-level abstractions, we should permit the decoder to learn from keyword-related information directly. To this end, we further augment our model with a document keyword distance graph similar to [48]. In the keyword distance graph $G_{kw} = \{V_{kw}, E_{kw}\}$, each vertex feature is the embedding vector of a keyword, and the edge weight between two vertices is the inverse average distance among the keywords. Specifically, for unique keywords k_i and k_j, we first locate all of their occurrences in the document. Next, we calculate the pair-wise distance between each pair of occurrences by counting the number of words between them. In other words, if there is no word between two occurrences, the distance is 1; if there is one word in-between, then the distance is 2, and so forth. We take the inverse of the average pair-wise distance. The intuition here is that keywords that are closer to each other are more closely related. Thus, their edge weight is higher in the keyword distance graph. Mathematically, this is expressed as

$$w_{ij} = \frac{1}{d_{ij}^{\text{avg}}} = \frac{m}{\sum_{j=1}^{m} d_{ij}}, \tag{2}$$

where w_{ij} is the edge weight between keywords k_i and k_j, and m is the total number of pair-wise keyword occurrences in the document. After the graph construction, we feed the node and edge features to a keyword-level GCN. The output is an enhanced keyword-level representation, which embodies the graphical keyword interactions within a document.

4.4 Transformer Decoder

Our decoder is a uni-directional Transformer [44] with two additional Multi-Head Attention modules for the GCN outputs. We choose the Transformer decoder over RNN decoders mainly because the Transformer decoder offers a more modularized internal structure, which permits us to readily incorporate more features into the decoding process. Also, the Multi-Head Attention mechanism is more efficient to train due to its easily parallelizable design.

The Multi-Head Attention module is the core component of the Transformer and it is made up of parallel layers of the Scaled Dot-Product Attention. On 2-D feature maps Q, K, and V, the Scaled Dot-Product Attention defines the operation of query Q attending on value V through keys K as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V, \tag{3}$$

where d_k is the feature dimension of K. For the masked self-attention block in Fig. 2, Q, K, V all correspond to the partially generated search query vector. For the remaining attention modules, Q is the partially generated search query vector, while K and V are the output representations from the encoder. The Multi-Head Attention module equally divides the feature dimension of Q, K and V into h heads, producing a set of $\{Q_i, K_i, V_i\}$ with i ranging from 1 to h. Then, a feed-forward layer is added before Q_i, K_i and V_i, which allows the network to learn and adjust these inputs. Finally, the output from each head is concatenated to restore the original shape and fed through another feed-forward network. Interested readers are referred to [44] for more details on the Transformer.

4.5 The Complete Model

We connect the components as depicted by Fig. 2. The word-level Bi-GRU first encodes each sentence in a document. Then, the aggregated sentence vectors propagate to the sentence-level Bi-GRU and sentence-level GCN for further learning. We believe this hierarchical encoding scheme combined with the additional sentence-level graphical embedding allow our model to better understand the semantic interactions within a document. On the other hand, the graphical embedding produced by the keyword-level GCN lets the decoder directly attend on potential query words while maintaining awareness of the document context at the same time, which aids the query term selection and generation process.

In addition to the aforementioned components, we also apply a single *Max-Pooling* layer onto all encoder outputs, since empirically, we find that the Multi-Head Attention mechanism in the decoder performs poorly when attending on long target sequences. Although we have already condensed document representations at the sentence and keyword levels, in reality, a query can be inferred from an even more compact representation—Max-Pooling serves as another stage of significance filtration before decoding.

5 EXPERIMENTATION

In this section, we present the experimental setup and compare results of our model on a real-world Chinese query document dataset against a number of generative baselines.

5.1 Dataset

A suitable dataset for our problem should contain the content of documents and the matching queries. Unfortunately, we cannot locate a publicly available dataset that meets this criteria. Summarization datasets like CNN or DailyMail [11] are not suitable because their summaries are too long to be considered queries. Query-document matching datasets like Ohsumed [12] contains too few queries, whereas datasets with enough unique user queries, such as the NFCorpus [3], have too few unique documents. The lack of a suitable dataset is another evidence that the search query generation problem is new and is less studied.

Therefore, we work with our industry partners at Tencent to create a Chinese query-document dataset, which includes real search queries collected anonymously from the QQ mobile browser. The documents in this dataset mainly consist of news articles or blog posts provided by the Tencent news engine. We construct the query-document pairs using the passive method from Sec. 1, where the first article the user clicks and spends some time reading is considered a matching document. Train, development, and test sets are randomly divided. Table 1 lists the important statistical information about our dataset.

Table 1: Dataset information

Train size	120773
Dev size	10000
Test size	10000
Average document length	286.8
Number of unique queries	10420
Min query length	2
Max query length	11
Average query length	3.2
Document vocabulary	479K
Query vocabulary	9.2K
Average doc-query word overlap	1.85

5.2 Experimental Setup

We adopt the word-overlap based ROUGE [19] and BLEU [33] metrics, which are widely used in summarization and machine translation tasks. In our work, we report the macro-averaged ROUGE-1, 2, L, BLEU-1 to 4 scores. Each metric variant cumulatively considers n-gram (1 to 4) or longest common sub-sequence (L) overlaps. ROUGE-1 is also the metric used to select the best performing model on the development set. ROUGE and BLEU scores reflect how well a generated sequence captures the truth sequence, with higher scores indicating better performance. Additionally, we report the Exact Match (EM) percentages. Because the search queries are a lot more concise, we are interested in how many generated queries match the truth exactly.

We tokenize documents and queries with the Stanford CoreNLP tool [28]. We limit the document word length at 300, All characters are lower-cased. All numbers are replaced with a # symbol. We construct the vocabulary by first combining the document and query vocabularies, then take the top 40000 most frequent words. In every experiment, we train a word embedding layer from scratch, and share it among the encoder and decoder.

For our model, we set the hidden layers of GCNs, the hidden layers of Bi-GRUs, as well as the attention dimension of the decoder to the same value. The position-wise feed-forward dimension is set to be 4 times this value. We also set a global dropout probability for all components. We fix the number of GCN and Transformer layers to 2, and the number of Bi-GRU layers to 1. The word embedding dimension is set to 300. The Max-Pooling window is set to 2. The number of Transformer attention heads is set to 8. We tune the global hidden size and dropout probability on the development set.

Table 2: Performance of various models on the test set

Models	#Weights	ROUGE-1	ROUGE-2	ROUGE-L	BLEU-1	BLEU-2	BLEU-3	BLEU-4	EM
TextRank-kw	-	12.94	0.49	7.97	8.33	2.75	1.92	1.60	0.0
TextRank-kw-cheat	-	15.38	1.11	14.75	15.38	6.94	5.17	4.43	0.0186
TextRank-kp	-	11.85	3.73	11.26	10.73	6.63	4.18	3.32	0.0276
Seq2Seq-Attn	35.7M	72.82	58.15	72.41	71.78	63.17	37.60	25.27	0.6220
Seq2Seq-Attn-Copy	35.7M	72.38	57.24	71.97	71.30	62.46	37.15	24.96	0.6147
HRED	23.7M	73.83	58.45	73.42	72.70	63.69	37.55	25.14	0.6381
Transformer	33.5M	68.87	52.51	68.36	67.51	58.14	34.66	23.31	0.5650
G-S2A	25.8M	**75.18**	**60.21**	**74.73**	**73.98**	**65.32**	**38.67**	**25.94**	**0.6426**

We use the Noam Optimizer [44] with a warm-up steps of 2000 and a label-smoothing constant of 0.1 for training. Our model converges in 100 epochs, where each epoch takes approximately 10 minutes with a mini-batch size of 128 on a GTX1070 GPU. Finally, We compare our model with the following baselines.

TextRank-kw is a simple keyword extractor based on the TextRank [29] method. We include three keyword & key-phrase extraction baselines to prove that search query generation is a more challenging task. We set the number of target keywords to 8.

TextRank-kw-cheat is also a TextRank-based keyword extractor. However, for this baseline we provide the number of words in the truth queries as additional clues.

TextRank-kp is a TextRank-based key-phrase extractor. We extract the most salient key-phrase from the document and use it as the generated query. TextRank baselines do not require training.

Seq2Seq-Attn is the classic encoder-decoder model [40] with the *general* attention mechanism [26]. We feed the entire document as one sequence into the model. We implement this baseline with a Bi-GRU encoder and a Uni-GRU attentional decoder.

Seq2Seq-Attn-Copy further incorporates the copy mechanism [10], which allows the model to directly copy important words from the input. Consider the fact that many query words also appears in the document, as indicated in Table 1. We believe this baseline is a strong competitor.

HRED is a RNN-based hierarchical encoder-decoder model proposed by [38]. Our implementation uses a hierarchical Bi-GRU as the encoder, which is essentially, the same hierarchical sequential encoder in our model. The decoder is a Uni-GRU.

Transformer is the new generative model proposed by [44]. We fix the number of encoder and decoder layers to 6 and the number of attention heads to 8. We tune the attention dimension and dropout probability. The position-wise feed-forward dimension is set to be 4 times of the attention dimension.

5.3 Results & Discussion

In addition to the metrics mentioned in Sec. 5.2, we report the number of trainable weights (#Weights) in every supervised model. We believe the number of trainable weights is critical factor to consider in real-world applications, because a large model often implies a higher cost of training.

The best hidden size for G-S2A is 128. As we can observe in Table 2, G-S2A outperforms all baselines on all metrics, which proves the effectiveness of the proposed additional components. We also note that the performance for keyword extractors are far worse than other models, which indicates that they are not suitable for our problem. Therefore, search query generation and keyword & key-phrase extraction are different problems.

Another interesting conclusion we could draw from Table 2 is that hierarchical models like HRED and G-S2A achieve better performance, but have less trainable weights. We believe this is because the hierarchical encoders break down long documents at an intuitive sentence-level, which effectively exposes its structural information. Other models without the hierarchical design would need to learn such information with more trainable weights.

Table 3: Ablation study results

Model variants	ROUGE-1	ROUGE-L
S2A	73.98	73.55
S2A+sGCN	74.15	73.72
S2A+sGCN+kwGCN	74.30	73.85
G-S2A	75.18	74.73

5.4 Ablation Study

We further evaluate G-S2A through an ablation study and report the changes in ROUGE-1 and ROUGE-L scores.

S2A only contains the hierarchical Bi-GRU encoder and the attentional Transformer decoder. We remove the two GCNs and the Max-Pooling layer from G-S2A to create this model.

S2A+sGCN incorporates the sentence-level GCN.

S2A+sGCN+kwGCN further includes the keyword-level GCN, yet without the Max-Pooling layer.

From Table 3, we observe that S2A initially outperforms the HRED model. We believe this is attributed to the Multi-Head Attention mechanism, which the HRED lacks. With the addition of more components, the performance of the model consistently improves. This proves our initial claim that exploring sentence-level and keyword-level graphical representations for long documents is extremely helpful in the search query generation task.

6 CONCLUSION AND FUTURE WORK

In this paper, we investigate the new problem of search query generation from long documents. We propose that graphical document representations help build a deeper semantic understanding of context and improve query term selection & inference. We introduce the G-S2A model, which incorporates a sentence-level GCN and keyword-level GCN to learn from the corresponding document

graphs. G-S2A outperforms a number of generative baselines, at the same time it is more light-weight. We believe the effective learning of graph features, combined with a modularized decoder, makes G-S2A a superior model for the problem of search query generation. For future work, we plan to extend our model toward multi-query generation and explore other forms of document representations.

REFERENCES

[1] Sanjeev Arora, Rong Ge, and Ankur Moitra. 2012. Learning topic models–going beyond SVD. In *Foundations of Computer Science (FOCS), 2012 IEEE 53rd Annual Symposium on*. IEEE, 1–10.

[2] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research* 3, Jan (2003), 993–1022.

[3] Vera Boteva, Demian Gholipour, Artem Sokolov, and Stefan Riezler. 2016. A Full-Text Learning to Rank Dataset for Medical Information Retrieval. In *European Conference on Information Retrieval*. Springer, 716–722.

[4] Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. *arXiv preprint arXiv:1603.07252* (2016).

[5] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).

[6] Adrien Bougouin Florian Boudin Béatrice Daille. 2013. Graph-Based Topic Ranking for Keyphrase Extraction. (2013).

[7] Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of artificial intelligence research* 22 (2004), 457–479.

[8] Eibe Frank, Gordon W Paynter, Ian H Witten, Carl Gutwin, and Craig G Nevill-Manning. 1999. Domain-specific keyphrase extraction. In *16th International joint conference on artificial intelligence (IJCAI 99)*, Vol. 2. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 668–673.

[9] Sebastian Gehrmann, Yuntian Deng, and Alexander M Rush. 2018. Bottom-up abstractive summarization. *arXiv preprint arXiv:1808.10792* (2018).

[10] Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. *arXiv preprint arXiv:1603.06393* (2016).

[11] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*. 1693–1701.

[12] William Hersh, Chris Buckley, TJ Leone, and David Hickam. 1994. OHSUMED: an interactive retrieval evaluation and new large test collection for research. In *SIGIR-94*. Springer, 192–201.

[13] Thomas Hofmann. 2000. Learning the similarity of documents: An information-geometric approach to document retrieval and categorization. In *Advances in neural information processing systems*. 914–920.

[14] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in neural information processing systems*. 2042–2050.

[15] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*. ACM, 2333–2338.

[16] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[17] Stanislas Lauly, Yin Zheng, Alexandre Allauzen, and Hugo Larochelle. 2017. Document neural autoregressive distribution estimation. *The Journal of Machine Learning Research* 18, 1 (2017), 4046–4069.

[18] Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of summaries. In *Proc. ACL workshop on Text Summarization Branches Out*. http://research.microsoft.com/~cyl/download/papers/WAS2004.pdf

[19] Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out* (2004).

[20] Marina Litvak, Mark Last, Hen Aizenman, Inbal Gobits, and Abraham Kandel. 2011. DegExt-A language-independent graph-based keyphrase extractor. In *Advances in Intelligent Web Mastering–3*. Springer, 121–130.

[21] Bang Liu, Ting Zhang, Di Niu, Jinghong Lin, Kunfeng Lai, and Yu Xu. 2018. Matching Long Text Documents via Graph Convolutional Networks. *arXiv preprint arXiv:1802.07459* (2018).

[22] Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman Sadeh, and Noah A Smith. 2018. Toward abstractive summarization using semantic representations. *arXiv preprint arXiv:1805.10399* (2018).

[23] Peter J Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018. Generating wikipedia by summarizing long sequences. *arXiv preprint arXiv:1801.10198* (2018).

[24] Zhiyuan Liu, Peng Li, Yabin Zheng, and Maosong Sun. 2009. 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*. Association for Computational Linguistics, 257–266.

[25] Zhengdong Lu and Hang Li. 2013. A deep architecture for matching short texts. In *Advances in Neural Information Processing Systems*. 1367–1375.

[26] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025* (2015).

[27] Lars Maaloe, Morten Arngren, and Ole Winther. 2015. Deep belief nets for topic modeling. *arXiv preprint arXiv:1501.04325* (2015).

[28] Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*. 55–60.

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

[30] Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. SummaRuNNer: A Recurrent Neural Network Based Sequence Model for Extractive Summarization of Documents.. In *AAAI*. 3075–3081.

[31] Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023* (2016).

[32] Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2016. Text Matching as Image Recognition.. In *AAAI*. 2793–2799.

[33] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, 311–318.

[34] Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304* (2017).

[35] Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends® in Information Retrieval* 3, 4 (2009), 333–389.

[36] Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. 2010. Automatic keyword extraction from individual documents. *Text Mining: Applications and Theory* (2010), 1–20.

[37] Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368* (2017).

[38] Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C Courville, and Joelle Pineau. 2016. Building End-To-End Dialogue Systems Using Generative Hierarchical Neural Network Models.. In *AAAI*, Vol. 16. 3776–3784.

[39] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. Learning semantic representations using convolutional neural networks for web search. In *Proceedings of the 23rd International Conference on World Wide Web*. ACM, 373–374.

[40] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. 3104–3112.

[41] Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2017. Abstractive document summarization with a graph-based attentional neural model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vol. 1. 1171–1181.

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

[43] Yasin Uzun. 2005. Keyword extraction using naive bayes. In *Bilkent University, Department of Computer Science, Turkey www. cs. bilkent. edu. tr/~guvenir/courses/CS550/Workshop/Yasin_Uzun. pdf*.

[44] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*. 5998–6008.

[45] Jingang Wang, Junfeng Tian, Long Qiu, Sheng Li, Jun Lang, Luo Si, and Man Lan. 2018. A Multi-task Learning Approach for Improving Product Title Compression with User Search Log Data. *arXiv preprint arXiv:1801.01725* (2018).

[46] Chengzhi Zhang. 2008. Automatic keyword extraction from documents using conditional random fields. *Journal of Computational Information Systems* 4, 3 (2008), 1169–1180.

[47] Kuo Zhang, Hui Xu, Jie Tang, and Juanzi Li. 2006. Keyword extraction using support vector machine. In *International Conference on Web-Age Information Management*. Springer, 85–96.

[48] Ting Zhang, Bang Liu, Di Niu, Kunfeng Lai, and Yu Xu. 2018. Multiresolution Graph Attention Networks for Relevance Matching. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM, 933–942.

[49] Yin Zheng, Yu-Jin Zhang, and Hugo Larochelle. 2016. A deep and autoregressive approach for topic modeling of multimodal data. *IEEE transactions on pattern analysis and machine intelligence* 38, 6 (2016), 1056–1069.