A Deep Generative Approach to Search Extrapolation and Recommendation

Fred X. Han[1], Di Niu[1], Haolan Chen[2]
Kunfeng Lai[2], Yancheng He[2], Yu Xu[2]
[1]University of Alberta, Edmonton, AB, Canada
[2]Platform and Content Group, Tencent, Shenzhen, China

ABSTRACT

Related search query recommendation is a standard feature in many modern search engines. Interesting and relevant queries often increase the active time of users and improve the overall search experience. However, conventional approaches based on tag extraction, keywords matching or click graph link analysis suffer from the common problem of limited coverage and generalizability, which means the system could only make suggestions for a small portion of well-formed search queries. In this work, we propose a deep generative approach to construct a related search query for recommendation in a word-by-word fashion, given either an input query or the title of a document. We propose a novel two-stage learning framework that partitions the task into two simpler subproblems, namely, relevant context words discovery and context-dependent query generation. We carefully design a Relevant Words Generator (RWG) based on recurrent neural networks and a Dual-Vocabulary Sequence-to-Sequence (DV-Seq2Seq) model to address these problems. We also propose automated strategies that have retrieved three large datasets with 500K to 1 million instances, from a search click graph constructed based on 8 days of search histories in Tencent QQ Browser, for model training. By leveraging the dynamically discovered context words, our proposed framework outperforms other Seq2Seq generative baselines on a wide range of BLEU, ROUGE and Exact Match (EM) metrics.

CCS CONCEPTS

• **Information systems** → **Query log analysis**; **Query suggestion**; *Query intent*.

KEYWORDS

search recommendation; natural language generation; information retrieval

ACM Reference Format:
Fred X. Han[1], Di Niu[1], Haolan Chen[2] and Kunfeng Lai[2], Yancheng He[2], Yu Xu[2]. 2019. A Deep Generative Approach to Search Extrapolation and Recommendation. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3292500.3330786

KDD '19, August 4–8, 2019, Anchorage, AK, USA

ACM ISBN 978-1-4503-6201-6/19/08…$15.00
https://doi.org/10.1145/3292500.3330786

1 INTRODUCTION

It is an essential ability for a modern search engine to extrapolate beyond the input query and recommend related queries that appeal to the user's interests, therefore improving his/her search experience. Google displays a list of recommended search queries in the "Searches related to" at the bottom of the results page, as illustrated in Figure. 1. Yahoo! offers a similar list of other query recommendations in "Also Try" before all the results. Search recommendation is different from query rewrite [1, 11, 25], the goal of which is to reformulate a search query into a new query that is easier for the search engine to process while still maintaining the overall meaning. In contrast, for example, if we input "what is a diet for weight loss" in Google, we get search recommendations such as "how to lose weight at home", and "7 day diet plan for weight loss." These suggested queries do not necessarily have the same meaning as the original query but are intended to attract the user's attention and boost click-through rates. For the same reason, many news feed apps including Tencent QQ Browser, may also provide several recommended search queries at the end of an article, aiming to prolong a user's activity and increase click-through rates.

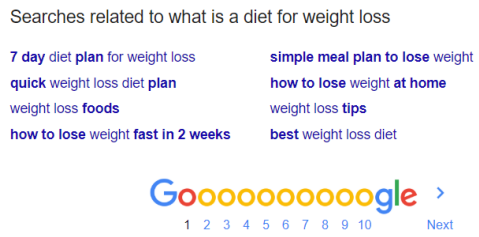

Figure 1: An example of query recommendation in Google.

It is a natural idea to identify related searches by analyzing the search logs, which form the *click graph*, a colossal, bipartite graph that records the documents that have been clicked on in response to past queries. For instance, Tencent QQ Browser[1] typically records approximately 100 million click histories per day, where each log instance consists of a user query and a document title which the user clicked. Although rich information about connections among queries and connections between queries and documents can be dug out from the click graph through extensive link analysis techniques [1, 13], heavily relying on data mining performed on the click graph (possibly with the help of semantic analysis) may yield limited search recommendation performance, mainly due to two reasons:

[1] Tencent QQ Browser has the largest market share in the Chinese mobile browser market with more than 100 million daily active users.

First, the click graph is inherently sparse. Only for very hot topics, e.g., "weight loss", "trade war", etc., a document is connected with multiple queries and a query may lead to the clicks of different documents. However, the vast majority of documents are retrieved by only a couple of queries, while most queries lead to the clicks of a single dominant document. This is also the reason that in most search engines, not all queries or articles would have a related query suggestion. *Second*, such a graph mining approach critically depends on the existence of highly similar queries in the click graph, while that is not always the case due to the flexibility of natural language. Similarly, the click graph cannot encapsulate all possible document titles as new documents are being generated on the web every day.

We argue that a deep generative model can serve as a generalizable alternative that overcomes the limitations of the graph analysis mentioned above. However, text generation based on the widely popular Seq2Seq models [31] suffers from a well-known weakness—the training complexity in the open domain, i.e., the vocabulary that is of interest to a search engine or a content feeds app is too large such that the model must be trained on overwhelmingly large datasets to yield any reasonable performance.

To tackle these challenges, we propose a two-stage generative framework to be used for related search query recommendation in the Tencent QQ Browser. In essence, we break down related query generation into two stages, context discovery and context-aware query generation, which are summarized as follows:

First, given either a user query or a document title, we propose a Relevant Words Generation model (RWG) to extrapolate the query or document into a set of relevant keywords. For instance, relevant words for query *fuel efficient SUVs* include *gas-mileage, cars, money-saving, price*, etc. The proposed RWG discovers additional latent semantical relations among words and learns context-dependent word co-occurrence patterns from similar queries.

Second, to generate the target query for recommendation, we propose the Dual-Vocabulary Sequence-to-Sequence (DV-Seq2Seq) model. It maintains two output vocabularies: a static vocabulary consisting of top X most frequent words, and a dynamic vocabulary composed of the input query words and relevant words discovered by the RWG model. During generation, DV-Seq2Seq selects the next predicted word from one of the two vocabularies with an attention mechanism based on a Multi-Head Attentional Relevant Words Selector (RWS) module, which is inspired by the Transformer [32].

Finally, we propose and describe an automatic procedure to generate the training data required by our two-stage framework, by analyzing word relations and the rich click behavior present in a large click graph constructed from 8 days of click logs. We evaluate our framework with around 1 million records for the RWG model, 1 million records for the query-to-query generation task and 500K records for the title-to-query generation task. We compare our proposed framework against several Seq2Seq generative baselines including the CopyNet. Evaluation results suggest that our approach outperforms all baselines on metrics including BLEU-n [23], ROUGE-n [17] and Exact Match (EM) ratio. We show in Sec. 5 and Sec. 6 that our proposed approach also strikes a good balance between performance, time-complexity, and interpretability.

2 RELATED WORKS

Our work draws inspiration from several recent research achievements in the field of Natural Language Processing (NLP), deep learning and Information Retrieval (IR).

2.1 Generative Models

Generative models construct phrases and sentences from an output vocabulary in a word-by-word fashion. The most popular generative models follow a sequence-to-sequence (Seq2Seq) architecture and composes of an Encoder Recurrent Neural Network (RNN) and a Decoder RNN [31]. Seq2Seq models are proven to be performant in a number of NLP tasks like Automatic Summarization [19], Dialogue Systems [28] and Reading Comprehension [18, 37].

The most influential augmentation to the Seq2Seq model is the attention mechanism [2, 21], where the next decoder output is combined with a weighted sum of encoder hidden representations. The copy mechanism (CopyNet) [9] is another useful augmentation. It permits the decoder to directly copy words from the input, which allows the model to generate words that are not from the static output vocabulary. CopyNet is a major inspiration to our work, in fact, we discuss in Sec. 3 that it is a special case of employing a dynamic output vocabulary, where the contextually relevant words are only taken from the input. More recently, [32] proposes a new generative model called the Transformer, which relies only on a Multi-Head Attention mechanism to directly learn complex semantic relations among words. Even without an RNN, the Transformer achieves state-of-the-art performance on multiple NLP tasks [5].

The idea of incorporating a dynamic output vocabulary into generative models has been touched upon by prior researches, with [34] being the most recent work on this subject. The main difference is between [34] and our model is that [34] constructs an end-to-end trainable Seq2Seq chatbot that jointly learns output generation and dynamic vocabulary selection. Although an end-to-end model may appear simpler, we argue that it is less practical for real-world applications. *First*, an end-to-end model with a dynamic vocabulary is trickier to train, since the loss function needs to be carefully designed. *Second*, it is more difficult to control the quality of the dynamic vocabulary. In [34], the loss on dynamic vocabulary construction is approximated with Monte Carlo sampling, which means the performance of the model is sensitive to the sample size. *Third*, dynamic vocabulary within an end-to-end model is less likely to be transferable to other tasks. Therefore, we explicitly assign the tasks of dynamic vocabulary generation and query generation to two models and train each individually.

Other works on machine translation [12, 16, 22] leverage bilingual word co-occurrence and word alignment features to find the semantically related words. However, such features are often task-specific. To the best of our knowledge, we are the first to apply a generative Seq2Seq model with a dynamic output vocabulary to search query recommendation.

2.2 Query Expansion and Generation

Query Expansion (QE) is classic research topic in IR. The goal of QE is to expand around the context of a search query and discover additional keywords or concepts, which is closely related to the problem of related query recommendation. [35] is an early work

that jointly combines word context features from the query and the retrieved documents. [4] is the first work to propose a probabilistic query expansion approach. [6] relies on association rules to build a query relation graph, then extract relevant concepts to expand on the input query. [14] retrieves the expansion candidates by considering word co-occurrences during a click session. [7, 8, 25, 26] train SMT models to learn word and phrase correspondence features from large amounts of click-through data.

More recently, direct generation of queries using Seq2Seq models is attracting increased attention. [20] incorporates an additional pointer decoder to directly copy words from the input text. [33] proposes a multi-tasking Seq2Seq model for title compression and query generation on E-commerce product titles. [36] and [11] perform direct query-to-query generations, while [10] combines a hierarchical Encoder and a Graph Convolutional Network (GCN) to generate queries from long documents.

3 MODEL

In this section, we review the details of our proposed related search query recommendation framework, as depicted in Fig. 2.

3.1 Relevant Words Generator

As identified by other works [20, 25], a major issue limiting query understanding is the problem of incomplete context, also known as the *Lexical Chase* problem [25]. For search queries, this problem is commonly caused by missing keywords, or unknown Named-Entities. For example, in the query "xs max price", the user is referring to "Apple iPhone". However, when building a generative search recommendation model, if the word "xs" is not in the output vocabulary and we do not explicitly specify that "xs" is related to "Apple iPhone", the model will then generate solely from "<OOV> max price", which often results in poor performance. To provide a more refined context, we design a novel Relevant Words Generator (RWG) to infer additional keywords given a query. It maintains a large output vocabulary to learn a more complete, context-dependent word co-occurrence pattern. This alleviates the prediction difficulty of the second stage Seq2Seq model, allowing it to carry a much smaller output vocabulary, which in turn improves performance and training efficiency.

Formally, we define the problem of relevant words generation as given an input query Q of n words, $Q = \{w_1^Q, w_2^Q, ..., w_n^Q\}$, and a large vocabulary of V_{RWG} words. We learn a model θ_{RWG} to maximize the probability of a relevant words set R_Q of t words, $R_Q = \{w_1^R, w_2^R, ...w_t^R\}$. This objective consolidates into

$$\theta_{RWG} = \text{argmax}_\theta \prod_{i=1}^{t} P(w_i|Q;\ \theta), \tag{1}$$

for every $w_i \in R_Q$. Note that R_Q could contain words from Q.

We employ a Bi-directional Gated Recurrent Unit (GRU) [3] as a context encoder. For a query Q, we first embed each of its words, then feed the embedding vectors into the Bi-GRU one by one in forward and reverse directions. The output hidden states from both directions are concatenated together to form the context vector of Q. Next, we feed this vector through a fully-connected [27] + Softmax layer to project into a V_{RWG}-dimensional space.

For input query Q, we train the model to maximize the probability of in R_Q in a single iteration, by minimizing the Binary Cross-Entropy between the output distribution and the binary target distribution. During inference, we take the top-k predicted words as relevant words for the next stage, where k is a hyper-parameter.

3.2 Dual-Vocab Seq2Seq

Formally, we define the problem of context-aware query generation as given an input sequence $I = \{w_1^I, w_2^I, ...w_n^I\}$, either a search query or a document title, and given a set of relevant words $R_I = \{w_1^R, w_2^R, ...w_t^R\}$, our model $\theta_{Seq2Seq}$ predicts an output search query $O = \{w_1^O, w_2^O, ...w_h^O\}$ in a word-by-word fashion, by maximizing the following conditional probability distribution:

$$\text{argmax}_\theta \prod_{i=1}^{h} P(w_i^O|w_{i-1}^O, w_{i-2}^O, ...w_1^O, Q, R_Q;\ \theta). \tag{2}$$

The only difference between (2) and the traditional Seq2Seq learning objective [31] is the incorporation of R_Q.

Similar to the RWG, our Seq2Seq model also adopts a Bi-GRU encoder for the input sequence I. We denote the output context vector from the encoder as $\boldsymbol{C}_I$, with a dimension of $n \times 2d$, where d is the hidden size of the Bi-GRU. In the decoder, when predicting the ith output word w_i^o, we combine features from three sources: 1) the encoder context vector $\boldsymbol{C}_I$, 2) the embedding vector $\boldsymbol{e}_{w_{i-1}^o}$ of the previously decoded word, and 3) the relevant words set R_Q.

Since R_Q has been generated from a much larger vocabulary in the RWG model, it likely contains words that are out-of-vocabulary (OOV) for the Seq2Seq model. Therefore, we cannot define a fixed-size fully-connected + Softmax projection output layer. Instead, we need an architecture that outputs a probability distribution over vocabularies of varying sizes and contents. Conveniently, the attention mechanism [2, 21, 32] achieves this. A *generalized* version of the attention mechanism can be written as

$$c_i = \sigma(\ f(\ \boldsymbol{h}_i^{Dec} \cdot \boldsymbol{C}_I^T\ )\ ), \tag{3}$$

$$\boldsymbol{h}_i^C = \sum_{j=1}^{n} s_i^j \boldsymbol{C}_I^j, \tag{4}$$

$$\boldsymbol{h}_i^{Dec} = g(\ \boldsymbol{h}_i^C,\ \boldsymbol{h}_i^{Dec}\ ), \tag{5}$$

where $\boldsymbol{h}_i^{Dec}$ denotes the current hidden state of the decoder. σ denotes a Softmax layer. "$\cdot$" represents matrix multiplication. c_i is a 1 x n dimensional vector of attention scores which captures the importance of each word in the input sequence for the ith output word. Furthermore, j in (4) indexes the jth element in c_i and the jth row of $\boldsymbol{C}_I$. And f and g are customized operations. In general, the attention mechanism first computes attention scores over $\boldsymbol{C}_I$ and use them to calculate a weighted sum of $\boldsymbol{C}_I$, i.e. a weighted context $\boldsymbol{h}_i^C$ which is then combined with the previous hidden state.

c_i already resembles a probability distribution over the words in I, because its values are between 0 and 1. The CopyNet [9] takes advantage of this by treating each attention score as a probability of copying over the corresponding word. We propose that the CopyNet already provides all the necessary construct for handling a dynamic vocabulary. In fact, it is a special case where all the relevant words inside the dynamic vocabulary are from the input sequence. After experimenting with several network architectures,

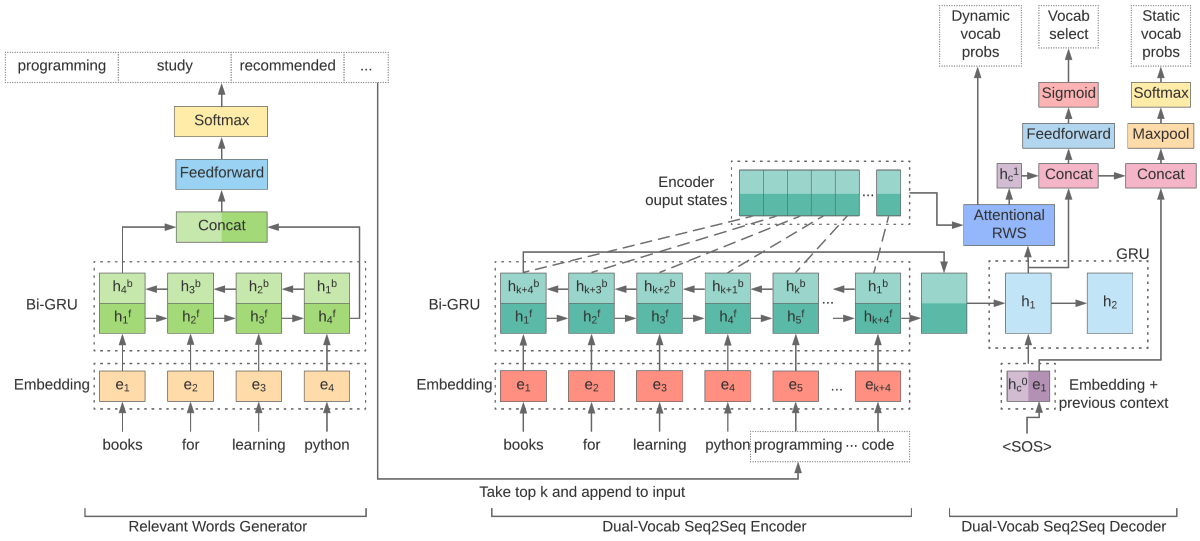

Figure 2: Our proposed two-stage generative framework. For simplicity, only one decoding step is shown.

we find that simply concatenating the relevant words after the input sequence and then utilizing the copy mechanism on this new sequence achieves the best performance. Therefore, this concatenated new sequence constitutes the dynamic vocabulary. For the ease of reference, we re-name our copy mechanism the *relevant words selector* (RWS). Different from the original CopyNet [9], we adopt a variant of the Multi-Head Attention proposed by [32] to formulate a novel Multi-Head attentional RWS module. We discuss its details in the following subsection.

The backbone of our decoder is a uni-directional GRU, which is initialized with the last hidden state of the encoder. A single decoding step involves the following operations,

$$\boldsymbol{h}_i^{GRU} = GRU(\,[\boldsymbol{e}_{w_{i-1}^o}; \boldsymbol{h}_{i-1}^C]\,), \tag{6}$$

$$\boldsymbol{h}_i^C, \boldsymbol{P}_i^{DV} = RWS(\,\boldsymbol{h}_i^{GRU}\,), \tag{7}$$

$$\boldsymbol{P}_i^{SV} = \sigma f(\,Max_m(\,[\boldsymbol{e}_{w_{i-1}^o}; \boldsymbol{h}_i^C; \boldsymbol{h}_i^{GRU}]\,)\,), \tag{8}$$

where [] represents the concatenation of vectors. $\boldsymbol{P}_i^{DV}$ denotes the probability distribution over the dynamic vocabulary and $\boldsymbol{P}_i^{SV}$ is the probability distribution over the static vocabulary. $\boldsymbol{e}_{w_{i-1}^o}$ and $\boldsymbol{h}_{i-1}^C$ are the previous word embedding and weighted context vectors. Max_m stands for a Maxpooling layer with a window of m. σf is the standard fully-connected + Softmax projection setup. We use a special Start-of-Sequence token as the first input word to the decoder and initialize $\boldsymbol{h}_0^C$ to all zeros.

Another important decision our model must make is which vocabulary to select for output. After the execution of (6) and (7), we perform the following operation alongside (8),

$$p_i^{cDV} = \text{sig}\, f(\,[\boldsymbol{h}_i^C; \boldsymbol{h}_i^{GRU}]\,), \tag{9}$$

where sig f is a fully-connected layer followed by a Sigmoid activation. The output is a scalar probability value for choosing the next output word from the dynamic vocabulary. We train on the CopyNet objective functions, where the loss from RWS corresponds to the copy loss, while the loss from word prediction on the static vocabulary corresponds to the generative loss. We omit details on these objectives and refer interested readers to [9].

3.2.1 Multi-Head Attentional RWS. The computational workload of the attention mechanism increases as more words are added to the input. Specifically, the matrix multiplications in (3) and (4) become slow and memory-hungry. Therefore, we adopt the Multi-Head Attention from [32]. It addresses this issue by first dividing each input vector into h heads. Then, it executes h head-to-head attentions in parallel, where each matrix multiplication is performed on a dimension that is h times smaller.

Since our decoder already includes a strong GRU learner, we further simplify the Multi-Head Attention to construct a Multi-Head Attentional RWS module, as illustrated by Fig. 3. Instead of three inputs [32], our module takes in $\boldsymbol{h}_i^{Dec}$ and $\boldsymbol{C}_I$. We also remove the last Feedforward layer after the concatenation of heads. From repeated experiments, we found that our module achieves similar performance compared to the original Multi-Head Attention. To generate a probability distribution over the dynamic vocabulary, we compute a weighted sum of the pre-Softmax attention heads through a fully-connected layer, followed by a Softmax projection.

4 DERIVING DATA FROM CLICK GRAPHS

We now introduce how to automatically retrieve training data from a click graph for both stages of our framework. Given an undirected, bipartite click graph G consisting of query vertices V_Q, document title vertices V_D, and weighted edges E, where the weight of an edge $e(q, d)$ represents how many clicks of document d are attributed to query q, we define the **K-hop sibling queries set** $S^K \subset V_Q$ as a set of query vertices such that

(1) S^K contains at least two unique queries;
(2) There exists a shortest path on G between any two queries in S^K.

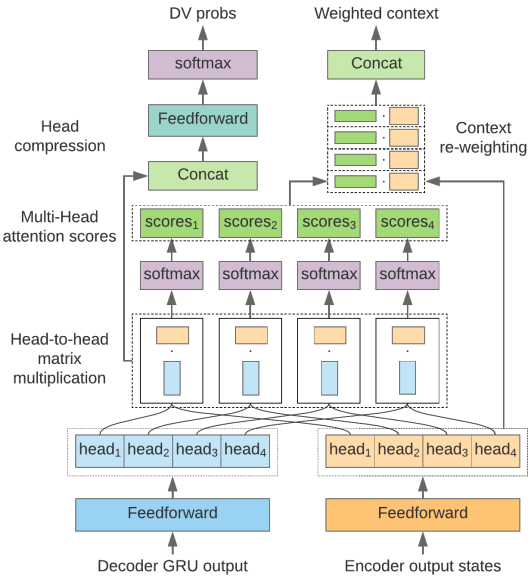

Figure 3: An illustration of the Multi-Head Attentional Relevant Words Selector (RWS) module with 4 heads.

(3) The maximum number of title vertices in V_D passed through by any shortest path in S^K is K.

Additionally, we define the set of document titles passed through by queries in S^K as $D(S^K)$. With this setup, two queries in S^K are likely to be semantically related, where the value of K determines the degree of relatedness. Fig . 4 is an example of two semantically related queries. As we can observe, with $K = 1$, the sibling queries are more likely to be semantically identical. When K increases, we discover additional related queries. After determining an appropriate value for K through statistical analysis, we discover all the S^K clusters within G. To reduce noise and computational cost, we limit the number of out-going edges by keeping only the top-p weighted edges in G for each query vertex, where a weight is the number of times that click occurs. We also constrain that each query can only appear in one cluster to avoid conflicts during data generation. If a query appears in more than one sibling set, we re-assign it to the set where the weight of the connecting edge is the highest and prune its other out-going edges.

Next, we separately retrieve training data for both stages of our framework:

Relevant Words Discovery. For every query Q in a sibling set S^K, we define its corresponding relevant words set R_Q as the *keywords* from all queries in S^K. Therefore, we have a single target R_Q for all queries in S_K.

Target Query Selection. To constrain the training data size, for every sibling set S^K, we select a query from it as the *representative* of the set, which will be the target query to generate for all other queries in S^K, and similarly, for all document titles in $D(S^K)$ as well. We also constrain that each unique document title can have only one corresponding target query by randomly pruning repeated entries.

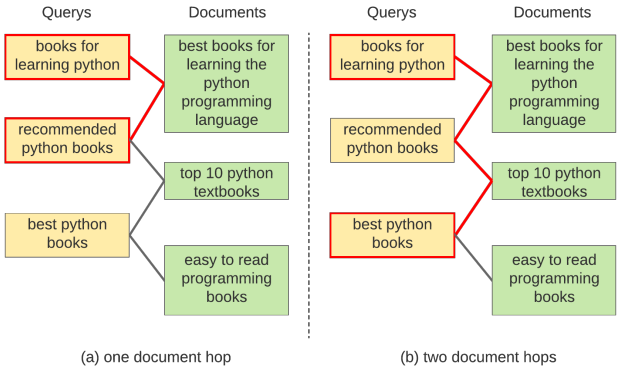

Figure 4: An example showing two sibling queries discovered with (a) one document hop and (b) two document hops, as highlighted in red.

The representative query for a sibling set S^K is select based on the following criteria:

- The *cumulative weight of outgoing edges* of a query is a strong indicator for its popularity, generalizability and correctness. We compute the sum of weights of outgoing edges for every query in S^K and normalize the results between 0 and 1. We denote this score c_{click}.
- The *number of words* in a query usually reflects its specificity. We want to constrain the complexity of target queries. Therefore, we normalize the number of words for every query in S^K between 0 and 1, and record a score c_{len} as 1 minus its normalized length.
- The *percentage of overlapping keywords* between keywords in a query in S^K and keywords in document titles from $D(S^K)$ is a strong indicator of relatedness. Similarly, we normalize this measure for every query among S^K between 0 and 1, denoted by $c_{overlap}$.

We compute a final score for each query $q \in S^K$ by taking a weighted sum of all features scores, namely,

$$c_{final}(q) = \alpha\, c_{click}(q) + \beta\, c_{len}(q) + \gamma\, c_{overlap}(q), \qquad (10)$$

where the weights are hyper-parameters and the query with the highest score is selected as the representative.

5 EXPERIMENTATION

We present the detailed experimental procedures [2] and results here. Table. 4 records the results for query-to-query generation while Table. 5 showcases the results for document title-to-query generation.

5.1 Dataset & Pre-processing

We collect 8 days of anonymous click logs recorded in the Tencent QQ mobile browser, which spans November and December, 2018. It contains over 800 million query to document-title entries in Chinese. We first execute the following sequential pre-processing steps:

(1) We remove vulgar entries using a tool developed by Tencent.

[2] Code is available at: https://github.com/xuefei1/RWG_DV-Seq2Seq

Table 1: Statistical information on datasets generated from our click graph.

	RWG			Query-to-Query			Title-to-Query		
	Train	Dev	Test	Train	Dev	Test	Train	Dev	Test
Size	894K	99K	99K	894K	99K	99K	530K	88K	88K
Avg # of words in inputs	5.08	5.04	5.05	5.08	5.04	5.05	11.40	11.37	11.34
Avg # of words in outputs	9.29	10.03	10.04	4.21	4.20	4.19	4.27	4.21	4.20
Avg # of overlapping words	2.19	2.17	2.18	1.98	1.99	1.98	2.42	2.46	2.45
Input vocabulary	209K	62K	62K	209K	62K	62K	210K	74K	74K
Output vocabulary	177K	84K	83K	143K	51K	51K	143K	51K	51K

Table 2: Examples of RWG and query-to-query generation training data.

Input query	Truth relevant words for training RWG	Truth query for training DV-Seq2Seq
箱货汽车大全 cargo vehicles catalog	箱货, 汽车, 大全, 微卡, 商务车, 轻卡, 重卡, 货车, 报价, 价格, 系列, 车型, 牵引车 cargo, vehicle, catalog, micro-truck, van, light-truck, semi-truck, boxer-truck, quote,price, series, model, tow-truck	轻卡之家 home of light-trucks
上海医院招聘 Shanghai hospital hiring	上海, 医院, 招聘, 官网, 医学, 卫生, 信息, 护士, 招聘会, 报名, 工作, 面试, 中心, 单位, 医生, 平台, 公立医院, 卫生局 Shanghai, hospital, hiring, official-site, medical, hygiene, information, nurse, career fair, enroll, job, interview, center, employer, doctor, platform, public hospital, bureau of health	上海卫生人才网 Shanghai medical human resource web
双硬盘怎么装 How to install two hard-drives	双硬盘, 怎么, 装, 电脑, 教程, 安装, 主板, 接线, 固态硬盘, 华硕, 方法, 硬盘, 金士顿, 固态, 台式电脑, 台式机 dual hard-drive, how to, install, compute, tutorial, installation, motherboard, wiring, SSD, Asus, method, hard-drive, Kingston, solid-state, desktop computer, desktop	固态和机械硬盘一起用 using solid-state and mechanical hard-drives together

(2) We remove entries that do not contain any Chinese words in either the query or the title.
(3) We remove entries that contain more than 25 words in either the query or the document title.
(4) To reduce noises generated by misclicks, we remove entries that appears less than 2 times.

About 6.5 million entries remain after the above steps. Next, we build a click graph and apply the data retrieval steps in Sec. 4.

To select the most appropriate K and p values, i.e., the number of document hops and top-p query out-going edges, when building the sibling sets S^K, we first analyze the edge properties of the click graph. Fig. 5 showcase two insightful distributions, (a) reports the percentage of queries with the corresponding number of unique out-going edges. We conclude that most queries (82.84%) only has a single corresponding document. However, a sizable portion (11.71%) of queries click two distinct documents. Second, in the case of more than one unique out-going edges, we investigate the differences among their weights, i.e. the number of times that click occurred. (b) suggests that on average, the second-largest weights (rank 2) are 0.76 times the largest weights (rank 1), which means that these edges are likely leading to another highly relevant document. Combining these statistics, we select $K = 2$ and $p = 2$. For simplicity, we define *keywords* as any verb or noun that are not stop-words.

For weights α, β, γ in (10), their purpose is to balance the three scores and prevent any one of them from dominating the results. Therefore, we manually sampled 1000 sibling sets and examined the representatives. After trying several combinations, we found that a setup of $\alpha = 0.4$, $\beta = 0.3$, $\gamma = 0.3$ well balances all three scores. We then generate data for relevant words discovery, query-to-query generation and title-to-query generation. Table. 1 reports statistical

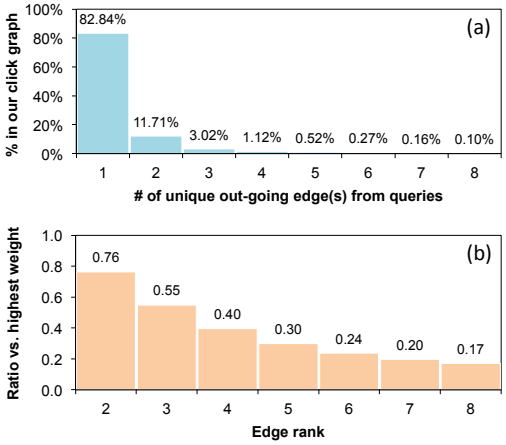

Figure 5: Percentage distribution of unique out-going edges from queries (a) and average ratio of secondary edge weights vs. the highest weights (b) in the click graph.

information on the three generated datasets. Table. 2 showcases examples of training data. Observe that all the retrieved words and queries are closely related to the input.

5.2 General Experimental Setup

We implement our two-stage framework using PyTorch 0.4[24]. We initialize all word embedding layers with the pre-trained, 200d Tencent AILab Chinese embedding [29] and allow each layer to be further fine-tuned. We select top-X most frequent words in the

training output when building a limited static vocabulary of size X and replace all out-of-vocabulary words with <*OOV*> tokens.

We train all models using the Adam optimizer [15] with an initial learn-rate of 0.01 and apply a simple learn-rate decay strategy: If the train/dev loss of one epoch is higher than the previous epoch, decay the learn rate by 0.9, lower-bounded by a minimum learn-rate of $1e-4$. For generative models, we choose the BLEU-4 score as the metric for selecting the best hyper-parameters and terminate training if the dev set performance does not improve for 5 consecutive epochs. For all models, we decode using beam-search with a beam-width of 2 during parameter-tuning and a beam-width of 4 during testing.

5.3 Training and Evaluating the RWG Model

We train the RWG model first by minimizing the Binary Cross-Entropy loss. The input and output word embedding dimensions are set to 209K and 177K, as indicated in Table. 1. We employ top-100 recall rate, i.e. the percentage of truth relevant words that appear in the top-100 predictions, as the metric for selecting the best hyper-parameters. We found that a hidden size of 512 for the Bi-GRU works best on the dev set. The RWG model converges in 35 epochs, with a batch size of 256 on an Nvidia GTX-1070 GPU. Each epoch takes about 30 minutes. We report the average top-$|T|$, top-$2|T|$, top-50, top-100 and top-500 recall rates on the test set in Table. 3, where $|T|$ is the number of truth relevant words for an input query.

Table 3: Top recall rates of the RWG model on the test set.

Top-$\|T\|$	Top-$2\|T\|$	Top-50	Top-100	Top-500
0.6522	0.7704	0.7967	0.8555	0.9207

Consider the fact from Table. 1 that the average number of relevant words per input query is around 10, we believe 20 is an appropriate choice for the top-k words to append after the input.

5.4 Training the DV-Seq2Seq Model

We train two DV-Seq2Seq models for query-to-query and title-to-query generation tasks. We use the same RWG model to generate the top-20 relevant words. We found that an encoder GRU hidden size of 256 works well on both tasks. For regularization, we utilize a Dropout [30] probability of 0.1 in the decoder.

We test two model variants, one with a static vocabulary size of 20K and another with a static vocabulary size of 40K. On the query-to-query generation task, our models converge in about 60 epochs, and 10 more epochs are needed to on the document-title-to-query task. RWG+DV-Seq2Seq-20K model variants are trained with a batch size of 64 on an Nvidia GTX-1070 GPU, where each epoch takes about 90 minutes, while the 40K variants are trained with a batch size of 32 and each epoch costs 120 minutes.

5.5 Baseline Models

We compare our generative framework against the following baseline models. We found an encoder hidden size of 256 for all baseline models also results in the best performance on the dev set, without running into memory problems. We also utilize a Dropout probability of 0.1 in the decoder of all baseline models.

Seq2Seq-*X*: Seq2Seq is the original sequence-to-sequence generative model [31]. We implement this baseline with a Bi-GRU encoder and a Uni-GRU decoder. X denotes the output vocabulary size. We experiment with several sizes to test its effect on the performance. When X = full, we use the complete output vocabulary.

Seq2Seq-Attn-*full*: We augment the Seq2Seq model with the *general* attention mechanism from [21].

CopyNet-*X*: We adopt the CopyNet implementation from [37]. We cannot include a $full$ variant here because the copy mechanism is only useful where there are OOV words in the target query.

5.6 Evaluation Metrics

We report and compare performance on the following metrics:

BLEU-1, 2, 3, 4: BLEU-n [23] is a widely adopted word-overlap metric for evaluating generative models. n means that variant considers at most n-gram overlaps. We reported the macro-averaged BLEU-1, 2, 3, 4 scores on the test set. Higher BLEU-n scores indicate more overlapping words between the generated output and truth.

ROUGE-1, 2, *L*: ROUGE-n[17] is another popular word-overlap metric. We report the macro-averaged uni-gram, bi-gram, and the longest common sub-sequence (L) variants of ROUGE.

Exact Match (EM): We are also interested in the average ratio of generated queries that exactly match the truth queries.

% OOV: The generation of <*OOV*> tokens significantly limits the real-world applicability of a generative model, since it usually renders the entire output useless. We examine the percentage of <*OOV*> among all the generated words. Smaller % OOV indicate less <*OOV*> are generated, but not necessarily, better performance.

6 EVALUATION

We begin by analyzing the effect of output vocabulary sizes. From Tables. 4 and 5, we notice that limiting the output vocabulary size often improves the performance. This makes sense because it alleviates the prediction difficulty of the projection layer. However, when the vocabulary size is too small, i.e. 20K, the BLEU, ROUGE, and EM scores decrease. This is likely caused by a large number of generated <*OOV*> tokens, as indicated by the increase in % OOV.

On both tasks, our best model variant with a 40K static vocabulary outperforms all baseline models on all metrics excluding % OOV. This proves the effectiveness of employing two vocabularies, i.e. a static output vocabulary with a fully-connected + Softmax projection layer and a context-aware dynamic vocabulary with an attention + copy mechanism, in a generative Seq2Seq model. A dual-vocabulary setup also results in considerably less <*OOV*> tokens in the output, compared to a standard Seq2Seq model or a CopyNet with the same sized static vocabulary.

Additionally, our approach achieves better performance on the title-to-query generation task, even if the first stage RWG model is trained on query/relevant words data. This suggests that the RWG model is able to learn useful, generalizable, context word co-occurrence patterns, and not just overfitting to the input.

As for the time-complexity, our two-stage framework is more efficient to train and use, because the RWG does not have sequential decoding steps and the DV-Seq2Seq has a smaller output vocabulary. Each stage can be trained on a consumer-grade GPU like

Table 4: Performance of query-to-query generation on the test set.

Models	ROUGE-1	ROUGE-2	ROUGE-L	BLEU-1	BLEU-2	BLEU-3	BLEU-4	EM	% OOV
Seq2Seq-$full$	43.54	28.46	40.68	35.67	29.62	25.92	16.52	0.1237	0.0
Seq2Seq-120K	45.13	30.35	42.47	38.09	31.81	28.19	18.23	0.1498	0.918
Seq2Seq-80K	45.70	31.71	43.17	38.81	32.87	29.42	18.73	0.1552	4.36
Seq2Seq-40K	45.37	31.29	43.09	39.28	32.89	29.55	19.03	0.1609	9.14
Seq2Seq-20K	42.70	27.51	40.35	36.54	29.58	25.87	16.72	0.1273	17.21
Seq2Seq-Attn-$full$	52.18	41.49	51.05	49.55	43.89	41.25	32.68	0.3406	0.0
CopyNet-120K	48.87	37.88	47.67	46.00	40.27	37.43	29.24	0.2932	2.95
CopyNet-80K	55.12	45.03	54.04	52.61	42.27	44.60	35.19	0.3712	1.55
CopyNet-40K	56.72	46.66	55.66	54.37	48.95	46.20	36.74	0.3871	4.75
CopyNet-20K	55.14	44.19	54.02	52.74	46.77	43.60	34.69	0.3446	7.59
RWG+DV-Seq2Seq-20K	58.10	47.05	56.99	55.84	49.74	46.72	37.46	0.3864	2.74
RWG+DV-Seq2Seq-40K	**58.25**	**47.93**	**57.16**	**55.99**	**50.37**	**47.63**	**38.22**	**0.4047**	1.88

Table 5: Performance of document-title-to-query generation on the test set.

Models	ROUGE-1	ROUGE-2	ROUGE-L	BLEU-1	BLEU-2	BLEU-3	BLEU-4	EM	% OOV
Seq2Seq-$full$	52.14	34.52	47.92	41.02	34.65	29.76	18.69	0.1248	0.0
Seq2Seq-120K	52.34	35.25	48.77	43.13	36.24	31.67	20.10	0.1512	1.82
Seq2Seq-80K	56.60	41.22	53.30	47.64	41.47	37.19	23.18	0.1927	4.34
Seq2Seq-40K	56.08	40.66	53.06	47.94	41.40	37.21	23.43	0.1998	8.68
Seq2Seq-20K	53.50	37.27	50.71	46.15	38.84	34.33	22.06	0.1800	14.77
Seq2Seq-Attn-$full$	59.29	47.67	57.78	55.63	49.82	46.94	35.95	0.3845	0.0
CopyNet-120K	58.10	47.05	56.99	55.84	49.74	46.72	37.46	0.3864	2.74
CopyNet-80K	68.97	60.50	67.91	66.54	62.12	59.64	47.51	0.5244	1.90
CopyNet-40K	72.11	63.82	71.16	69.97	65.54	62.98	50.52	0.5577	3.50
CopyNet-20K	71.65	62.64	70.73	69.71	64.79	61.76	49.66	0.5350	6.21
RWG+DV-Seq2Seq-20K	72.0	63.34	71.17	70.25	65.47	62.63	50.62	0.5502	4.85
RWG+DV-Seq2Seq-40K	**72.75**	**65.18**	**71.93**	**71.00**	**66.90**	**64.61**	**52.32**	**0.5840**	2.89

the GTX-1070 with 8GB of VRAM. During decoding, our framework only needs to project to a large vocabulary *once* in the RWG, whereas end-to-end baselines with larger output vocabularies, such as the Seq2Seq-$full$, Seq2Seq-Attn-$full$ need to perform this time-consuming operation in every step. Even on a GPU with twice the VRAM, like the Nvidia Tesla-P100, these baselines can only train with a maximum batch size of 16.

Furthermore, our proposed framework offers better interpretability, because relevant words generated by the RWG are directly appended to the inputs for the next stage, hence, they are fully visible to the end-users and be customized to suit a variety of applications.

6.1 Case Study

We conduct a simple case study on the query-to-query generation task. In Table. 6, we compare the outputs from two strong competitors, i.e. CopyNet-40K and RWG+DV-Seq2Seq-40K.

Consider the first two cases in Table. 6. Our model generated higher quality queries compared with CopyNet-40K. We believe this is attributed to the relevant words provided by the RWG model, because output words such as *Apple*, or *Kirin* 970 *(another CPU model)* are not from the original input query. Even if the target query does not include any additional words, such as case 3, our model still outperforms the competitor. We speculate that the additional relevant words also helped to better define the overall context. In other words, the decoder in our model has access to more conceptually-related *clues* through the attention mechanism, therefore, its outputs are much more predictable.

7 CONCLUSION AND FUTURE WORK

We introduce a two-stage learning framework for related queries generation. We first retrieve massive amounts of training data from a click graph. Next, our framework breaks down related query generation from input queries as well as document titles into two steps, namely, relevant words discovery and context-aware query generation. We carefully design a Relevant Words Generator (RWG) model and a Dual-Vocabulary sequence-to-sequence (DV-Seq2Seq) model for each sub-problem. A RWG+DV-Seq2Seq setup with a 40K static output vocabulary surpasses all baseline models on both query-to-query and title-to-query generation, in terms of BLEU-1, 2, 3, 4, ROUGE-1, 2, L and Exact Match (EM) metrics. Additionally, these results verify the feasibility and practicality of a deep generative model in tackling the query recommendation task. For future work, we plan to deploy our model in a production environment and further improve its decoding efficiency to meet the strict time-complexity requirements of real-world applications.

REFERENCES

[1] Ioannis Antonellis, Hector Garcia Molina, and Chi Chao Chang. 2008. Simrank++: query rewriting through link analysis of the click graph. *Proceedings of the VLDB Endowment* 1, 1 (2008), 408–421.
[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).
[3] 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).
[4] Hang Cui, Ji-Rong Wen, Jian-Yun Nie, and Wei-Ying Ma. 2002. Probabilistic query expansion using query logs. In *Proceedings of the 11th international conference on*

Table 6: Queries generated by CopyNet-40K and RWG+DV-Seq2Seq-40K, in the query-to-query generation task.

Models	Input query	Target query	Generated query
CopyNet-40K	5s买什么电池	5s 电池推荐	5s 电池<OOV>歌词 5s battery <OOV> lyrics
RWG+DV-Seq2Seq-40K	what battery to buy for 5s	5s battery recommendation	苹果5s换电池多少钱 Apple 5s battery replacement cost
CopyNet-40K	高通670跟与945处理器	骁龙670和845跑分	骁龙670和玩游戏 SnapDragon 670 and playing games
RWG+DV-Seq2Seq-40K	Qualcomm 670 and 945 processor	SnapDragon 670 and 845 benchmark results	骁龙670和麒麟970 SnapDragon 670 and Kirin 970
CopyNet-40K	死侍2斯坦李在哪	死侍斯坦李	<OOV>被<OOV>了 <OOV> got <OOV>
RWG+DV-Seq2Seq-40K	where is Stan Lee in Deadpool 2	Deadpool Stan Lee	斯坦李死侍 Stan Lee Deadpool

World Wide Web. ACM, 325–332.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[6] Bruno M Fonseca, Paulo Golgher, Bruno Pôssas, Berthier Ribeiro-Neto, and Nivio Ziviani. 2005. Concept-based interactive query expansion. In *Proceedings of the 14th ACM international conference on Information and knowledge management*. ACM, 696–703.

[7] Jianfeng Gao, Xiaodong He, Shasha Xie, and Alnur Ali. 2012. Learning lexicon models from search logs for query expansion. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. 666–676.

[8] Jianfeng Gao and Jian-Yun Nie. 2012. Towards concept-based translation models using search logs for query expansion. In *Proceedings of the 21st ACM international conference on Information and knowledge management*. ACM, 1.

[9] 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).

[10] Fred.X Han, Di Niu, Weidong Guo, Kunfeng Lai, Yancheng He, and Yu Xu. 2019. Inferring Search Queries from Web Documents via a Graph-Augmented Sequence to Attention Network. In *Proceedings of The Web Conference 2019*.

[11] Yunlong He, Jiliang Tang, Hua Ouyang, Changsung Kang, Dawei Yin, and Yi Chang. 2016. Learning to rewrite queries. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM, 1443–1452.

[12] Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2014. On using very large target vocabulary for neural machine translation. *arXiv preprint arXiv:1412.2007* (2014).

[13] Glen Jeh and Jennifer Widom. 2002. SimRank: a measure of structural-context similarity. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 538–543.

[14] Rosie Jones, Benjamin Rey, Omid Madani, and Wiley Greiner. 2006. Generating query substitutions. In *Proceedings of the 15th international conference on World Wide Web*. ACM, 387–396.

[15] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[16] Gurvan L'Hostis, David Grangier, and Michael Auli. 2016. Vocabulary selection strategies for neural machine translation. *arXiv preprint arXiv:1610.00072* (2016).

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

[18] Bang Liu, Mingjun Zhao, Di Niu, Kunfeng Lai, Yancheng He, Haojie Wei, and Yu Xu. 2019. Learning to Generate Questions by Learning What not to Generate. In *Proceedings of The Web Conference 2019*.

[19] 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).

[20] Xiaoyu Liu, Shunda Pan, Qi Zhang, Yu-Gang Jiang, and Xuanjing Huang. 2018. Generating Keyword Queries for Natural Language Queries to Alleviate Lexical Chasm Problem. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM, 1163–1172.

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

[22] Haitao Mi, Zhiguo Wang, and Abe Ittycheriah. 2016. Vocabulary manipulation for neural machine translation. *arXiv preprint arXiv:1605.03209* (2016).

[23] 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.

[24] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *NIPS-W*.

[25] Stefan Riezler and Yi Liu. 2010. Query rewriting using monolingual statistical machine translation. *Computational Linguistics* 36, 3 (2010), 569–582.

[26] Stefan Riezler, Yi Liu, and Alexander Vasserman. 2008. Translating queries into snippets for improved query expansion. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*. Association for Computational Linguistics, 737–744.

[27] Frank Rosenblatt. 1961. *Principles of neurodynamics. perceptrons and the theory of brain mechanisms*. Technical Report. CORNELL AERONAUTICAL LAB INC BUFFALO NY.

[28] Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. *arXiv preprint arXiv:1503.02364* (2015).

[29] Yan Song, Shuming Shi, Jing Li, and Haisong Zhang. 2018. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. Association for Computational Linguistics, 175–180. http://aclweb.org/anthology/N18-2028

[30] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* 15 (2014), 1929–1958. http://jmlr.org/papers/v15/srivastava14a.html

[31] 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.

[32] 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.

[33] 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).

[34] Yu Wu, Wei Wu, Dejian Yang, Can Xu, and Zhoujun Li. 2018. Neural response generation with dynamic vocabularies. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[35] Jinxi Xu and W Bruce Croft. 2017. Quary expansion using local and global document analysis. In *Acm sigir forum*, Vol. 51. ACM, 168–175.

[36] Zi Yin, Keng-hao Chang, and Ruofei Zhang. 2017. Deepprobe: Information directed sequence understanding and chatbot design via recurrent neural networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2131–2139.

[37] Qingyu Zhou, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao, and Ming Zhou. 2017. Neural Question Generation from Text: A Preliminary Study. *arXiv preprint arXiv:1704.01792* (2017).