Enhancing Search Query Understanding with Deep Learning

by

Xuefei Han

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Software Engineering & Intelligent Systems

Department of Electrical and Computer Engineering

University of Alberta

Abstract

Search query understanding is a trending topic in the field of Information Retrieval (IR). The goal is to learn higher-level representations for the intents or concepts behind a search query and utilize these representations to further enhance down-stream services like content recommendation. There are several challenges associated with search query understanding. First of which is the Lexical Chase problem, where the surrounding context of a query could not be accurately established by considering only the words in the query. Second, we need an efficient way to build context representations for search queries in the open-domain, which could encapsulate massive amounts of entities and knowledge. Third, we must ensure that down-streams tasks are indeed benefiting from these representations.

The rapid advancements of deep machine learning models introduce new possibilities for tackling these challenges. In this thesis, we begin by investigating whether word-by-word deep generative models provide a unique yet feasible alternative approach for enhancing the process of query understanding and recommendation. We first attempt to directly generate search queries from long news documents, which is of great value to search engines and recommenders in terms of locating potential target users and ranking content. By combining a hierarchical Recurrent Neural Network (RNN) encoder with a sentence-level and a keyword-level Graph Convolutional Networks (GCNs), we build structural document representations. A Transformer based decoder incorporates each feature stream through the Multi-Head Attention mechanism.

Next, we study generative query recommendation from short inputs, e.g. queries and document titles. We partition the task of query generation into two simpler sub-problems, namely, relevant words discovery and context-aware query generation. In the first stage, an RNN-based Relevant Words Generator shortlists a dynamic vocabulary of contextually relevant words, which eases the learning process for the attentional Sequence-to-Sequence (Seq2Seq) model in the second stage. Overall, our proposed framework achieves better performance and alleviates the issue of high resource-consumption in many generative language models.

Finally, we study the problem of relatedness matching between a search query and a large set of high-level concepts. We re-adopt the Relevant Words Generator from previous work as an enhanced shortlisting scheme and meta-fine-tune a BERT matching model for fine-grained relatedness classification. By employing four closely related tasks and training under the Reptile algorithm, we achieve zero-shot transfer learning on the problem of query-concept matching.

On real-world datasets provided by our industry research partner, Tencent, we show that deep learning models learn better representations for search queries, and our approaches competitively outperform many popular baselines. Furthermore, we conduct various ablation tests and case studies to verify the usefulness of each proposed component.

Preface

All datasets used for experimentation are provided by Tencent, which is part of an international research collaboration between Dr. Di Niu's group at the University of Alberta and the Platform Content Group of Tencent.

Chapter 2 of this thesis has been published as Fred X. Han, Di Niu, Kunfeng Lai, Weidong Guo, Yancheng He and Yu Xu."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). I was responsible for model design, experimentation as well as manuscript composition. Dr. Di Niu, who is the supervisory author, assisted with problem formation and contributed to manuscript edits. Weidong Guo, Kunfeng Lai assisted with data collection and problem formation.

Chapter 3 of this thesis has been published as Fred X. Han, Di Niu, Haolan Chen, Kunfeng Lai, Yancheng He and Yu Xu. "A Deep Generative Approach to Search Extrapolation and Recommendation." In Proceedings of *KDD '19*. ACM, 2019. I was responsible for model design, experimentation and manuscript composition. Dr. Di Niu was the supervisory author who assisted with problem formation and contributed to manuscript edits. Haolan Chen assisted with data collection and problem formation.

Acknowledgments

During my study as a Master of Science student at the University of Alberta, I have received enormous guidance and support from my supervisor, **Dr. Di Niu**, who introduced me to the field of machine learning, data mining, and natural language processing. Dr. Niu's expertise and passion motivated me to overcome the obstacles in my research. He also taught me to always be open to new ideas, believe in yourself and never be afraid to explore. I would like to give my sincerest gratitude to Dr. Niu for making my graduate study fulfilling and enjoyable.

I would also like to express my appreciation to all the wonderful people I have met at the Platform and Content Group, Tencent, during my research internship, especially to Haolan Chen, Kunfeng Lai, and Weidong Guo for their attentive help and support. They showed me how the knowledge I learned in class could be applied to real-world problems and improve the experience of millions of users.

Furthermore, I would like to thank my group members, Bang Liu, Mingjun Zhao, Yaochen Hu, Chenglin Li, and many others, for creating a harmonious work environment, and for always be willing to sharing their knowledge and experiences, as well as discussing new potential approaches with me.

Last but not least, I would like to thank my family; my dearest wife, Qianyu Zhang, for her unconditional love and support; my father Hua Han and my mother Xiuyun Wu for their constant trust and encouragement. This thesis would not be possible without them.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

While past research efforts mainly focus on improving the relevance matching between search queries and candidate resources, we argue that to further enhance user experiences in modern search engines, news feed applications and recommendation systems, the key is to develop a better understanding of the high-level concepts behind every search query, which would help uncover more hidden user interests and benefit down-stream services. For example, by issuing the query *RAV4 fuel efficiency*, the user may only want plain fuel economy numbers, but more often, he/she is interested in fuel-efficient SUVs in general. An intelligent recommender would provide more articles under this concept, allowing the user to discover more interesting facts and entities, therefore, leading to an increase in click-through. To achieve such functionality, the recommender should have the knowledge that *RAV4* is an entity under the concept of *fuel-efficient SUVs*, which is extremely challenging due to the vastness of open-domain knowledge.

Fortunately, the advancements of deep learning models may open up new possibilities for improving search query understanding. In this thesis, we employ deep learning to enhance the two crucial steps of search query understanding, namely, query context representation and utilization. In the first step, the goal is to identify the high-level intents or concepts behind every search query, then consolidate them into condensed representations. Deep learning have been proven to be effective at representation learning in the field of Computer Vision [LeCun *et al.*, 2015], Natural Language Processing[Vaswani *et al.*, 2017; Devlin *et al.*, 2018; Peters *et al.*, 2018] and Information Retrieval [Severyn and Moschitti, 2015; Palangi *et al.*, 2016; Liu *et al.*, 2015]. Therefore, we adopt competitive deep models to encode search queries into fix-sized vectors, where each vector serves as a generalizable representation and reflects the overall context. The second step requires us to efficiently utilize the learned representations to improve down-stream tasks. To tackle these challenges, we focus on the recommendation task and study two unique approaches. First, we learn an implicit mapping from context vectors to words with deep generative language models. By directly generating queries in a word-by-word fashion, we improve the recommendation of related

documents and queries. Second, with the help of knowledge conceptualization, we study the matching problem between the representations of a query and a concept. By improving the matching quality, we could discover more related queries for an input search query through its matched concepts and recommend them to the user.

Our evaluations are conducted on two large datasets provided our industry partner Tencent, including 120K instances of query-document pairs and 8 days of click histories from the QQ mobile browser spanning the November and December of 2018. We evaluate our proposed deep learning approaches on these datasets to showcase their competitiveness against other popular baselines. Furthermore, we conduct case studies and human evaluations to verify that the results from our approaches match well with human intuition.

1.2 Research Roadmap

We investigate three concrete problems related to search query understanding. First, since most queries contain only a few words, they are known to suffer from the *Lexical Chase* problem [Riezler and Liu, 2010], where the surrounding context cannot be accurately established by considering only the words in the queries. In the example query from the previous section, we would not know that RAV4 is an SUV, or that it is under the Toyota brand, without sufficient background knowledge. Therefore, we first take a *backward* approach, that is, we work from our resources, e.g. news documents, and attempt to develop a better understanding of search queries by reverse-engineering the most probable queries for them. Chapter. 2 depicts our detailed solution to this problem. The primary benefit of deriving queries from documents is that we are less likely to encounter the Lexical Chase problem because long documents often provide enough context for learning stable representations. However, longer inputs carry more noises and require careful feature extraction, which is the main challenge we tackle in Chapter. 2.

Second, a more conventional approach to address the Lexical Chase problem is Query Expansion (QE), where a system actively searches for more related entities. We follow this idea and design a novel two-stage generative framework for query recommendation. Based on an input query or a document title, a set of relevant words is generated in the first stage, which constitutes a dynamic vocabulary. After this step of context completion, an attentional Sequence-to-Sequence (Seq2Seq) model processes this vocabulary and selects the appropriate words for the final output query. The two-stage setup effectively mitigates the issue of high VRAM consumption and long training time associated with large output vocabularies. Chapter. 3 describes this process in detail.

Third, We study the problem of tagging related high-level concepts to queries in Chapter. 4. A concept could be a keyword, a short phrase or a complete sentence that associates real-word entities under the *isA* relation. We acquire a set of more than 150K concepts from Tencent, which is mined with a combination of bootstrapping, heuristic and machine learning approaches [Liu *et al.*, 2019*b*]. Our goal is to locate the best-matching concepts

for a search query, which would enable us to recommend other queries under the same concepts. Due to the size of the concept set, it would be impossible to manually label sufficient amounts of unbiased data for each concept. This motivates our adoption of the Reptile [Nichol *et al.*, 2018] meta-learning algorithm, under which we meta-fine-tune a BERT [Devlin *et al.*, 2018] matching model and achieve zero-shot transfer learning on the query-concept matching task. We also re-apply the Relevant Words Generator (RWG) from Chapter. 3 and design a novel shortlisting scheme, which is superior at finding the related concepts for a query from a large candidate set, compared to pre-trained wording embeddings [Song *et al.*, 2018*c*] or deep contextualized representations [Peters *et al.*, 2018].

Chapter. 5 summarizes our contributions. We then review unsolved challenges and discuss potential directions for future research.

Chapter 2

Inferring Queries from Documents

2.1 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 chapter, 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.

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 within a document, in G-S2A, we extend the existing hierarchical Recurrent Neural Network (RNN) based encoder [Serban *et al.*, 2016] by novelly incorporating two Graph Convolutional Networks (GCN) [Kipf and Welling, 2016], 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 [Vaswani *et al.*, 2017] 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 [Serban *et al.*, 2016], on all variants of ROUGE [Lin, 2004] and BLEU [Papineni *et al.*, 2002] scores as well as the Exact Match (EM) ratio.

2.2 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 our problem setting, 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 [Sutskever *et al.*, 2014], 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. 2.1 is an example from our training data that best illustrates the differences. We use TextRank [Mihalcea and Tarau, 2004] 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 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

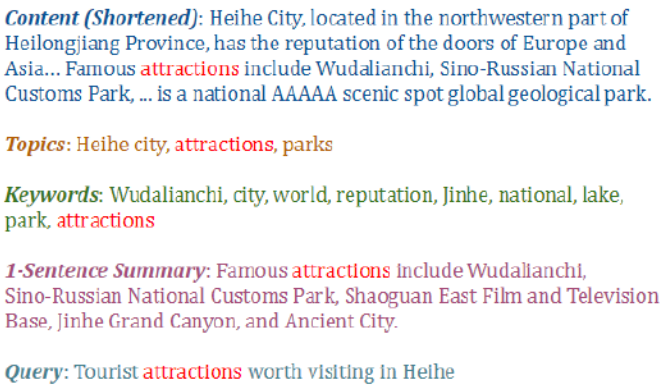

Figure 2.1: An example illustrating the differences between topics, keywords, summaries and queries. Note that the keyword "attractions" appears in the content and all outputs, whereas the rest of the query words are inferred from the context of the article.

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. 2.1. 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. 2.1. 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 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. 2.4, we design our generative query inference model taking these challenges into consideration.

2.3 Review of Related Literature

In this section, we review the literature in closely related fields including text summarization, keyword extraction, query-document matching, topic models, as well as the advancements of Graph Neural Networks (GNN).

Summarization: Traditional summarization approaches like TextRank [Mihalcea and Tarau, 2004], LexRank [Erkan and Radev, 2004], and LSA [Landauer *et al.*, 1998] are extractive based, where a ranking metric is evaluated on the entire document and salient sentences are selected as summaries. With the advancement of deep learning methods, more accurate extractive models like [Nallapati *et al.*, 2017; Cheng and Lapata, 2016; Paulus *et al.*, 2017] are able to learn more useful latent features, and they easily outperform traditional methods. On the other hand, generative summarization models are attracting increased attention due to their potential in generating diverse and interesting summaries. Most representative works on generative models include [Nallapati *et al.*, 2016], which is the first to apply an Encoder-Decoder RNN architecture with an attention mechanism to summarization. And [See *et al.*, 2017] further refines this framework by adding a pointer network to permit copy and a coverage mechanism to prevent output repetition. [Tan *et al.*, 2017] introduces a hierarchical Encoder-Decoder with a graph attention mechanism to better capture multi-granularity features in a document, which offers great inspiration to our work. Other works such as [Liu *et al.*, 2018*c*; Gehrmann *et al.*, 2018; Liu *et al.*, 2018*d*] attempt to improves summary quality from the perspective 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 [Mihalcea and Tarau, 2004; Rose *et al.*, 2010; Liu *et al.*, 2009; Litvak *et al.*, 2011] are all single document-oriented. In other words, in the context of only the current document, they rank all the words according to different criteria and select the top candidates as keywords. They are fast but do not explicitly learn and generalize from past extractions. In contrast, supervised approaches such as [Frank *et al.*, 1999; Turney, 2000] initially model this task as a binary classification problem, where key-phrases are distinguished from non-key-phrases, and the model is able to generalize across different documents. Then, statistical machine learning methods like Naive Bayes [Uzun, 2005], Conditional Random Fields (CRFs) [Zhang, 2008] and Support Vector Machines (SVMs) [Zhang *et al.*, 2006] 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 [Mihalcea and Tarau, 2004; Litvak *et al.*, 2011; Daille, 2013] 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 a graphical representation in our model.

Query-Doc Matching & Query Generation: Another field relevant to our problem

is query-document matching, where the best query for a document is selected from a set of candidate queries. Trivial approaches simply compute an aggregated similarity score between query and document word embeddings. Metrics like cosine similarity, TF-IDF similarity or Okapi BM25 [Robertson *et al.*, 2009]. Many deep learning models have also been proposed, including DSSM [Huang *et al.*, 2013], C-DSSM [Shen *et al.*, 2014], ARC-I [Hu *et al.*, 2014], DeepMatch [Lu and Li, 2013] and MatchPyramid [Pang *et al.*, 2016]. These methods are either representation-based that focus on feature extraction from text, or interaction-based that emphasize pair-wise matching. The MGAN [Zhang *et al.*, 2018] is a novel framework specifically for query-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. On two learn-to-rank datasets, MGAN achieves excellent performance. This work inspires us to employ the Graph Convolutional Network (GCN) [Kipf and Welling, 2016] when building document graphical representations. The only related work on search query generation, to the best of our knowledge, is [Wang *et al.*, 2018]. 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 [Blei *et al.*, 2003; Hofmann, 2000; Arora *et al.*, 2012]. Deep learning based methods have also been proposed, [Maaloe *et al.*, 2015] applies Deep Belief Nets (DBN) to discover topics from digital media content. [Lauly *et al.*, 2017] applies deep neural networks to bag-of-words document representations and achieve state-of-the-art topic modelling performance. [Zheng *et al.*, 2016] 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.

Graph Neural Networks: Graph Neural Networks (GNNs) are a family of neural networks specially designed to capture features from graphs. [Gori *et al.*, 2005; Scarselli *et al.*, 2009] each introduces an early GNN variant. And they both propose to encode a graph in a vertex-by-vertex fashion, where each vertex is represented by its own features, neighbouring vertices and edges. [Li *et al.*, 2015] augments the GNN by employing a Gated Recurrent Unit [Chung *et al.*, 2014] to unroll graph recurrence and to enable sequential outputs. Graph Convolutional Networks (GCNs) [Kipf and Welling, 2016] utilize a fast approximate convolution technique when aggregating vertices. Unlike the Gated GNN, the output of a GCN has the same dimension as the input, which is a desirable characteristic for our work. There are also a number of GNN based generative models, Graph2Seq [Xu *et al.*, 2018] follows the encoder-decoder setup, but replaces the encoder with a GNN and experiments with three light-weight vertex aggregators. [Song *et al.*, 2018*a*] is another model designed to generate sequences from Abstract-Meaning-Representation (AMR) [Banarescu *et al.*, 2013] graphs. The Graph2Seq model proposed by [Venkatakrishnan *et al.*, 2018] tackles a different challenge. It represents vertices as time-series, which is highly scalable

to large graphs or dynamically growing graphs.

2.4 Model

In this section, we present a detailed description of the proposed G-S2A network. Fig. 2.2 is a complete illustration of the model.

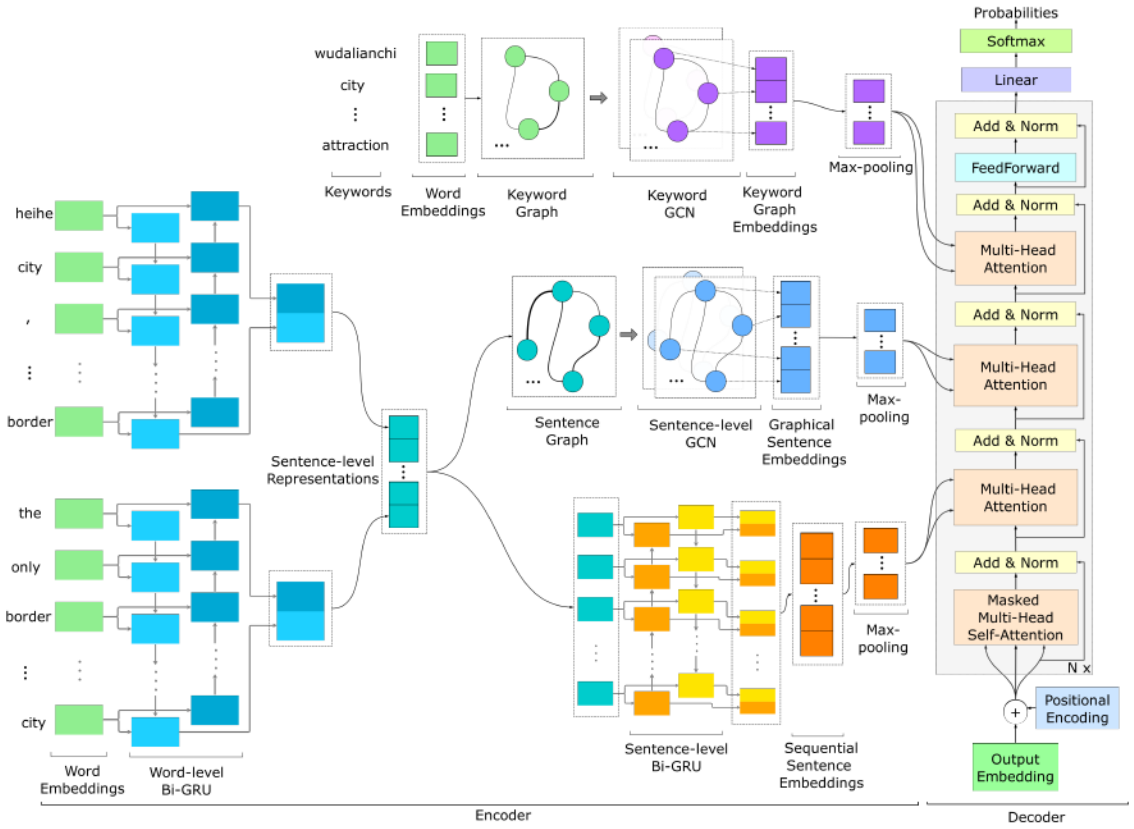

Figure 2.2: Complete overview of our proposed G-S2A model.

2.4.1 Hierarchical Bi-directional RNN

To reveal the sequential structure of a document, we first utilize a hierarchical RNN-based encoder [Tan *et al.*, 2017; Serban *et al.*, 2016] 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. Specifically, given a document D composed of n sentences $\{S_1, S_2, ...S_n\}$, and assume the ith sentence contains j words i.e. $\{w_1, w_2, ...w_j\}$. We generate the single vector representation for S_i using a Bi-directional Gated Recurrent Unit (GRU) [Chung *et al.*, 2014] network. The output of a GRU $\boldsymbol{h}^j$ in one direction for a sequence of words from w_1 upto w_j is expressed as:

$$z^j = \sigma(\mathbf{W}_z \mathbf{x}^j + \mathbf{U}_z \mathbf{h}^{j-1}), \tag{2.1}$$

$$r^j = \sigma(\mathbf{W}_r \mathbf{x}^j + \mathbf{U}_r \mathbf{h}^{j-1}), \tag{2.2}$$

$$\hat{h}^j = tanh(\mathbf{W}_f \mathbf{x}^j + \mathbf{U}_f (r^j \odot \mathbf{h}^{j-1}), \tag{2.3}$$

$$h^j = (1 - \mathbf{z}^j) \odot \mathbf{h}^{t-1} + \mathbf{z}^j \odot \hat{h}^j. \tag{2.4}$$

Where z^j is known as the update gate, r^j is known as the reset gate. σ is the sigmoid activation, $\hat{h}^j$ is the proposed activation and $\mathbf{h}^j$ is the output of the network for sequence j. Also, $\odot$ denotes element-wise multiplication, $\mathbf{W}_{z,r,f}$ and $\mathbf{U}_{z,r,f}$ are trainable weights. For a bi-directional GRU, the input sequence is simply reversed and feed through the network again. In this case the output is produced by concatenating $\mathbf{h}^j_{fwd}$ and $\mathbf{h}^j_{bwd}$.

In our model, we first utilize a word-level Bi-GRU to encode all words in a sentence. We concatenate the last states from two directions into a 1-D sentence representation. Next, we join all the sentence vectors on the first axis to form a 2-D representation for a document. This 2-D vector is then fed into another sentence-level Bi-GRU. The goal here is to learn crucial sequential dependencies among the sentences. For instance, one sentence logically infers the next sentence, or the second sentence refers to entities in the first sentence. We include the classic hierarchical Bi-GRU in our encoder because we agree that natural language is sequential in nature. However, for long documents, graphical representations are also worth exploring, we present those components in the next two sub-sections.

2.4.2 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) [Kipf and Welling, 2016] is partially established on top of the word-level Bi-GRU from Sec. 2.4.1. Specifically, the feature of a vertex input to the GCN is the sentence vector produced by the word-level Bi-GRU. Here, we do not create a graphical representation for a sentence. There are two motivations behind this design. The first motivation is computational efficiency, as it is costly to generate graphs for each sentence. Second, we believe a sentence-level document graph representation, where each node corresponds to a sentence, is the most intuitive. Therefore, we need each node feature to fully capture the meaning of the sentence, which is already achieved by the word-level Bi-GRU, and there may not exist a useful graphical structure in a shorter sentence.

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:

$$H^{l+1} = \sigma(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^{l}W^{l}), \tag{2.5}$$

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 [Kipf and Welling, 2016] for more GCN-related details.

2.4.3 Keyword-Level GCN

As has been mentioned in Sec. 2.2, 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 [Zhang *et al.*, 2018].

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

$$w_{ij} = \frac{1}{d_{ij}^{\text{avg}}} = \frac{m}{\sum_{j=1}^{m} d_{ij}}, \tag{2.6}$$

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.

2.4.4 Transformer Decoder

Our decoder network is a uni-directional Transformer [Vaswani *et al.*, 2017] with two additional attention blocks for the GCN outputs. The Transformer is an alternative approach to RNNs for modeling sequential data. As illustrated in figure 2.2, the input to the decoder are word embedding vectors of partially generated user query. Then, we add positional encoding values to capture word-level location information. These values are pre-computed from sine and cosine waves at different frequencies,

$$PE_{(pos,2i)} = sin(\frac{pos}{10000^{2i/d_{model}}}), \tag{2.7}$$

$$PE_{(pos,2i+1)} = cos(\frac{pos}{10000^{2i/d_{model}}}). \tag{2.8}$$

pos is the word position, i is the ith dimension of the input feature. d_{model} is the total number of dimensions in the input feature.

The core component of the Transformer decoder is the Multi-Head Attention module, which is made up of parallel layers of the Scaled Dot-Product Attention. On 2-D sequential 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 the following,

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V. \tag{2.9}$$

d_k is the feature dimension of K. The output of this operation has the same shape as the input query Q. The difference is Q now incorporates information from value V through attention. Notice that there are no trainable weights here. For the masked self-attention block in figure 2.2, Q, K, V all correspond to the partially generated user query vector. The purpose of the mask is to prevent the decoder from attending subsequent positions during training, which makes the Transformer uni-directional. For the remaining attention modules, Q is the partially generated user query vector, K and V are the output representations from the encoder, specifically, the sentence-level sequential embedding, the sentence-level graph embedding and the word-level keyword embedding. 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 to 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 recreate the original feature dimension and feed through one more layer of feed-forward network. This process is expressed as,

$$\begin{aligned}\text{MH-Attn}(Q, K, V) = g_f(\text{Concat}\{&\text{Attn}_1(g_1^q(Q_1), g_1^k(K_1), g_1^v(V_1)), ... \\ &\text{Attn}_h(g_h^q(Q_h), g_h^k(K_h), g_h^v(V_h))\}).\end{aligned}$$

Attn corresponds to the Scaled Dot-Product Attention. g is a one layer feed-forward network. After completing all the attention operations, the output passes through a Position-wise Feed-Forward Network (FFN), which consists of two linear layers with a ReLU activation in-between,

$$FFN(x) = max(0, W_1 x + b_1)W_2 + b_2. \tag{2.10}$$

We choose the Transformer decoder instead of a traditional RNN-based decoder for three reasons. First, the Transformer decoder is more light-weight since it does not maintain a hidden state. Second, the Multi-Head Attention operations are able to execute in parallel, which means that we can feed the entire output sequence through the network to train one batch. This is much more efficient compared to a traditional RNN decoder, where the decoder learns from the output in a word-by-word fashion. Third, the Transformer is more versatile due to its modularized internal structure, which permits us to incorporate more features easily. In our model, we simply declare two additional Multi-Head Attention modules to attend on sentence-level and keyword-level graphical features.

2.4.5 The Complete Model

We connect the components as depicted by Fig. 2.2. 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 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.

2.5 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.

Table 2.1: 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

2.5.1 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 [Hermann *et al.*, 2015] are not suitable because their summaries are too long to be considered queries. Query-document matching datasets like Ohsumed [Hersh *et al.*, 1994] contains too few queries, whereas datasets with enough unique user queries, such as the NFCorpus [Boteva *et al.*, 2016], 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. Compared with user queries entered from a computer, queries in our dataset are even more concise. 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. 2.1, 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 2.1 lists the important statistical information about our dataset.

2.5.2 Experimental Setup

We adopt the word-overlap based ROUGE [Lin, 2004] and BLEU [Papineni *et al.*, 2002] metrics, which are widely used in summarization and machine translation tasks. Here, 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 [Manning *et al.*, 2014*b*]. 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.

We use the Noam Optimizer [Vaswani *et al.*, 2017] 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. For all RNN baselines we tune their hidden layer sizes and the dropout probability on the development set and choose the best performing model for testing.

TextRank-kw is a simple keyword extractor based on the TextRank [Mihalcea and Tarau, 2004] 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 [Sutskever *et al.*, 2014] with the *general* attention mechanism [Luong *et al.*, 2015]. 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 [Gu *et al.*, 2016], 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 2.1. We believe this baseline is a strong competitor. Similar to the previous baseline, we use a Bi-GRU for the encoder and a Uni-GRU for the decoder.

HRED is a RNN-based hierarchical encoder-decoder model proposed by [Serban *et al.*, 2016]. 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 [Vaswani *et al.*, 2017]. 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.

2.5.3 Results & Discussion

In addition to the metrics mentioned in Sec. 2.5.2, 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 2.2, 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 2.2 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.

2.5.4 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 2.3, 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.

Table 2.2: 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**

Table 2.3: 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

Chapter 3

Generative Query Recommendation

3.1 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. 3.1. Yahoo! offers a similar list of other query recommendations in "Also Try" before all the results. Search recommendation is different from query rewrite [Antonellis *et al.*, 2008; Riezler and

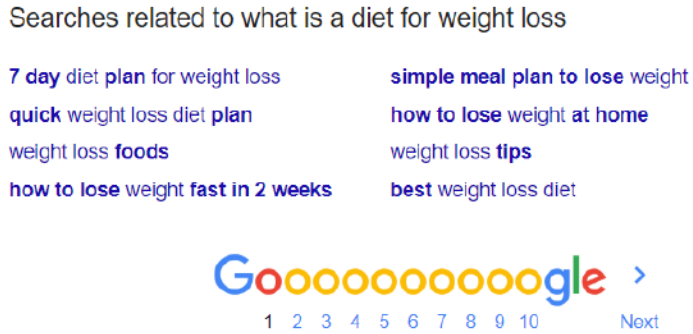

Figure 3.1: An example of query recommendation in the Google search engine.

Liu, 2010; He *et al.*, 2016], 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.

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 [Antonellis *et al.*, 2008; Jeh and Widom, 2002], 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:

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 [Sutskever *et al.*, 2014] 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 a certain degree, CopyNet [Gu *et al.*, 2016] alleviates this issue by encouraging the model to directly copy some words from the input query. However, such verbatim copying seriously undermines the opportunities of query expansion, which causes the model to simply rephrase an input query instead of suggesting related queries that the user will find interesting.

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

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

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 [Vaswani *et al.*, 2017].

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 [Papineni *et al.*, 2002], ROUGE-n [Lin, 2004] and Exact Match (EM) ratio. We show in Sec. 3.5 and Sec. 3.6 that our proposed approach also strikes a good balance between performance, time-complexity, and interpretability. The two-stage division of labor in our proposed approach mitigates the need for a large output vocabulary when generating each query word, which is often the bottleneck of performance and training time in Seq2Seq models [Jean *et al.*, 2014]. In addition, the latent contextual information of a query or document discovered by the RWG model in the first stage can also be transferable to other tasks. Furthermore, We conduct case studies on the generated queries to further illustrate the superiority of the proposed generative learning framework as a solution to related query recommendation.

3.2 Review of Related Literature

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

3.2.1 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 [Sutskever *et al.*, 2014]. Seq2Seq models are proven to be performant in a number of NLP tasks like Automatic Summarization [Liu *et al.*, 2018*c*], Dialogue Systems [Shang *et al.*, 2015] and Reading Comprehension [Zhou *et al.*, 2017; Liu *et al.*, 2019*a*].

The most influential augmentation to the Seq2Seq model is the attention mechanism [Bahdanau *et al.*, 2014; Luong *et al.*, 2015], where the next decoder output is combined

with a weighted sum of encoder hidden representations. The copy mechanism (CopyNet) [Gu *et al.*, 2016] 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. 3.3 that it is a special case of employing a dynamic output vocabulary, where the contextually relevant words are only taken from the input. [Vaswani *et al.*, 2017] 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 [Devlin *et al.*, 2018].

The idea of incorporating a dynamic output vocabulary into generative models has been touched upon by prior researches, with [Wu *et al.*, 2018] being the most relevant work on this subject. The main difference is between [Wu *et al.*, 2018] and our model is that [Wu *et al.*, 2018] 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 [Wu *et al.*, 2018], 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 [Jean *et al.*, 2014; L'Hostis *et al.*, 2016; Mi *et al.*, 2016] 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.

3.2.2 Query Expansion & 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. [Xu and Croft, 2017] is an early work that jointly combines word context features from the query and the retrieved documents. [Cui *et al.*, 2002] is the first work to propose a probabilistic query expansion approach. [Fonseca *et al.*, 2005] relies on association rules to build a query relation graph, then extract relevant concepts to expand on the input query. [Jones *et al.*, 2006] retrieves the expansion candidates by considering word co-occurrences during a click session. [Gao *et al.*, 2012; Gao and Nie, 2012; Riezler *et al.*, 2008] train SMT models to learn word and

phrase correspondence features from large amounts of click-through data.

Direct generation of queries using Seq2Seq models is attracting increased attention. [Liu *et al.*, 2018*d*] incorporates an additional pointer decoder to directly copy words from the input text. [Wang *et al.*, 2018] proposes a multi-tasking Seq2Seq model for title compression and query generation on E-commerce product titles. [Yin *et al.*, 2017] and [He *et al.*, 2016] perform direct query-to-query generations, while [Han *et al.*, 2019] combines a hierarchical Encoder and a Graph Convolutional Network (GCN) to generate queries from long documents.

3.2.3 Query Reformulation

Unlike Query Expansion, Query Reformulation studies the procedure of re-writing a search query into a new query, such that the overall meaning is maintained but the new query is easier for the search engine to process. Query reformation techniques improve the search results displayed to the user and are commonly benchmarked using ranking quality metrics like the nDCG scores [Järvelin and Kekäläinen, 2002]. The earliest query reformulation approaches focus on query term deletion [Jones and Fain, 2003] or substitution [Terra and Clarke, 2004]. Clustering based [Beeferman and Berger, 2000; Wen *et al.*, 2002] and active-learning based [Zhang *et al.*, 2007] methods have also been explored. In the meantime, direct analysis on click graphs is attracting increased attention. SimRank [Jeh and Widom, 2002] is a method for computing query similarities from links within a click graph. [Antonellis *et al.*, 2008] extends SimRank by considering the link weights and supporting similarity evidences. It then utilizes the proposed metric to select re-write candidates to improve the click-through rates on ads. [Riezler and Liu, 2010] uses Statistical Machine Translation (SMT) methods that map query re-writing into a translation problem. [He *et al.*, 2016] employs a Long Short Term Memory (LSTM) [Gers *et al.*, 1999] network to achieve fully generative query re-writing. It demonstrates that deep neural network models outperform traditional statistical machine translation approaches, thus proving their feasibility for the query reformulation task.

3.3 Model

In this section, we review the details of our proposed related search query recommendation framework, as depicted in Fig. 3.2.

3.3.1 Relevant Words Generator

As identified by other works [Riezler and Liu, 2010; Liu *et al.*, 2018*d*], a major issue limiting query understanding is the problem of incomplete context, also known as the *Lexical Chase* problem [Riezler and Liu, 2010]. For search queries, this problem is commonly caused by missing keywords, or unknown Named-Entities. For example, in the query "xs max

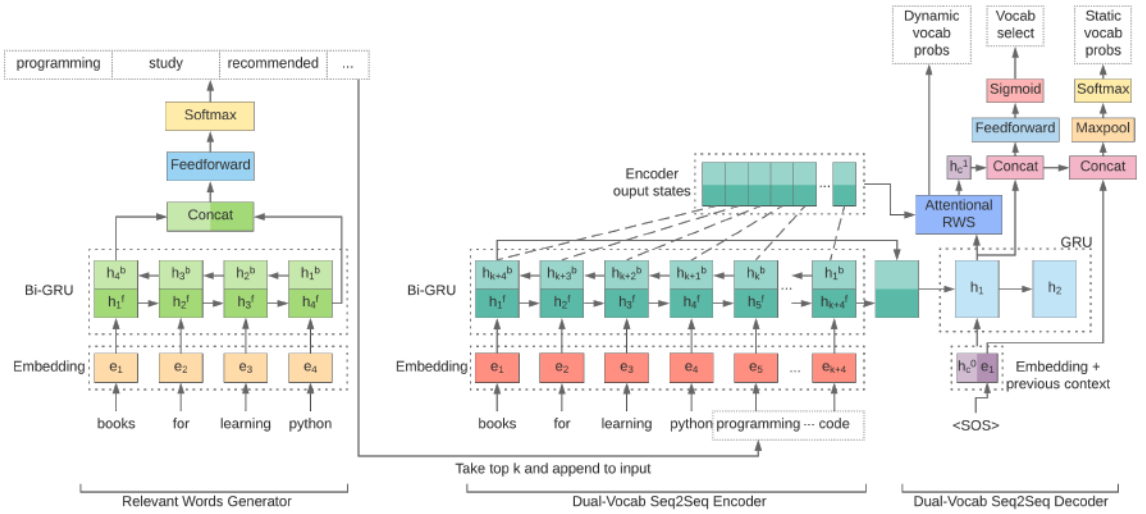

Figure 3.2: Our proposed two-stage generative framework. For simplicity, only one decoding step is shown.

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 "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

$$\theta_{RWG} = \text{argmax}_\theta \prod_{i=1}^{t} P(w_i|Q;\ \theta), \tag{3.1}$$

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) [Chung *et al.*, 2014] 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 [Rosenblatt, 1961] + 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.

3.3.2 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:

$$\text{argmax}_\theta \prod_{i=1}^{h} P(w_i^O | w_{i-1}^O, w_{i-2}^O, ...w_1^O, Q, R_Q;\ \theta). \tag{3.2}$$

The only difference between (3.2) and the traditional Seq2Seq learning objective [Sutskever *et al.*, 2014] 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 $\boldsymbol{e}_{w_i^o\ 1}$ 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 [Bahdanau *et al.*, 2014; Luong *et al.*, 2015; Vaswani *et al.*, 2017] achieves this. A *generalized* version of the attention mechanism can be written as

$$\boldsymbol{c}_i \ = \ \sigma(\ f(\ \boldsymbol{h}_i^{Dec} \cdot \boldsymbol{C}_I^T\ )\ ), \tag{3.3}$$

$$\boldsymbol{h}_i^C \ = \ \sum_{j=1}^{n} \boldsymbol{s}_i^j \boldsymbol{C}_I^j, \tag{3.4}$$

$$\boldsymbol{h}_i^{Dec} \ = \ g(\ \boldsymbol{h}_i^C,\ \boldsymbol{h}_i^{Dec}\ ), \tag{3.5}$$

where $\boldsymbol{h}_i^{Dec}$ denotes the current hidden state of the decoder. σ denotes a Softmax layer. "$\cdot$" represents matrix multiplication. $\boldsymbol{c}_i$ 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 (3.4) indexes the jth element in $\boldsymbol{c}_i$ 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 [Gu *et al.*, 2016] 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, 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 [Gu *et al.*, 2016], we adopt a variant of the Multi-Head Attention proposed by [Vaswani *et al.*, 2017] 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,

$$\boldsymbol{h}_i^{GRU} = GRU(\ [\boldsymbol{e}_{w_i^o\ 1}; \boldsymbol{h}_{i-1}^C]\ ), \tag{3.6}$$

$$\boldsymbol{h}_i^C, \boldsymbol{P}_i^{DV} = RWS(\ \boldsymbol{h}_i^{GRU}\ ), \tag{3.7}$$

$$\boldsymbol{P}_i^{SV} = \sigma f(\ Max_m(\ [\boldsymbol{e}_{w_i^o\ 1}; \boldsymbol{h}_i^C; \boldsymbol{h}_i^{GRU}]\ )\ ), \tag{3.8}$$

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. $\boldsymbol{e}_{w_i^o\ 1}$ 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.

In a decoding step, the GRU learns from $[\boldsymbol{e}_{w_i^o\ 1}; \boldsymbol{h}_{i-1}^C]$. The output of the GRU propagates through the RWS module, during which the probability distribution over the dynamic vocabulary will be generated. Next, we make a large concatenated vector $[\boldsymbol{e}_{w_i^o\ 1}; \boldsymbol{h}_i^C; \boldsymbol{h}_i^{Dec}]$ from 3 vectors, namely, the input word embedding, the attention-weighted context vector and the output of the GRU. To reduce the model size and prevent overfitting, we down-size this vector through a Max-pooling layer. Finally, in a standard fully-connected + softmax projection setup, we get the probability distribution over the static vocabulary.

An important decision for the model is which vocabulary to select for output. After the execution of (3.6) and (3.7), we perform the following operation alongside (3.8),

$$p_i^{cDV} = \text{sig}\ f(\ [\boldsymbol{h}_i^C; \boldsymbol{h}_i^{GRU}]\ ), \tag{3.9}$$

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 [Gu *et al.*, 2016].

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 (3.3) and (3.4) become slow and memory-hungry. Therefore, we adopt the Multi-Head Attention from [Vaswani *et al.*, 2017]. 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. The Multi-Head setup significantly speeds up attention computation with no performance degradation [Vaswani *et al.*, 2017].

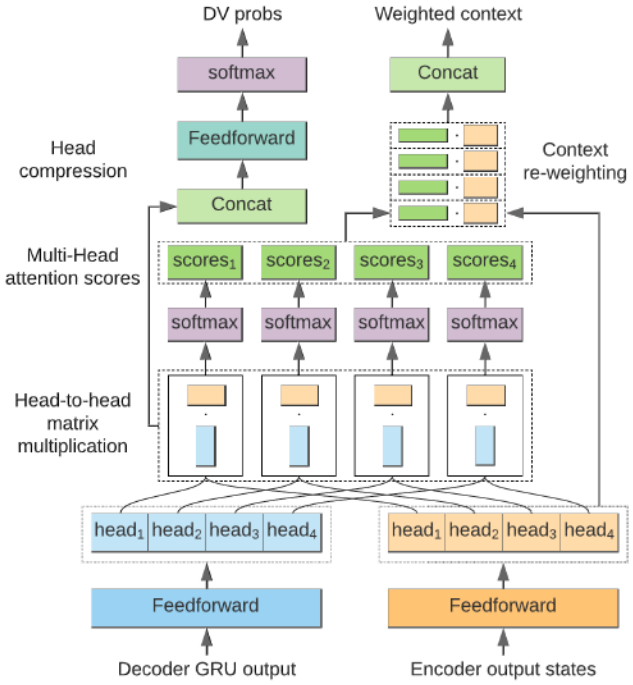

Figure 3.3: An illustration of the Multi-Head Attentional Relevant Words Selector (RWS) module with 4 heads.

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. 3.3. Instead of three inputs [Vaswani *et al.*, 2017], 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.

3.4 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.
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 . 3.4 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.

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

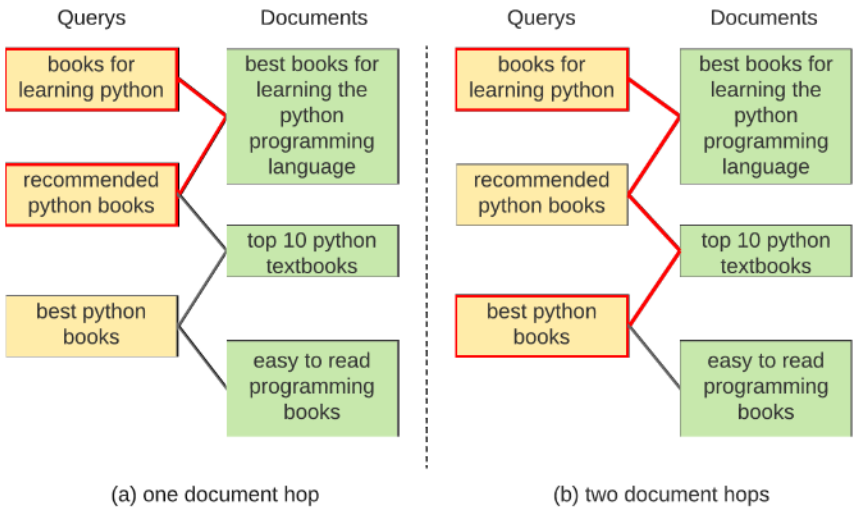

Figure 3.4: An example showing two sibling queries discovered with (a) one document hop and (b) two document hops, as highlighted in red.

- 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,

$$c_{final}(q) = \alpha\ c_{click}(q) + \beta\ c_{len}(q) + \gamma\ c_{overlap}(q), \tag{3.10}$$

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

3.5 Experimentation

We present the detailed experimental procedures [2] and results here. Table. 3.5 records the results for query-to-query generation while Table. 3.6 showcases the results for document

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

Table 3.1: 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

title-to-query generation.

3.5.1 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. 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. 3.4.

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. 3.5 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.

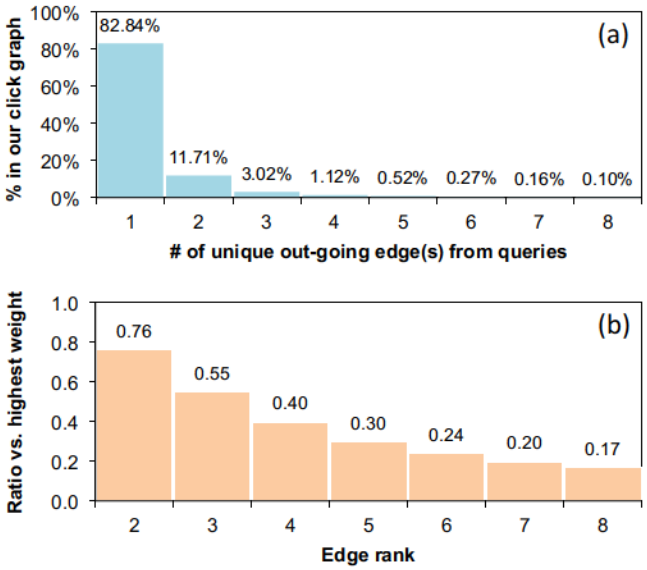

Figure 3.5: Percentage distribution of unique out-going edges from queries (a) and average ratio of secondary edge weights vs. the highest weights (b) in the click graph.

For weights α, β, γ in (3.10), 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. 3.1 reports statistical information on the three generated datasets. Table. 3.2 showcases examples of training data. Observe that all the retrieved words and queries are closely related to the input.

3.5.2 General Experimental Setup

We implement our two-stage framework using PyTorch 0.4[Paszke *et al.*, 2017]. We initialize all word embedding layers with the pre-trained, 200d Tencent AILab Chinese embedding [Song *et al.*, 2018*b*] 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 [Kingma and Ba, 2014] 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.

3.5.3 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. 3.1. 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. 3.3, where $|T|$ is the number of truth relevant words for an input query.

Consider the fact from Table. 3.1 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.

Table. 3.4 provides additional insights on the effectiveness of the RWG model. Observe that for the training data of the query to query generative task, only 43.85% of Out-Of-Vocabulary words, i.e. words that are not in the static vocabulary, appear in the input query, which suggests that even if a CopyNet model learns to copy all OOV words in the input, there would still be more than 60% of unknown OOV words, which would significantly hinder the overall performance. Comparatively, with the RWG model more than 80% of OOV words are covered. As a result, the second stage model would simply have more materials to learn from.

3.5.4 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 [Srivastava *et al.*, 2014] 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.

3.5.5 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 [Sutskever *et al.*, 2014]. 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 [Luong *et al.*, 2015]. We do not vary the vocabulary size for this baseline because the attention mechanism does not operate on the output vocabulary. Therefore, we expect the results of varying output vocabulary sizes to be similar to Seq2Seq.

CopyNet-X: We adopt the CopyNet implementation from [Zhou *et al.*, 2017]. We cannot include a *full* variant here because the copy mechanism is only useful where there are OOV words in the target query.

3.5.6 Evaluation Metrics

We report and compare performance on the following metrics:

BLEU-1, 2, 3, 4: BLEU-n [Papineni *et al.*, 2002] 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[Lin, 2004] 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. Similar to BLEU, higher scores indicate better performance.

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.

3.6 Evaluation

3.6.1 Performance analysis

We begin by analyzing the effect of output vocabulary sizes. From Tables. 3.5 and 3.6, 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. Therefore, it is beneficial to try several output vocabulary sizes when

training generative models, to find the balancing point between the best performance and the least number of OOV tokens.

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 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.

3.6.2 Case Study

We conduct a simple case study on the query-to-query generation task. In Table. 3.7, we compare the outputs from two strong competitors, i.e. CopyNet-40K and RWG+DV-Seq2Seq-40K.

Consider the first two cases in Table. 3.7. 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.

Table 3.2: 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	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	轻卡之家
cargo vehicles catalog	cargo, vehicle, catalog, micro-truck, van, light-truck, semi-truck, boxer-truck, quote,price, series, model, tow-truck	home of light-trucks
英朗2017款	英朗, 2017, 款, 别克, 耗油量, 型号, 汽车, 报价, 发动机, 二手车, 朗逸, 价格, 汽油	别克英朗2017款
Excelle 2017 series	Excelle, 2017, series, Buick, fuel-economy, model, vehicle, quote, engine, used-car, Lacross, price, gas	Buick Excelle 2017 series
vpn安卓下载 vpn Android download	vpn, 安卓, 下载, 官网, 手机, 软件, 苹果, 加速器, 流程, 时间, 设置, 免费 vpn, Android, download, official-site, cellphone, software, Apple, booster, procedure, time, setting, free	免费vpn free vpn
上海医院招聘	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	上海卫生人才网
Shanghai hospital hiring	Shanghai, hospital, hiring, official-site, medical, hygiene, information, nurse, career fair, enroll, job, interview, center, employer, doctor, platform, public hospital, bureau of health	Shanghai medical human resource web
双硬盘怎么装	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	固态和机械硬盘一起用
How to install two hard-drives	dual hard-drive, how to, install, compute, tutorial, installation, motherboard, wiring, SSD, Asus, method, hard-drive, Kingston, solid-state, desktop computer, desktop	using solid-state and mechanical hard-drives together

Table 3.3: 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

Table 3.4: Percentage of OOV words in output that appears in either the input query or input query + top-20 RWG results, with an output vocabulary size of 20K.

	Query to Query			Title to Query		
	Train	Dev	Test	Train	Dev	Test
Input words only	43.85	44.14	43.98	53.75	54.42	54.09
Input + top-20	**81.19**	**56.47**	**56.43**	**59.70**	**58.74**	**58.86**

Table 3.5: 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-120*K*	45.13	30.35	42.47	38.09	31.81	28.19	18.23	0.1498	0.918
Seq2Seq-80*K*	45.70	31.71	43.17	38.81	32.87	29.42	18.73	0.1552	4.36
Seq2Seq-40*K*	45.37	31.29	43.09	39.28	32.89	29.55	19.03	0.1609	9.14
Seq2Seq-20*K*	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-120*K*	48.87	37.88	47.67	46.00	40.27	37.43	29.24	0.2932	2.95
CopyNet-80*K*	55.12	45.03	54.04	52.61	42.27	44.60	35.19	0.3712	1.55
CopyNet-40*K*	56.72	46.66	55.66	54.37	48.95	46.20	36.74	0.3871	4.75
CopyNet-20*K*	55.14	44.19	54.02	52.74	46.77	43.60	34.69	0.3446	7.59
RWG+DV-Seq2Seq-20*K*	58.10	47.05	56.99	55.84	49.74	46.72	37.46	0.3864	2.74
RWG+DV-Seq2Seq-40*K*	**58.25**	**47.93**	**57.16**	**55.99**	**50.37**	**47.63**	**38.22**	**0.4047**	1.88

Table 3.6: 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-120*K*	52.34	35.25	48.77	43.13	36.24	31.67	20.10	0.1512	1.82
Seq2Seq-80*K*	56.60	41.22	53.30	47.64	41.47	37.19	23.18	0.1927	4.34
Seq2Seq-40*K*	56.08	40.66	53.06	47.94	41.40	37.21	23.43	0.1998	8.68
Seq2Seq-20*K*	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-120*K*	58.10	47.05	56.99	55.84	49.74	46.72	37.46	0.3864	2.74
CopyNet-80*K*	68.97	60.50	67.91	66.54	62.12	59.64	47.51	0.5244	1.90
CopyNet-40*K*	72.11	63.82	71.16	69.97	65.54	62.98	50.52	0.5577	3.50
CopyNet-20*K*	71.65	62.64	70.73	69.71	64.79	61.76	49.66	0.5350	6.21
RWG+DV-Seq2Seq-20*K*	72.0	63.34	71.17	70.25	65.47	62.63	50.62	0.5502	4.85
RWG+DV-Seq2Seq-40*K*	**72.75**	**65.18**	**71.93**	**71.00**	**66.90**	**64.61**	**52.32**	**0.5840**	2.89

Table 3.7: Queries generated by CopyNet-40K and RWG+DV-Seq2Seq-40K, in the query-to-query generation task.

Models	Target query	Input 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 recommendations	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 processors	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

Chapter 4

Matching Queries to Concepts

4.1 Introduction

4.1.1 Knowledge Conceptualization

In this chapter, we propose a matching-based approach to discover all related high-level concepts for a search query. The formal definition for concepts is first introduced in [Liu *et al.*, 2019*b*]. To better fit the context of matching, we summarize and rephrase the formal definition here as follows:

- A concept is a short text label which could be a keyword, a short phrase or a short sentence.
- A concept associates one or more text entities under the *isA* relation.
- A query or a document could be tagged with multiple matching concepts.
- Concepts are extracted from news documents and search queries with a combination of bootstrapping, heuristic, as well as supervised learning methods [Liu *et al.*, 2019*b*]. The concept set considered in this chapter contains 159,148 unique concepts.
- A concept is comprehensive enough such that it covers many matching text entities. A concept is also specific enough so that it *only* covers entities of the same *isA* relation.

For example, search queries containing specific bands of fuel-efficient SUVs falls naturally under the concept of *fuel-efficient SUVs*, and search queries like "Resident Evil film 2018" match well with the concept of *zombie movies* or the concept of *popular movies in 2018*. Conversely, topic words such as "movies", "SUVs" or "entertainment" are not useful concepts under the above definition. Because of their broadness, they cover a massive amount of entities and lose the ability to embody useful knowledge. Ultimately, the goal behind conceptualization is to create a tractable abstraction for the entities in the open-domain. Under such an abstraction, the problem of search query understanding translates to finding all the matching concepts, i.e. a text matching problem. Down-stream tasks like

recommending related queries are also significantly simplified by this abstraction. Given an input query, we would select other queries from its matched concepts for recommendation.

One might argue that there are many existing approaches to the problem of knowledge abstraction, such as topic modelling, keyword/key-phrase extraction, and structured Knowledge Graphs [Auer *et al.*, 2007]. Next, we compare conceptualization to these approaches to demonstrate why it is the more suitable method for enhancing query understanding.

Conceptualization vs. Topic modelling: As we mentioned before, one major drawback of topics is their broadness. Topics like "entertainment", "sports", or "games" could potentially cover millions of articles or queries, which do not provide sufficient resolutions for the down-stream tasks. Take query recommendation as an example; a user browsing for sports cars may not be interested in economy cars, even though both concepts fall under the topic of cars. Another way to relate conceptualization to topic modelling is that concepts could be visualized as more refined topics. However, as the number of topics increases, many topic modelling techniques [Blei *et al.*, 2003] fail due to the curse of dimensionality.

Conceptualization vs. Keyword/key-phrase extraction: While words in a concept are likely to be keywords in the matching document or query, the inverse is usually not true. A long document may contain many keywords or key-phrases, but not all of them would appear in a matching concept or even be related to the concept. For example, an article under the concept of "hot zombie movies" may contain relevant keywords like "film" or "horror", but there may also be unrelated keywords like "directors" or "box-office". Readers of this article are more likely to be interested in the zombie movies versus the directors. If we employ keyword/key-phrase extraction instead of concepts, the down-stream tasks would need to further prune the irrelevant keywords, which could hurt the overall performance.

Conceptualization vs. Knowledge Graphs: Structured knowledge graphs like DBPedia [Auer *et al.*, 2007] indeed provide a cleaner and more accurate knowledge representation. However, this comes at the cost of more time and resource spend on knowledge cleaning and graph construction. Additionally, most knowledge graphs are limited in which they only capture concrete facts and relations from well-structured Wikipedia sites. In contrast, the web is continuously evolving at a rapid pace, where a popular search query today may be forgotten entirely in the next month. Knowledge graphs are excellent at presenting clean entities under formalized relations, but not at adapting to constant changes in user interests. For example, if a new dance move suddenly goes viral, conceptualization would enable us to create a concept for it and immediately cover matching queries and articles. We could also discover connections between it and other entities by analyzing concepts under the same article or query. In comparison, a knowledge graph would first need to create a new entity. Then, for this new entity to be useful for down-stream tasks, the knowledge graph must work out its relations to other entities, which could be extremely time-consuming.

4.1.2 The Query-Concept Matching Problem

Although knowledge conceptualization simplifies search query understanding into a text matching problem, it is still unrealistic to directly match every input against all 159,148 concepts. Therefore, we must adopt a coarse-to-fine approach where we first shortlist a much smaller set of candidates based on the input query, then, the matching model determines the relatedness between the input and each candidate concept. However, in doing so, we introduce another potential performance bottleneck. If the scheme is unable to select the most related concepts, it would cripple the performance of the entire system. Utilizing pre-trained word embeddings, conventional shortlisting schemes pre-compute a vector representation for each candidate and rely on fast similarity searches [Johnson *et al.*, 2017] to retrieve the potential candidates for an input. We argue that pre-trained, general-purpose word embeddings are not the most suitable representations for concepts. Because these embeddings are trained on a large corpus of Wikipedia or news data, they only capture the most general degree of relatedness among words. Additionally, word embeddings do not incorporate the context of the surrounding words or the sequence, which results in inaccurate representations. For instance, in the concept "hot zombie movies", it is clear that the word "zombie" and "movie" should contribute more to the concept representation. With pre-trained word embeddings, we could not achieve such context-dependent term weighting.

To improve the shortlisting stage, We re-adopt the Relevant Words Generator (RWG) model from Chapter. 3. Given an input query, we retrieve the top-k relevant concept words and select concepts with the most word-overlaps. Intuitively, the RWG model functions as an adaptation layer on top of pre-trained word embeddings, and it has three advantages compared to the conventional scheme. First, the RWG model learns a projection between queries and concept words, which achieves a preliminary form of query understanding. Second, the output space of RWG is much smaller compared to the pre-trained word embeddings because we only need to project over words that appear in concepts. As a result, training is fast and resource-efficient. Third, since the RWG model explicitly outputs k relevant concept words, it provides better interpretability to the end-users, i.e. end-users could understand why a concept is chosen as a potential candidate by examining the generated relevant words.

In the matching stage, the goal is to predict the relationship between a query and a concept as either related or not related. We formally define relatedness as,

Definition 1 *A search query is related to a concept only if there exists a isA relationship among them.*

For example, "Toyota RAV4" *is-a* "fuel-efficient SUV". Due to the number of candidate concepts, it is impossible to manually-label sufficient amount of unbiased training data. Inspired by [Finn *et al.*, 2017; Nichol *et al.*, 2018], we meta-fine-tune a pre-trained BERT model on 4 highly-similar tasks, namely, query-query matching, query-title matching, title-title matching and the matching between a query and a relevant word. Utilizing the Reptile

algorithm [Nichol *et al.*, 2018], we achieve zero-shot, two-way classification on the problem of query-concept matching.

4.2 Review of Related Literature

Our work draws inspirations from several trending fields. In this section, we begin by reviewing related literature on vectorized word representations, which is highly-relevant to our RWG model. We then survey influential works on text matching and meta-learning.

4.2.1 Word Representations

Representing words by vectors is a classic research problem in the field of Natural Language Processing. The main idea is that we train each word in a large corpus to have a vector representation that is more similar to its surround words. Word2Vec [Mikolov *et al.*, 2013] is one of the most influential works on word embedding learning. It leverages Continuous Bag-Of -Words (CBOW) and Skip-Gram (SG) models as well as sub/negative sampling to learn generalizable representations. Glove [Pennington *et al.*, 2014] embeddings are learned by combining global matrix factorization methods with local context windows. FastText [Bojanowski *et al.*, 2017] augments the Skip-Gram (SG) model with sub-word information which results in faster training. While the Tencent AI Lab embedding [Song *et al.*, 2018*b*] focuses on adding directional information about neighboring words to the Skip-Gram (SG) model. Other approaches like [Faruqui *et al.*, 2014; Kiela *et al.*, 2015; Nguyen *et al.*, 2016; Yu and Dredze, 2014] enhance the learned embeddings by utilizing external knowledge or adding supervised objectives.

One issue that is not addressed by the pre-trained word embeddings is that the same word might have different meanings under a different context. To this end, contextualized word embeddings are proposed where the representation for a word is influenced by other words in the sequence. [McCann *et al.*, 2017] directly adopts an LSTM encoder from a sequence-to-sequence model trained on neural machine translation as a context enhancer for pre-trained word embeddings. ELMo [Peters *et al.*, 2018] is another popular approach where a Bi-LSTM is trained as with a language model objective, and contextualized word embeddings are derived from its hidden states. Both of these works are similar to the idea behind our RWG model. The main difference is that our RWG model is designed for contextualized word generation where the output vocabulary is usually much smaller compared to the vocabularies of general-purpose word representations.

4.2.2 Text Matching

Based on their internal structures, text matching models could be classified as either representation-based or interaction-based. The focus of a representation-based matching model is to improve feature extraction. Several layers of neural networks are constructed

on top of each input vector to extract the most salient information, then an aggregation module output the final score by combining the input representations. The Deep Semantic Similarity Model (DSSM) [Huang *et al.*, 2013] constructs 5 layers of feedforward neural network to process each input, then the cosine similarity between the input representations is returned as the matching score. CDSSM [Shen *et al.*, 2014] replaces the feedforward layers of DSSM with convolution layers to improve generalization. MultiGranCNN [Yin and Schütze, 2015] enables the comparison of multigranular representations. CNTN [Qiu and Huang, 2015] models the interaction of encoded sentence vectors through an additional tensor layer. Later research works adopt a Siamese-architecture [Liu *et al.*, 2018*b*; Mueller and Thyagarajan, 2016] for pairwise representation learning, where a shared convolutional or recurrent feature extractor further improves generalization.

In contrast, interaction-based matching models first establish the crossing of input features, which enable one input to incorporate useful information from another. Next, deep feature extraction layers are employed to capture the most insightful interactions and derive the output score. In pairwise matching tasks, interaction-based models like ARC-II [Hu *et al.*, 2014], MatchPyramid [Pang *et al.*, 2016], DRMM [Guo *et al.*, 2016] and MIX [Chen *et al.*, 2018] often construct matching matrices between input features. Next, several layers of convolution and feedforward operations are performed over these matrices to select the most salient interactions. To the best of our knowledge, there has been no concrete conclusion on the superiority of representation-based or interaction-based methods.

Alternatively, We could classify text matching models by the length of the targeted inputs. While most models are designed for the matching between short text inputs, such as matching queries to document titles, there exist other families of models that facilitate the matching for short vs. long, or long vs. long text inputs. For example, [Zhang *et al.*, 2018] directly matches search queries to news documents, and [Liu *et al.*, 2018*a*] accomplishes matching between long news articles with Graph Convolutional Networks.

Based on the type of outputs, text matching models perform either semantic or relevance matching, where semantic matching models such as DSSM, CDSSM, ARC-II and MIX output a score indicating the overall similarity between a pair of inputs. Relevance matching instead focus on the ranking of inputs, e.g. ranking documents according to their relevance to a query for Ad-hoc retrieval. DRMM [Guo *et al.*, 2016] is a representative model for relevance matching, where the local-interactions between terms in a query and a document is mapped to fixed-length matching histograms. Next, a deep feedforward network learns the hierarchical matching patterns and computes a relevance score. Our meta-fine-tuned matching model classifies the relatedness between a query and a concept, which is a binary semantic matching problem since the similarity score is either 0 or 1.

General-purpose deep language models like BERT [Devlin *et al.*, 2018], ERNIE [Sun *et al.*, 2019], GPT-2 [Radford *et al.*, 2019] and XLNet [Yang *et al.*, 2019] attain revolutionary progress on Natural Language Understanding. On several benchmark tasks, their

performance exceeds human intelligence [Devlin *et al.*, 2018]. The success of these models proves that the pre-training of deep neural networks on large open-domain corpus helps the extraction of generalizable linguistic knowledge. Therefore, we could effortlessly transform a pre-trained deep language model into a text matching model by fine-tuning on labelled matching data.

4.2.3 Meta-Learning

Meta-learning studies the problem of learning how to learn [Thrun and Pratt, 2012; Naik and Mammone, 1992]. Early works focus on the design of meta-trainers, i.e. a model that learns how to train another model such that it performs better on a given task [Schmidhuber, 1992; Bengio *et al.*, 1992]. [Andrychowicz *et al.*, 2016] transfers this idea to deep neural networks and proposes an optimizer-optimizee setup, where each component is optimized by gradient-descent. [Li and Malik, 2016] follows a guided policy search strategy and automatically learns the optimization procedure for updating a model. Meta-learning is also studied as a promising solution to few-shot classification problems, where a model learns to recognize new classes given a limited amount of training data for each class. [Ravi and Larochelle, 2016] proposes an LSTM meta-learner to learn an optimization procedure for few-shot image classification. [Li *et al.*, 2017] instead develops an SGD-like meta-learning process and also experiment on few-shot regression and reinforcement learning problems. MAML [Finn *et al.*, 2017] is another popular approach which does not impose a constraint on the architecture of the learner. Finally, Reptile [Nichol *et al.*, 2018], i.e. the approach adopted in this chapter, simplifies the learning process of MAML by conducting first-order gradient updates on the meta-learner.

Meta-leaning could also be achieved with non-parametric methods. [Koch *et al.*, 2015] trains a Siamese network to classify whether two images are from the same class, which implicitly functions as a meta-learner. During meta-testing, the network compares the new, unseen input to each input in the meta-training set. Matching networks [Vinyals *et al.*, 2016] refines this idea by imitating the meta-testing procedure during meta-training, where the learned embedding of a test input is compared to embedding vectors of inputs from known classes, and the most matching class is selected by the method of weighted k-nearest-neighbors. Prototypical Networks [Snell *et al.*, 2017] learn an entirely new metric space for comparing the similarities between inputs. [Sung *et al.*, 2018] proposes that separating embedding learning and relation matching further improves the performance.

4.3 Framework

We first formalize the learning objective for the problem of search query understanding under knowledge conceptualization. Suppose that we are given a fixed set C consisting of m unique concepts, i.e. $C = \{c_1, c_2, ...c_m\}$. For a search query q drawn from a distribution $p(q)$, suppose that there exists a non-empty set $C_t^q \subset C$, where each concept in C_t^q is a

match to q according to Definition. 1. Our goal is to learn a model f such that,

$$\arg\max_{f} \prod_{c_i \in C_t^q} p(c_i|q, C;\ f) \implies \arg\max_{f} \sum_{c_i \in C_t^q} \log p(c_i|q, C;\ f). \tag{4.1}$$

The main issue when learning f under Eq. 4.1 is that the cost of training scales linearly with m, which is why a shortlisting mechanism is needed when m is large. Let us define a shortlisting model g, which takes in the query q and the complete concept set C, then outputs a new set $C_s^q \subset C$. Mathematically, g solves for the following problem,

$$\min_{|C_s^q|} \max |C_t^q \cap C_s^q|,\ |C_t^q| \leq |C_s^q| \ll |C|, \tag{4.2}$$

which states that an useful g retains more matching concepts in C_s^q. In reality, g is often created without any training, and it relies on pre-computed vector representations for fast similarity search. In our case, the most conventional way to setup g is to utilize pre-trained word embeddings and pre-compute the Mean-of-Word-Embeddings (MoWE) representation for a concept c.

With the addition of g, the original objective from Eq. 4.1 becomes more tractable since $|C_s^q| \ll |C|$, and it is expressed as,

$$\arg\max_{f} \sum_{c_i \in C_t^q} \log p(c_i|q, C_s^q;\ f). \tag{4.3}$$

To transform Eq. 4.3 into a text matching problem, we define the relatedness between a concept and a query as r. In our problem setting, r is a binary label of either 0 or 1. And the objective becomes,

$$\arg\max_{f} \sum_{c_i \in C_t^q} \log p(r_{q,c_i}|q, c_i;\ f). \tag{4.4}$$

Figure. 4.1 depicts our training and testing procedure. To better locate candidate concepts, we learn a Relevant Words Generator model to enhance the shortlisting procedure. Since there is no labelled data for directly learning f, we meta-learn an approximation to f in the second stage. In the end, our two-stage framework is evaluated on the problem of query-concept matching.

4.3.1 Shortlisting by Relevant Words

As we mentioned before, a major weakness of pre-trained embedding-based schemes is that the global context is not taken into consideration. Although deep contextualized representations such as ELMo [Peters *et al.*, 2018] already addresses this weakness, we argue that most contextualized word embeddings are trained on Wikipedia or news data and do not generalize well to the query-concept matching problem. Therefore, we learn an additional adaptation model on top of pre-trained word embeddings to provide more accurate representations.

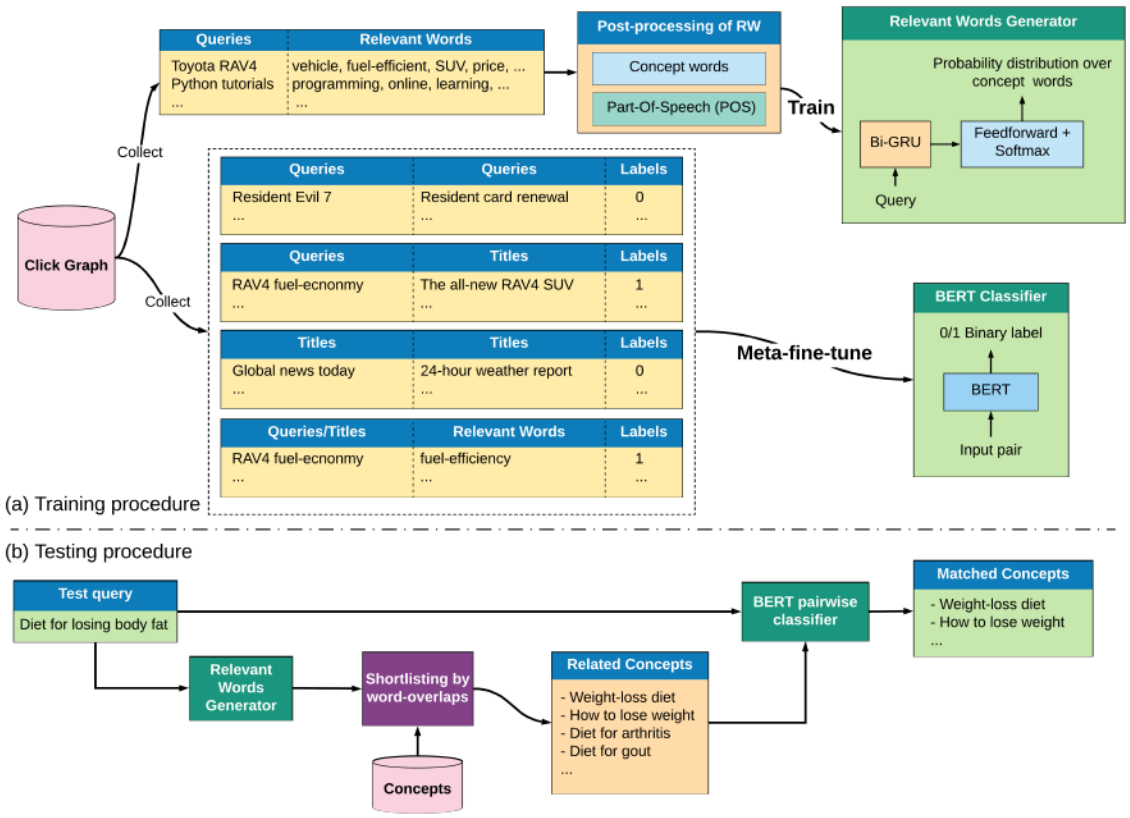

Figure 4.1: The training (a) and testing (b) procedures for our query-concept matching framework.

We first extend the formal definition for a concept c, where c is further partitioned into a sequence of words, i.e. $c = \{w_1^c, w_2^c, w_3^c, ...\}$. Similarly, a query q is represented by $q = \{w_1^q, w_2^q, w_3^q, ...\}$. Next, let V_C be a vocabulary of all the unique words that appear in concepts. We observe that when the number of unique concepts, m, is sufficiently large, $m > |V_C|$. This is an intuitive observation since if m is sufficiently large, when we add a new concept to C there is a high probability that all of its words are already in V_C. In other words, $|V_C|$ does not increase linearly with m. Therefore, as opposed to directly selecting candidate concepts from C, we approach from a different perspective and search for the most related concept words in $|V_C|$. Fortunately, we addressed a similar problem in Section. 3.3.1, where a Relevant Words Generator (RWG) is introduced to tackle the *Lexical Chase* problem. We propose that to perform shortlisting on $|V_C|$ is equivalent to performing context-completion for the input query in the domain of V_C. Together, these factors motivate us to apply the RWG model here as the very first step of concept shortlisting.

We now define the learning objective for the RWG model in our problem setting. Given an input query q of n words, i.e. $q = \{w_1^q, w_2^q, ..., w_n^q\}$, and $|V_C|$. We learn θ such that the log probabilities of the words which are relevant to q, are maximized. Suppose that the

set R^q contains all the true relevant words of q, we could then learn θ with the following objective,

$$\arg\max_{\theta} \sum_{w_c \in R^q} \log p(w_c|q;\ \theta), \tag{4.5}$$

where the generated relevant words i.e. w_c, are limited to only words that appear in $|V_C|$.

As indicated by Figure. 4.1, the Relevant Words Generator is constructed the same way as in Section. 3.3.1. Where a Bi-directional Gated Recurrent Unit (GRU) [Chung *et al.*, 2014] encodes a query word-by-word to learn its context representation, from which a fully-connected projection layer followed by a Softmax operation produces a probability distribution over all words in V_C. In essence, the RWG model learns an explicit mapping from query context vectors to concept words, and it is analogous to the way humans think. For example, when a knowledgeable person hears the query "Resident Evil", highly-relevant words like "game", "zombie", "film" would immediately pop up in his/her mind. Furthermore, if the same person is then asked to find the most matching concepts for the query, he/she would naturally choose the concepts that have the most overlaps with the aforementioned relevant words, like "zombie games" or "zombie films". Consequently, to shortlist concepts, we adopt a simple overlap-based matching approach.

Let us define $\hat{R}^q_k$ as the set of predicted relevant words for a query q, which is constructed by taking the top-k results from the output probability distribution of the RWG model. We then define $\hat{C}^q_k$ as the set of all concepts where each concept has at least one word that is in $\hat{R}^q_k$. Finally, we define $\hat{C}^q_s$ as the output set of candidate concepts, where $\hat{C}^q_s \subseteq \hat{C}^q_k$. For each concept $c \in \hat{C}^q_k$, we define the overlap score as the percentage of unique words in c that are also in $\hat{R}^q_k$, i.e.

$$score(c) = \frac{1}{|\{c\}|} \sum_{w_c \in \{c\}} \lambda(w_c, \hat{R}^q_k), \tag{4.6}$$

where $\{c\}$ denotes the set of unique words in c and $\lambda(w_c, \hat{R}^q_k)$ is simple indicator function which returns 1 if w_c is in $\hat{R}^q_k$ and 0 otherwise. To get $\hat{C}^q_s$, we rank the candidates in $\hat{C}^q_k$ by their overlap scores, and filter out candidates with scores less than a predefined threshold γ, which is expressed as,

$$\hat{C}^q_s = \{c \mid c \in \hat{C}^q_k,\ score(c) \geq \gamma\}. \tag{4.7}$$

The general objective of shortlisting (Eq. 4.2) is stable only if C^q_t is non-empty, which is an over-optimistic assumption in real-world applications. In fact, no matter how comprehensive C is, there would always be queries that are not related to any concepts. Hence, in addition to pruning unrelated concepts for an input query, another important goal is to reject such outliers. Compared to conventional schemes, our approach is also superior in achieving this goal. For the ill-formed query "tree chair doctor", the RWG model is unable to learn a stable context, which means the $\hat{R}^q_k$ is more likely to contain completely irrelevant words like "hospital", "table", or "forest", and concepts in $\hat{C}^q_k$ would have limited

word-overlaps with $\hat{R}_k^q$. Setting a large γ would filter out all the candidate concepts, which effectively rejects the ill-formed query. To handle the same ill-formed query in a conventional scheme, one would need to define a similarity threshold and reject the input if no concepts could meet this threshold. We argue that it is a lot trickier to set a generalizable similarity threshold than γ, and we show in Sec. 4.5 that simply setting $\gamma = 1.0$ produces competitive results.

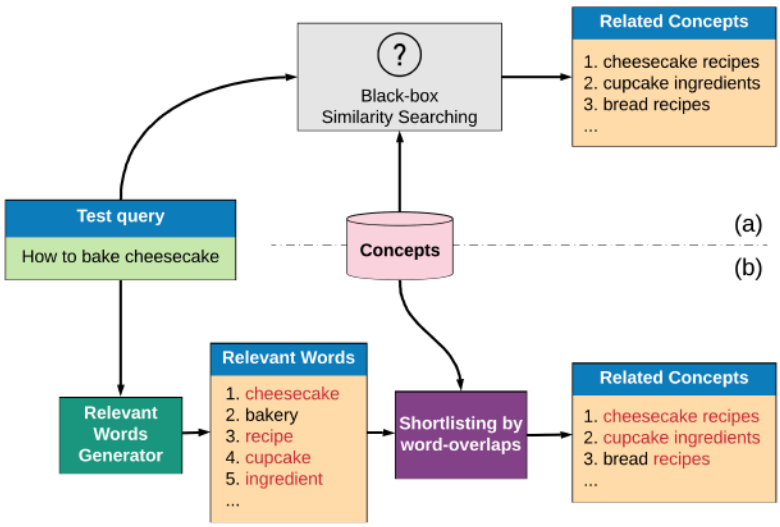

Figure 4.2: Comparison of the interpretability of conventional shortlisting schemes (a) vs. our approach (b). Relevant words highlighted in red help provide more insight on selection process of candidate concepts.

Last but not least, our approach further enhances the interpretability of the shortlisting process. Consider the comparison in Fig. 4.2. Conventional method (a) does not provide any clues on the selection and ranking process. For example, it is difficult to understand why the concept of "cupcake ingredients" is more similar to the input than "bread recipes". In contrast, by explicitly generating a set of relevant words, our approach offers more evidence. End-users would know that "cheesecake recipes" and "cupcake ingredients" is ranked higher in the output set because all of their words are highly-relevant to the input query.

4.3.2 Meta-Learned Matching Model

Without labelled training data, we could not directly learn f under Eq. 4.4. Fortunately, since a significant portion of concepts is extracted from click histories [Liu *et al.*, 2019*b*], it is reasonable to assume that there exist distributional similarities between the task of query-concept matching and other tasks derived from a click graph, e.g. query-query relatedness matching. For this reason, we take a meta-learning approach and learn a model ϕ such that ϕ performs well on all matching tasks derived from a click graph.

A key assumption behind optimization-based meta-learning algorithms [Finn *et al.*,

2017; Nichol *et al.*, 2018] is that under a machine learning problem, there exists a distribution of tasks $p(\mathcal{T})$. And it is possible for a model ϕ to adapt to $p(\mathcal{T})$ as opposed to a sampled task $\mathcal{T}_i$, by minimizing the following loss function [Finn *et al.*, 2017],

$$\min_{\phi} \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}(\phi - \alpha \nabla_{\phi} \mathcal{L}(\phi, \mathcal{D}_i^{train}), \mathcal{D}_i^{test}), \tag{4.8}$$

where $\mathcal{D}_i^{train}$ and $\mathcal{D}_i^{test}$ are the train and test set for the i-th task, and α is the meta-learning rate. The intuition behind Eq. 4.8 is that we meta-learn a model ϕ, by learning how training ϕ on a task $\mathcal{T}_i$ affects its generalizability on a held-out test set from the same task. We repeat this process on several tasks sampled from $p(\mathcal{T})$ to learn ϕ. A successful ϕ would have low test error on every task, i.e. ϕ adapts well to the underlying distribution $p(\mathcal{T})$.

The main advantage of model ϕ is that for an unseen task $\mathcal{T}_j$ sampled from $p(\mathcal{T})$, we could directly transfer ϕ to $\mathcal{T}_j$ and expect decent performance from it. If a small amount of labelled data is available for $\mathcal{T}_j$, ϕ also provides the best possible starting point for fine-tuning a new model $\hat{\phi}$, which is known as the *adaptation* [Finn *et al.*, 2017],

$$\hat{\phi} \;=\; \arg\max_{\phi} \; \log p(\phi | \mathcal{D}_j^{fine-tune}, \; \phi). \tag{4.9}$$

With a large number of concepts, it becomes impossible to manually label an unbiased dataset, even for fine-tuning purposes. Consequently, we do not perform any adaptation here and use ϕ for the zero-shot classification of query-concept pairs.

In Eq. 4.8, the ∇ symbol suggests that to minimize this loss by gradient-descent, we need to perform a second-order differentiation, which often imposes a burden on training, especially in deep models. For this reason, we utilize Reptile [Nichol *et al.*, 2018], a first-order meta-learning algorithm to learn ϕ. Reptile simplifies Eq. 4.8 by converting the outer loss function into a step in the direction of $\nabla_{\phi} \mathcal{L}(\phi, \mathcal{D}_i^{train})$ after K batches of training on $\mathcal{T}_i$. Reptile speeds up learning and eliminates the need for test sets, i.e. $\mathcal{D}_i^{test}$ in Eq. 4.8. Similar to conventional supervised training, Reptile iteratively updates a model, where every iteration is made up of a task-learning phase followed by a meta-learning phase. In the task-learning phase, we first make a copy of the current model, then, we sample a new task and train the copied model by performing K steps of gradient-descent on this task, where $K > 1$. In the meta-learning phase, we update the original model by the difference between its current weights and the weights of the copied model after task-learning, which is expressed by,

$$\phi \;\leftarrow\; \phi + \alpha(\tilde{\phi} - \phi), \tag{4.10}$$

where $\tilde{\phi}$ is the copied model after the task-learning phase and α is the meta-learning rate. It is also worth mentioning that if $K = 1$, Reptile is equivalent to sequentially training a model on all available tasks. However, for all $K > 1$, Reptile converges to entirely different solutions [Nichol *et al.*, 2018].

Under the definition of Reptile, the objective for query-concept matching becomes the objective for the task-learning phase,

$$\arg\max_{\tilde{\phi}} \sum_{u_i,u_j \in U} \log p(r_{u_i,u_j}|u_i,u_j;\ \tilde{\phi}), \tag{4.11}$$

where u_i, u_j and U denote the two input candidates and the training data for the current task, respectively. r_{u_i,u_j} is still a label of 0 or 1. The meta-learning objective for our problem setting is

$$\arg\max_{\phi} \log p(\phi|\boldsymbol{\mathcal{D}}^{meta}), \tag{4.12}$$

where $\boldsymbol{\mathcal{D}}^{meta} = \{\mathcal{D}_1^{train}, \mathcal{D}_2^{train}, ...\}$, i.e. it encapsulates all the tasks used for training ϕ.

To establish a good starting point for learning ϕ, we utilized the pre-trained BERT model [Devlin *et al.*, 2018], which is shown to capture generalizable linguistic knowledge that could be easily transferred to other tasks through fine-tuning. BERT contains 12 layers of Transformer [Vaswani *et al.*, 2017] blocks, where each block has a hidden dimension of 768. We convert a pre-trained, general-purpose BERT model into a deep matching model by first concatenating the input candidates u_i, u_j from Eq. 4.11 into one sequence, separated by a `[SEP]` token. Then, we append a FeedForward layer after BERT to project its output into a 2-dimensional space. The matching result is a binary label of 0 or 1, where 1 indicates the two inputs are deemed matched under the definition in Sec. 4.1.2. To learn ϕ, we *meta-fine-tune* BERT under the Reptile-learning procedures defined by Eq. 4.11 and 4.12.

4.4 Data Collection from Click Graphs

A click graph is a bipartite graph, where each vertex corresponds to either a search query or a document title. An edge between a query vertex and a title vertex indicate that the user who issued the query clicked on the corresponding title. For search engines, click graph data are abundantly available and easy-to-retrieve. For instance, the QQ mobile browser from Tencent records over 100 million query-document clicks on a daily basis. We believe click graphs naturally reflect the degree relatedness between various entities in the open-domain. After issuing a search query, a user browses the returned document titles and chooses the documents that match his/her intents, and the same intent may also be expressed by other search queries. Therefore, starting from a query vertex, we could potentially reach many neighboring queries or titles by hopping the edges. We assume that a fewer number of hops between two vertices indicate a higher degree of relatedness.

Under this assumption, we collect the training data by following a k-hop Breadth-First-Search (BFS) strategy on a click graph. We define 1 hop as going from a query vertex to a neighboring query vertex through a title vertex, or vice versa. We build upon the definitions in Sec. 3.4 and further define the k-*hop neighbors set* for a vertex v in a click graph $\mathcal{G}$ as $\mathcal{S}_{[.]}^{v,\mathrm{k}}$, where [.] determines the type of vertices to include. For instance, $\mathcal{S}_{[Q]}^{v,\mathrm{k}}$ includes only

neighboring queries while $\mathcal{S}^{v,\mathrm{k}}_{[Q,T]}$ includes both queries and document titles. Regardless of whether v is query or a title, $\mathcal{S}^{v,\mathrm{k}}_{[.]}$ satisfies the following conditions,

- $\mathcal{S}^{v,\mathrm{k}}_{[.]}$ is non-empty.
- There exists a shortest path in $\mathcal{G}$ between any two vertices in $\mathcal{S}^{v,\mathrm{k}}_{[.]}$.
- The number of edges passed by any shortest path is no more than 2k.

Each edge e in $\mathcal{G}$ is also weighted, where the weight reflects the number of times that click occurred. We use the same click graph from Chapter. 3, which is constructed using 8-days of click logs from the QQ mobile browser. We apply the same pre-processing steps, as mentioned in Sec. 3.4, where for every query we only keep the top-3 outgoing edges.

4.4.1 Relevant Words Data

To acquire training data for the RWG model, we follow a similar procedure as described in Sec. 3.4. For a query q from $\mathcal{G}$, we first discover its 3-hop neighboring queries set, i.e. $\mathcal{S}^{q,3}_{[Q]}$. We then define the set of targets relevant words for q as $\mathcal{R}^q_t$. Next, we iterate through every query $\hat{q}$ in $\mathcal{S}^{q,3}_{[Q]}$ and perform Part-Of-Speech (POS) tagging utilizing the StanfordCoreNLP tagger [Manning *et al.*, 2014*a*]. For every word w in $\hat{q}$, We add it to $\mathcal{R}^q_t$ only if it satisfies the following conditions,

1. w is in V_C, i.e. w is a concept word.
2. w has a POS tag of Named Entity (NR), Noun (NN), or Verb (VV).

After checking every $\hat{q}$, we add q and $\mathcal{R}^q_t$ as a new training instance for the RWG model if $\mathcal{R}^q_t$ is non-empty and contains no more than 100 words. Finally, we repeat the above process for every query in the click graph to derive the complete dataset, from which we then split train, dev and test sets.

4.4.2 Meta-learning Data

Meta-learning requires a set of tasks which are all sampled from the same underlying task distribution, as stated in Eq. 4.12. And in Eq. 4.11 we constrain the type of each task to the binary classification of relatedness between two inputs. Under these requirements, we propose the following tasks,

- **Query-to-query (Q2Q)**, where the goal is to classify whether two queries are related. For a query q from $\mathcal{G}$, we create positive matching instances by randomly pairing q with 3 of its neighbors in $\mathcal{S}^{q,2}_{[Q]}$. In the next two tasks, positive instances are created in the same fashion, only with different inputs and 2-hop neighbor sets.
- **Query-to-title (Q2T)**. Whether a query is related to a document title.

- **Title-to-title (T2T)**, where we classify whether two document titles are related.
- **Query/title-to-word (QT2W)**, which is inspired by the relevant word generation task. The first input is a sequence, either a query or a title, and the second input is a word. The goal is to classify whether the word is relevant to the sequence. We create positive instances by pairing the input v to all the derivable relevant words in $\mathcal{S}^{v,3}_{[Q,T]}$, following the same procedure in Sec. 4.4.1. However, the relevant words here are no longer constrained by condition 1 from Sec. 4.4.1.

We argue that search queries Q and document titles T follow different underlying distributions, i.e. $p(Q) \nsim p(T)$, because in general, a query reflects a user's intent for a specific type of resource, while a title is a concise summary of the document content. Therefore, it is beneficial for our model to learn from both Q2Q and T2T tasks.

We sample negative instances by randomly selecting other inputs of the same type, which is similar to one of the original training task of BERT [Devlin *et al.*, 2018]. For example, for a positive pair $q - q^+$ from the Q2Q task, we randomly select a q^- from all unique queries Q. The same procedure is applied to the Q2T and T2T tasks. For the QT2W task, we randomly choose 3 words from all possible relevant words to create 3 negative pairs for every positive pair.

4.5 Experimentation

4.5.1 Datasets

Before we extract any training data from our click graph, we first randomly sample $400,000$ queries from it as the held-out test set. After removing queries that contain invalid characters, we get $398,447$ test queries. To avoid information leakage, we then prune these queries and their corresponding links from the click graph. Finally, we extract training data by following the approaches in Sec. 4.4.

Table. 4.1 and 4.2 provide general statistical information on the datasets used for learning the RWG and BERT matching model, respectively. To ensure that every task contributes evenly to the meta-learning process, we constrain the train/dev set of each task to have the same size by randomly pruning larger datasets.

4.5.2 Baseline Models

We compare our proposed framework against the following baseline models.

MoWE. This baseline follows the conventional Mean-of-Word-Embeddings setup. We utilize the Tencent AI Lab pre-trained Chinese word embedding [Song *et al.*, 2018*b*]. We set a cosine similarity threshold of 0.7 as the decision boundary. In other words, for every input query, we find the most similar concepts and only keep a concept if its cosine similarity with the input query is larger than or equal to 0.7. We adopt the FAISS [Johnson *et al.*, 2017] tool for fast similarity searches.

Table 4.1: Statistical information on datasets for training the RWG model.

	Train	Dev	Test
Size	7.9M	980K	880K
Avg # of words in inputs	6.95	6.94	6.94
Avg # of relevant words	4.94	4.94	4.93
Input vocabulary size	435,642		
Output vocabulary size	18,717		

Table 4.2: Statistical information on datasets for meta-fine-tuning BERT.

	Q2Q		Q2T		T2T		QT2W	
	Train	Dev	Train	Dev	Train	Dev	Train	Dev
Size	4.85M	539K	4.85M	539K	4.85M	539K	4.85M	539K
Avg # of words in inputs	4.65	4.65	7.78	7.78	10.87	10.87	4.06	4.06
Vocabulary size	406K	162K	755K	245K	586K	235K	222K	63K

ELMo-pre-trained. As a contextualized representation, ELMo [Peters *et al.*, 2018] incorporates the context of the sequence when generating word embeddings. We use the pre-trained Chinese ELMo model with a hidden layer size of 1024, which is provided by [Che *et al.*, 2018; Fares *et al.*, 2017]. The procedure to select the matching concepts is the same as the MoWE baseline, where a fixed decision threshold of 0.7 is applied.

BERT-pre-trained. We are interested in how much of the generalizable knowledge learn by the pre-trained BERT language model could directly transfer to the problem of query-concept matching. Therefore, we set up a pre-trained BERT-base model [1] in the same fashion as the ELMo-pre-trained baseline, where the output of the last Transformer layer is taken as the sequence representation.

BoW. In the Bag-of-Words baseline, we directly match concepts from words in the query. The relatedness of each concept is scored in the same way as described in Sec. 4.3.1. The BoW baseline is essentially our proposed first stage without the RWG model.

RW. The Relevant Words (RW) baseline is our proposed first stage. Here, we directly use the shortlisted concepts as the matching results. This baseline serves as an ablation comparison to help verify the effectiveness of the second stage models.

In addition to the above baselines, we report the performance of the following two-stage frameworks.

MoWE-BERT-fine-tune. We additionally fine-tune a pre-trained BERT-base model

[1] We use the pre-trained Chinese BERT model from `https://github.com/huggingface/pytorch-transformers`

as a matching model on only the Q2Q task. We then pair it with a MoWE shortlisting scheme, which is constructed the same way as the MoWE baseline except that we do not set a similarity threshold here.

RW-BERT-fine-tune. In this baseline, we adopt the same fine-tuned BERT model from MoWE-BERT-fine-tune and instead pair it with our proposed RW first stage.

Following the naming conventions above, we refer to our proposed two-stage framework as **RW-BERT-meta**. For all the two-stage setups, we limited the first stage to only pass the top-10 most similar concepts to the second stage. For all the RW setups, we take the top-15 predicted relevant words and set $\gamma = 1.0$. The decision threshold for all second stage classifiers is 0.5.

4.5.3 Training

We implement both stages of our model using Pytorch [Paszke *et al.*, 2017] 0.4 and train them with the Adam [Kingma and Ba, 2014] optimizer.

Training the RWG model

We train the RWG model by minimizing the Binary Cross-Entropy loss on the target relevant words. The input and output vocabulary sizes are set to the values presented in Table. 4.1. We initialize the embedding layers with the Tencent AI Lab [Song *et al.*, 2018*b*] 200*d* pre-trained embedding and allow them to be further trained. We choose the top-100 recall rate, i.e. the percentage of truth words that appear in the top-100 predictions, as the metric for hyper-parameter tuning. We select a hidden size of 2048 for the Bi-GRU and a Dropout [Srivastava *et al.*, 2014] rate of 0.5. For the optimizer, We set an initial learn-rate of 0.001 and follow a simple learn-rate decay strategy: If the train/dev loss of the current epoch is higher than the previous epoch, decay the learn-rate by 0.5, where the minimum learn-rate is set to 0.0001. The RWG model converges in 20 epochs, with a batch size of 256 on an Nvidia RTX-2080Ti GPU. Each epoch takes approximately 90 minutes to finish.

Meta-fine-tuning BERT

We meta-fine-tune the pre-trained BERT-base model under the Reptile algorithm [Nichol *et al.*, 2018], where the objective for each task is to minimize the Cross-Entropy loss on the predicted labels. We choose a fixed task-learn-rate of 0.0001 for the optimizer, and a fixed meta-learn-rate of 0.1. On two RTX-2080Ti GPUs, we carry out the meta-fine-tuning with a batch size of 32, and we terminate the process if the loss on the development set does not change by more than 0.01 between two epochs. In the end, our BERT-meta model converges in 5 epochs with an average classification accuracy of 95.5% on the dev sets, and each epoch takes approximately one day to complete.

Table 4.3: Top-N recall scores on the RWG test set.

	Coverage	Top-10	Top-20	Top-50	Top-100	Top-500
ELMo Embedding	1.0	0.139	0.169	0.216	0.258	0.383
TAL Embedding	0.957	0.411	0.461	0.523	0.567	0.668
RWG	0.884*	0.726	0.783	0.843	0.882	0.951

*The coverage is not 100% because we reject inputs that contain Out-Of-Vocabulary (OOV) words.

4.5.4 Offline Evaluation

We conduct two types of offline evaluations. First, we evaluate the RWG model on its own held-out test set. Next, we propose a new approach for evaluating query-concept matching results on a click graph.

Evaluating the RWG model

As a sanity check, we want to showcase that our RWG model is learning the word relevance features conveyed in a click graph. We also wish to demonstrate that such features cannot be accurately captured by other word representations. Therefore, we report the top-N recall scores on the test set of the RWG model, where the input to the model is a query, and the target output is a set of relevant words. We define top-N recall score as the percentage of truth relevant words that appear in the top-N predicted words.

Next, we compare our approach to the Tencent AI Lab (TAL) pre-trained embeddings and the pre-trained ELMo representations. For these baselines, the query representation is the mean of its word embeddings, and we find the top-N most similar words to it. We also report the coverage to ensure that the recall scores are reliable. Table. 4.3 reports the results of this experiment.

Offline evaluation using click graphs

As we mentioned before, it is difficult to manually label an unbiased query-concept matching dataset due to the overwhelming number of available concepts. To mitigate this problem, we propose an automatic procedure to evaluate the results of query-concept matching. We first note that many concepts are also popular queries. Therefore, some concepts could exist in our click graph as query vertices. In fact, on our click graph, we found that it contains 2997 concepts as queries. Next, under our previous assumption, the hop distance between two vertices is a rough estimate for their relatedness. Combining these intuitions, we argue that if a concept is a good match to a query, and they are in the same click graph, then they should be within a certain number of hops from each other. Although the exact hop distance may not be a reliable measure for the relatedness, we could still get a rough

Table 4.4: Offline evaluation results using our complete click graph.

	# supports	Avg # hops	MAP	MAR
MoWE	1661	4.77	0.639	0.769
ELMo-pre-trained	1446	4.90	0.627	0.624
BERT-pre-trained	1029	5.56	0.528	1.000
BoW	2641	4.99	0.364	0.819
RW	2420	5.68	0.368	0.748
MoWE-BERT-fine-tune	1661	4.77	0.217	0.950
RW-BERT-fine-tune	1690	5.16	0.674	0.960
RW-BERT-meta	1892	5.61	**0.801**	0.916

estimate for the Mean-Average Precision (MAP) and Mean-Average-Recall (MAR) metrics.

To summarize our strategy, we assume the truth label between a concept and a query is 1 if both are in our click graph, and they could reach each other within 10 hops. Note that the click graph used for evaluation here is the original graph, i.e. it contains all the test queries. With truth labels, we could compute the macro-averaged MAP and MAR scores for the query-concept matching results, which are defined as,

$$\text{MAP} = \frac{1}{\mathcal{N}} \sum_{i=1}^{\mathcal{N}} \frac{TP_i}{TP_i + FP_i}, \tag{4.13}$$

$$\text{MAR} = \frac{1}{\mathcal{N}} \sum_{i=1}^{\mathcal{N}} \frac{TP_i}{TP_i + FN_i}, \tag{4.14}$$

where $\mathcal{N}$ is the number of results with positive predictions. TP, FP, FN denotes true-positives, false-positives and false negatives.

Table. 4.4 reports these metrics, the average number of hops between a query and a concept, as well as the number of queries that contributed to the computations.

4.5.5 Human Evaluation

Without a labelled test set, offline evaluation alone is not sufficient in revealing the real performance. We argue that human judgment is a more reliable evaluation metric. Therefore, we randomly select 206 test instances where every model outputs a different matching value. We then take the top-3 matched concepts of every model according to either the cosine similarities (MoWE, ELMo-pre-trained, BERT-pre-trained), the predicted probabilities (two-stage frameworks), or the overlap-scores (RW and BoW) [2]. Finally, we hire 2 human judges and assign each judge to asses half of the test instances according to the following criteria,

[2]We randomly take 3 concepts if there are more than 3 top concepts with the same score

Table 4.5: Human evaluation results on 206 randomly selected test instances.

	# of concepts matched	# of 0s assigned	# of 1s assigned	# of 2s assigned	Total score
MoWE	562	307	192	63	318
ELMo-pre-trained	538	401	104	33	170
BERT-pre-trained	623	524	85	14	113
BoW	589	295	216	78	372
RW	617	255	250	112	474
MoWE-BERT-fine-tune	618	471	125	22	169
RW-BERT-fine-tune	488	176	200	112	424
RW-BERT-meta	517	**137**	238	**142**	**522**

- For a test instance, the judge reviews the top-3 concepts of each model, then assign a score for each concept.
- A concept receives a score of 2 if there exists a *isA* relation between it and the input query. If a concept only match part of the query, for example, if the query "Resident Evil background story" is matched to the concept "zombie movies", then the judge assigns a score of 1 for this concept. Otherwise, for every non-related concept, a score of 0 is assigned.
- On a test instance, the maximum attainable score for a model is 6, and the minimum score is 0.

We report the results of human evaluation in Table. 4.5. In Sec. 4.6, We discuss the implications behind these results and conduct case studies to get a more intuitive visualization on the performance of the competing models.

4.6 Performance Analysis

4.6.1 Offline Evaluation Results

We begin with the RWG test results in Table. 4.3. The RWG model easily outperforms the other two pre-trained word representations, which proves our original claim that the word-relatedness derived from a click graph is different from the word similarities learned by pre-trained embeddings.

For the matching results, we first observe in Table. 4.4 that most models achieve high MAR scores, which means that most models are unlikely to miss-classify matching concepts as non-matching. However, we see a wide range of MAP scores, where a lower MAP score indicates that a model tends to classify more non-matching concepts as matching.

Specifically, the MoWE-BERT-fine-tune baseline scores lowest on the metric. We suspect that this is because we do not set a similarity threshold for the first stage, which means the second stage alone need to filter out all the non-matching concepts and rejects difficult inputs. Clearly, fine-tuning BERT on the Q2Q task does not learn enough transferable knowledge for this goal. As a comparison, by replacing the first stage with our proposed module, the MAP score improves significantly in the RW-BERT-fine-tune baseline.

Among the single-stage baselines, we observe that MoWE with a similarity threshold of 0.7 performs the best, while pre-trained, deep contextualized representations like ELMo or BERT have difficulties transferring their knowledge to our problem setting. This implies that the problem of query-concept matching requires unique contextual representations for the input sequences. Additionally, we observe that the RW baseline performs worse than the BoW baseline. We suspect that, while the imaginative nature of the RWG model enables us to discover of additional relevant words, it could "wander-off" too far and match completely irrelevant concepts. Therefore, we employ the meta-fine-tuned BERT model as another stage of quality assurance to effectively prune all the non-matches. In the end, our proposed framework outperforms all other baselines by a large margin on the MAP metric.

4.6.2 Human Evaluation Results

According to Table. 4.5, our framework obtains the highest total score and also receives more 2s than the baselines, which means that it discovers *isA* relationships more accurately. Specifically, compared to the RW baseline, our framework finds less matching concepts overall (517 vs. 617) but with a higher total score. Therefore, the proposed matching model serves its purpose well as a quality assurance stage for the RW shortlisting scheme. Compared to RW-BERT-fine-tune baseline, we also prove the effectiveness of the Reptile meta-learning algorithm on learning a more generalizable text matching model.

4.6.3 Case Studies

In Table. 4.6, we show representative examples of the top-10 relevant words generated by our RWG model and compared with the top-10 most similar words found by the Tencent AI Lab (TAL) word embeddings. We show the original query and relevant words in Chinese and their English translations.

In general, our RWG model produces words that are related to the input on a higher conceptual-level, especially for the highlighted words. In comparison, the cosine similarity search based on word embeddings discovers words that are related only to certain words in the input. For instance, in the first row of Table. 4.6, our RWG model knows that "RongWei rx5" is a car, and the query is asking for tutorials on how to project displays between a cell phone and a car. Therefore, it discovers conceptually relevant words like "car", "inter-connect" and "tutorial", while in the baseline approach, the meaning of the query is not captured, and the majority of related words are only similar to "screen".

Another representative case is the input query "Girl group fox", two seemingly unrelated entities that convey a unique meaning when put together. In this case, "fox" actually refers to a champion named Ahri from the game League-of-Legends, who is a nine-tailed fox. And "Girl group" refers to a popular skin for Ahri at that time. Without this background knowledge, the TAL baseline only finds similar words to "Girl group", whereas the RWG model could successfully uncover keywords related to the game. Overall, Table. 4.6 provides valuable evidence on the usefulness of the RWG model.

Table 4.6: Comparison of the top-10 words found by the RWG model and the TAL pre-trained word embeddings.

Input query	Model	Top-10 words
荣威rx5映射手机屏幕 RongWei rx5 project-cellphone screen	RWG	荣威, 手机, 映射, 汽车, 屏幕, 论坛, 问题, 互联, 教程, 大屏 RongWei, cellphone, project, car, screen, forum, question, inter-connect, tutorial, big-screen
	TAL	墨水屏, 显示屏, 屏幕, 运存, 曲面屏, koobee, amoled, 全面屏, 折叠屏, 投屏 ink screen, display screen, screen, RAM, curved screen, koobee, amoled, bezel-less display, folding display, screen projection
酸汤鲈鱼 Sour soup bass	RWG	做法, 食物, 钓鱼, 技巧, 鲈鱼, 食谱, 功效, 百科, 汤, 美食 method, food, fishing, technique, bass, recipe, efficacy, Wiki, soup, delicacy
	TAL	高汤, 剁椒, 鸭肉, 马鲛鱼, 鱼片, 花鲢, 姜丝, 鲳鱼, 黄骨鱼, 干贝 soup-stock, diced pepper, duck meat, mackerel, fish fillet, spotted silver carp, sliced ginger, butter fish, Amur catfish, dried scallop
女团狐狸 Gril group fox	RWG	女团, 皮肤, 看见, 联盟, 图片, 英雄, 狐狸, 游戏, 哥们, 阿狸 girl group, skin, see, league, picture, champion, fox, game, buddy, Ahri
	TAL	秀智, apink, 裴秀智, 金泰妍, exo, twice, 女团, snh48, 男团, 朴智妍 Suzy, spink, Suzy Bae, Taeyeon, exo, twice, girl group, snh 48, boy group, Park Ji-yeon
性格测试免费版 Personality test-free version	RWG	测评, 软件, 分析, 在线, 下载, 测试, 百科, 性格, 心理, 成长 evaluation, software, analysis, online, download, test, wiki, personality, psychology, growth
	TAL	体质, 竞技, 人格, 眉形, 天赋, 性格, 烤机, 血型, 测试, 情商 physique, competition, character, brow-shape, talent, personality, stress-test, blood type, test, emotional quotient

We now examine how well the RWG model and the BERT matching model cooperate together. In Table. 4.7, we show the top-3 matched concepts for every test query. Due to space concerns, we compare our results to only a few competitive baselines models.

Overall, our model is superior at both finding related candidates and picking out the best matching concepts. Consider the first instance of Table. 4.7, here, Omen and Legion are gaming laptop models from HP and Lenovo, respectively. Our proposed RW stage accurately discovers this relation. Then, the second stage reliably filters out non-matching concepts such as "Laptop keyboard". In contrast, the RW-BERT-fine-tune baseline mistakenly prunes the best matching concept, "Gaming laptop", from the shortlisted candidates.

Table 4.7: Comparison of the top-3 concepts matched by our proposed framework and competitive baseline models.

Input query	Model	Matched concepts
暗夜精灵和拯救者 Omen and Legion	RW-BERT-meta	游戏笔记本电脑, 联想笔记本电脑 Gaming laptop, Lenovo laptop
	RW	游戏笔记本电脑, 笔记本电脑键盘, 联想笔记本电脑 Gaming laptop, Laptop keyboard, Lenovo laptop
	MoWE	拯救者小说 Legion novel
	RW-BERT-fine-tune	联想笔记本电脑 Lenovo laptop
壁纸尺寸修改 Wallpaper size-modification	RW-BERT-meta	修改图片软件 Picture modification software
	RW	修改图片软件. 手机修改软件 Picture modification software, Cellphone modification software
	MoWE	新款电视墙 new television wall
	Bow	尺寸对照表, 修改网游, 修改软件 Size reference chart, Modify online game, Modification software
创新创业策划书模板 Innovation-entrepreneurship-proposal template	RW-BERT-meta	大学生创业计划书, 大学生创业 University student entrepreneurship proposal, University student entrepreneurship
	RW	创业青年. 大学生创业, 大学生创业计划书 Young entrepreneur, University student entrepreneurship University student entrepreneurship proposal
	MoWE	大学生创业计划书, 个人简历ppt模板, ppt模板 University student entrepreneurship proposal, Resume ppt template, Ppt template
	RW-BERT-fine-tune	大学生创业, 大学活动策划, 大学生创业计划书 University student entrepreneurship, University activity planning, University student entrepreneurship proposal

Chapter 5

Summary and Conclusions

5.1 Contributions

In this thesis, we propose three novel approaches under the topic of search query understanding. The focus of search query understanding is to develop a better representation for the intents behind a query and efficiently utilize them to enhance down-stream services.

We start by creatively taking a backward strategy where the most probable search query is inferred from a document. Under this setting, we develop a better understanding of search queries by learning to generate them in a word-by-word fashion. We combine several deep learning architectures, such as Hierarchical Recurrent Encoders, Graph Convolution Networks (GCNs) and Transformers into the G-S2A generative model. The idea is to explore graphical document representations, then utilize them to develop a deeper understanding on the context of the document, which in turn would improve the process of query term selection & inference. With the addition of a sentence-level GCN and a keyword-level GCN to learn from the corresponding document graphs, our G-S2A outperforms several competitive baselines on BLEU-$1, 2, 3, 4$, ROUGE-$1, 2, L$ and Exact Match (EM) metrics.

Next, we introduce a two-stage learning framework for related queries generation. We first retrieve massive amounts of training data from a click graph. Then, our framework breaks down related query generation from input queries as well as document titles into two steps, namely, relevant words discovery, which alleviates the Lexical Chase problem, and context-aware query generation, which improves the process of query understanding and recommendation. We carefully design a Relevant Words Generator (RWG) model and a Dual-Vocabulary sequence-to-sequence (DV-Seq2Seq) model for each sub-problem. Together, our framework balances performance, time-complexity and interpretability. 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, also in terms of BLEU-$1, 2, 3, 4$, ROUGE-$1, 2, L$ and Exact Match (EM) metrics. Together with the G-S2A model, we verify the feasibility and practicality of deep generative models for the problem of search query understanding.

In Chapter. 4, we take a text-matching approach. We acquire a set of more than 150K

concepts from our industry partner, which serves as an abstraction for the massive amount of knowledge in the open-domain. Under this abstraction, we transform the problem of search query understanding to the problem of query-concept matching. To this end, we first re-adopt the RWG model to construct a new shortlisting scheme. With no labelled data, We meta-fine-tune a pre-trained BERT model into a competitive matching model using the Reptile meta-learning algorithm. By conducting both offline and human evaluations, we prove the superiority of our proposed two-stage matching framework against conventional and deep learning baselines. To the best of our knowledge, we are the first to apply the Reptile meta-learning algorithm to fine-tune a deep text matching model.

In summary, we approach search query understanding from generative and text-matching perspectives. We test deep learning models and propose novel modifications or frameworks to further improve their performance. We prove that deep learning indeed opens up many exciting new possibilities for this field.

5.2 Directions for Future Work

We propose the following directions for future research,

- A major challenge for deep generative language models is how to achieve better quality control on the outputs. The same issue also exists for query generation, i.e. how to ensure the generated queries are fluent and grammatically correct.
- Although the Relevant Words Generator model proves to be extremely useful, it is actually biased by word frequencies and the click position bias. To create an unbiased RWG, one could adopt related ideas from works on generalized word representations and learning-to-rank tasks.
- To simplify computation, we do not incorporate the weights on the edges of the click graph when gathering data or training our models. However, these weights may provide additional insights on relatedness. For instance, one could further classify the degree of relatedness beyond a binary label based on the cumulative edges weights between two vertices.
- It would be interesting to see how a structure similar to the Matching Network [Vinyals *et al.*, 2016], with a non-parametric matching metric, would perform in comparison to our meta-fine-tuned BERT matching model.
- The success of pre-trained deep language models like BERT makes us wonder whether it is possible to train a generalizable deep representation for queries. Since click graph data is easy to retrieve and abundantly available, one might be able to train deep query language models based on the hop distances and edge weights.

Bibliography

Andrychowicz, Marcin, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford and Nando De Freitas (2016). Learning to learn by gradient descent by gradient descent. In: *Advances in neural information processing systems*. pp. 3981–3989.

Antonellis, Ioannis, 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), 408–421.

Arora, Sanjeev, 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. pp. 1–10.

Auer, Sören, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak and Zachary Ives (2007). Dbpedia: A nucleus for a web of open data. In: *The semantic web*. pp. 722–735. Springer.

Bahdanau, Dzmitry, Kyunghyun Cho and Yoshua Bengio (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Banarescu, Laura, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer and Nathan Schneider (2013). Abstract meaning representation for sembanking. In: *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*. pp. 178–186.

Beeferman, Doug and Adam Berger (2000). Agglomerative clustering of a search engine query log. In: *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. pp. 407–416.

Bengio, Samy, Yoshua Bengio, Jocelyn Cloutier and Jan Gecsei (1992). On the optimization of a synaptic learning rule. In: *Preprints Conf. Optimality in Artificial and Biological Neural Networks*. Univ. of Texas. pp. 6–8.

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

Bojanowski, Piotr, Edouard Grave, Armand Joulin and Tomas Mikolov (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics* **5**, 135–146.

Boteva, Vera, 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. pp. 716–722.

Che, Wanxiang, Yijia Liu, Yuxuan Wang, Bo Zheng and Ting Liu (2018). Towards better UD parsing: Deep contextualized word embeddings, ensemble, and treebank concatenation. In: *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Association for Computational Linguistics. Brussels, Belgium. pp. 55–64.

Chen, Haolan, Fred X Han, Di Niu, Dong Liu, Kunfeng Lai, Chenglin Wu and Yu Xu (2018). Mix: Multi-channel information crossing for text matching. In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM. pp. 110–119.

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

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

Cui, Hang, 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 World Wide Web*. ACM. pp. 325–332.

Daille, Adrien Bougouin Florian Boudin Béatrice (2013). Graph-based topic ranking for keyphrase extraction.

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

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

Fares, Murhaf, Andrey Kutuzov, Stephan Oepen and Erik Velldal (2017). Word vectors, reuse, and replicability: Towards a community repository of large-text resources. In: *Proceedings of the 21st Nordic Conference on Computational Linguistics*. Association for Computational Linguistics. Gothenburg, Sweden. pp. 271–276.

Faruqui, Manaal, Jesse Dodge, Sujay K Jauhar, Chris Dyer, Eduard Hovy and Noah A Smith (2014). Retrofitting word vectors to semantic lexicons. *arXiv preprint arXiv:1411.4166*.

Finn, Chelsea, Pieter Abbeel and Sergey Levine (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org. pp. 1126–1135.

Fonseca, Bruno M, 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. pp. 696–703.

Frank, Eibe, 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. pp. 668–673.

Gao, Jianfeng 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. p. 1.

Gao, Jianfeng, 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*. pp. 666–676.

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

Gers, Felix A, Jürgen Schmidhuber and Fred Cummins (1999). Learning to forget: Continual prediction with lstm.

Gori, Marco, Gabriele Monfardini and Franco Scarselli (2005). A new model for learning in graph domains. In: *Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on*. Vol. 2. IEEE. pp. 729–734.

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

Guo, Jiafeng, Yixing Fan, Qingyao Ai and W Bruce Croft (2016). A deep relevance matching model for ad-hoc retrieval. In: *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM. pp. 55–64.

Han, Fred X., 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*.

He, Yunlong, 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. pp. 1443–1452.

Hermann, Karl Moritz, 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*. pp. 1693–1701.

Hersh, William, Chris Buckley, TJ Leone and David Hickam (1994). Ohsumed: an interactive retrieval evaluation and new large test collection for research. In: *SIGIR'94*. Springer. pp. 192–201.

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

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

Huang, Po-Sen, 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 Information & Knowledge Management*. ACM. pp. 2333–2338.

Järvelin, Kalervo and Jaana Kekäläinen (2002). Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)* **20**(4), 422–446.

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

Jeh, Glen 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. pp. 538–543.

Johnson, Jeff, Matthijs Douze and Hervé Jégou (2017). Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*.

Jones, Rosie and Daniel C Fain (2003). Query word deletion prediction. In: *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*. ACM. pp. 435–436.

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

Kiela, Douwe, Felix Hill and Stephen Clark (2015). Specializing word embeddings for similarity or relatedness. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pp. 2044–2048.

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

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

Koch, Gregory, Richard Zemel and Ruslan Salakhutdinov (2015). Siamese neural networks for one-shot image recognition. In: *ICML deep learning workshop*. Vol. 2.

Landauer, Thomas K, Peter W Foltz and Darrell Laham (1998). An introduction to latent semantic analysis. *Discourse processes* **25**(2-3), 259–284.

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

LeCun, Yann, Yoshua Bengio and Geoffrey Hinton (2015). Deep learning. *nature* **521**(7553), 436.

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

Li, Ke and Jitendra Malik (2016). Learning to optimize. *arXiv preprint arXiv:1606.01885*.

Li, Yujia, Daniel Tarlow, Marc Brockschmidt and Richard Zemel (2015). Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*.

Li, Zhenguo, Fengwei Zhou, Fei Chen and Hang Li (2017). Meta-sgd: Learning to learn quickly for few-shot learning. *arXiv preprint arXiv:1707.09835*.

Lin, Chin-Yew (2004). Rouge: A package for automatic evaluation of summaries. In: *Proc. ACL workshop on Text Summarization Branches Out*.

Litvak, Marina, 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*. pp. 121–130. Springer.

Liu, Bang, Mingjun Zhao, Di Niu, Kunfeng Lai, Yancheng He, Haojie Wei and Yu Xu (2019*a*). Learning to generate questions by learning what not to generate. In: *Proceedings of The Web Conference 2019*.

Liu, Bang, Ting Zhang, Di Niu, Jinghong Lin, Kunfeng Lai and Yu Xu (2018*a*). Matching long text documents via graph convolutional networks. *arXiv preprint arXiv:1802.07459*.

Liu, Bang, Ting Zhang, Fred X Han, Di Niu, Kunfeng Lai and Yu Xu (2018*b*). Matching natural language sentences with hierarchical sentence factorization. In: *Proceedings of the 2018 World Wide Web Conference*. International World Wide Web Conferences Steering Committee. pp. 1237–1246.

Liu, Bang, Weidong Guo, Di Niu, Chaoyue Wang, Shunnan Xu, Jinghong Lin, Kunfeng Lai and Yu Xu (2019*b*). A user-centered concept mining system for query and document understanding at tencent. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. KDD '19. ACM. New York, NY, USA. pp. 1831–1841.

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

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

Liu, Xiaodong, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh and Ye-Yi Wang (2015). Representation learning using multi-task deep neural networks for semantic classification and information retrieval.

Liu, Zhiyuan, 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. pp. 257–266.

Lu, Zhengdong and Hang Li (2013). A deep architecture for matching short texts. In: *Advances in Neural Information Processing Systems*. pp. 1367–1375.

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

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

Manning, Christopher D., Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard and David McClosky (2014*a*). The Stanford CoreNLP natural language processing toolkit. In: *Association for Computational Linguistics (ACL) System Demonstrations*. pp. 55–60.

Manning, Christopher, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard and David McClosky (2014*b*). The stanford corenlp natural language processing toolkit. In: *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*. pp. 55–60.

McCann, Bryan, James Bradbury, Caiming Xiong and Richard Socher (2017). Learned in translation: Contextualized word vectors. In: *Advances in Neural Information Processing Systems*. pp. 6294–6305.

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

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

Mikolov, Tomas, Kai Chen, Greg Corrado and Jeffrey Dean (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mueller, Jonas and Aditya Thyagarajan (2016). Siamese recurrent architectures for learning sentence similarity. In: *Thirtieth AAAI Conference on Artificial Intelligence*.

Naik, Devang K and RJ Mammone (1992). Meta-neural networks that learn by learning. In: *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*. Vol. 1. IEEE. pp. 437–442.

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

Nallapati, Ramesh, Feifei Zhai and Bowen Zhou (2017). Summarunner: A recurrent neural network based sequence model for extractive summarization of documents.. In: *AAAI*. pp. 3075–3081.

Nguyen, Kim Anh, Sabine Schulte im Walde and Ngoc Thang Vu (2016). Integrating distributional lexical contrast into word embeddings for antonym-synonym distinction. *arXiv preprint arXiv:1605.07766*.

Nichol, Alex, Joshua Achiam and John Schulman (2018). On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*.

Palangi, Hamid, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song and Rabab Ward (2016). Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)* **24**(4), 694–707.

Pang, Liang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan and Xueqi Cheng (2016). Text matching as image recognition.. In: *AAAI*. pp. 2793–2799.

Papineni, Kishore, 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. pp. 311–318.

Paszke, Adam, 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*.

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

Pennington, Jeffrey, Richard Socher and Christopher Manning (2014). Glove: Global vectors for word representation. In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. pp. 1532–1543.

Peters, Matthew E, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee and Luke Zettlemoyer (2018). Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Qiu, Xipeng and Xuanjing Huang (2015). Convolutional neural tensor network architecture for community-based question answering. In: *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Radford, Alec, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei and Ilya Sutskever (2019). Language models are unsupervised multitask learners. *OpenAI Blog*.

Ravi, Sachin and Hugo Larochelle (2016). Optimization as a model for few-shot learning.

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

Riezler, Stefan, 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. pp. 737–744.

Robertson, Stephen, Hugo Zaragoza et al. (2009). The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval* **3**(4), 333–389.

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

Rosenblatt, Frank (1961). Principles of neurodynamics. perceptrons and the theory of brain mechanisms. Technical report. CORNELL AERONAUTICAL LAB INC BUFFALO NY.

Scarselli, Franco, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner and Gabriele Monfardini (2009). The graph neural network model. *IEEE Transactions on Neural Networks* **20**(1), 61–80.

Schmidhuber, Jürgen (1992). Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation* 4(1), 131–139.

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

Serban, Iulian Vlad, 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. pp. 3776–3784.

Severyn, Aliaksei and Alessandro Moschitti (2015). Learning to rank short text pairs with convolutional deep neural networks. In: *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*. ACM. pp. 373–382.

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

Shen, Yelong, 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. pp. 373–374.

Snell, Jake, Kevin Swersky and Richard Zemel (2017). Prototypical networks for few-shot learning. In: *Advances in Neural Information Processing Systems*. pp. 4077–4087.

Song, Linfeng, Yue Zhang, Zhiguo Wang and Daniel Gildea (2018*a*). A graph-to-sequence model for amr-to-text generation. *arXiv preprint arXiv:1805.02473*.

Song, Yan, Shuming Shi, Jing Li and Haisong Zhang (2018*b*). 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. pp. 175–180.

Song, Yan, Shuming Shi, Jing Li and Haisong Zhang (2018*c*). Directional skip-gram: Explicitly distinguishing left and right context for word embeddings. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. pp. 175–180.

Srivastava, Nitish, 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**, 1929–1958.

Sun, Yu, Shuohuan Wang, Yukun Li, Shikun Feng, Hao Tian, Hua Wu and Haifeng Wang (2019). Ernie 2.0: A continual pre-training framework for language understanding. *arXiv preprint arXiv:1907.12412*.

Sung, Flood, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr and Timothy M Hospedales (2018). Learning to compare: Relation network for few-shot learning. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pp. 1199–1208.

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

Tan, Jiwei, 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. pp. 1171–1181.

Terra, Egidio and Charles LA Clarke (2004). Scoring missing terms in information retrieval tasks. In: *Proceedings of the thirteenth ACM international conference on Information and knowledge management*. ACM. pp. 50–58.

Thrun, Sebastian and Lorien Pratt (2012). *Learning to learn*. Springer Science & Business Media.

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

Uzun, Yasin (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*.

Vaswani, Ashish, 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*. pp. 5998–6008.

Venkatakrishnan, Shaileshh Bojja, Mohammad Alizadeh and Pramod Viswanath (2018). Graph2seq: Scalable learning dynamics for graphs. *arXiv preprint arXiv:1802.04948*.

Vinyals, Oriol, Charles Blundell, Timothy Lillicrap, Daan Wierstra et al. (2016). Matching networks for one shot learning. In: *Advances in neural information processing systems*. pp. 3630–3638.

Wang, Jingang, 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*.

Wen, Ji-Rong, Jian-Yun Nie and Hong-Jiang Zhang (2002). Query clustering using user logs. *ACM Transactions on Information Systems* **20**(1), 59–81.

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

Xu, Jinxi and W Bruce Croft (2017). Quary expansion using local and global document analysis. In: *Acm sigir forum*. Vol. 51. ACM. pp. 168–175.

Xu, Kun, Lingfei Wu, Zhiguo Wang and Vadim Sheinin (2018). Graph2seq: Graph to sequence learning with attention-based neural networks. *arXiv preprint arXiv:1804.00823*.

Yang, Zhilin, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov and Quoc V Le (2019). Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

Yin, Wenpeng and Hinrich Schütze (2015). Multigrancnn: An architecture for general matching of text chunks on multiple levels of granularity. In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. pp. 63–73.

Yin, Zi, 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. pp. 2131–2139.

Yu, Mo and Mark Dredze (2014). Improving lexical embeddings with semantic knowledge. In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. pp. 545–550.

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

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

Zhang, Ting, 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. pp. 933–942.

Zhang, Wei Vivian, Xiaofei He, Benjamin Rey and Rosie Jones (2007). Query rewriting using active learning for sponsored search. In: *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM. pp. 853–854.

Zheng, Yin, 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), 1056–1069.

Zhou, Qingyu, 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*.