Meta-Learning for Query Conceptualization at Web Scale

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

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

Haolan Chen
Tencent
ShenZhen, China
haolanchen@tencent.com

Weidong Guo
Tencent
BeiJing, China
weidongguo@tencent.com

Shengli Yan
Tencent
ShenZhen, China
victoryyan@tencent.com

Bowei Long
Tencent
ShenZhen, China
kamalong@tencent.com

ABSTRACT

Concepts naturally constitute an abstraction for fine-grained entities and knowledge in the open domain. They enable search engines and recommendation systems to enhance user experience by discovering high-level abstraction of a search query and the user intent behind it. In this paper, we study the problem of query conceptualization, which is to find the most appropriate matching concepts for any given search query from a large pool of pre-defined concepts. We propose a coarse-to-fine approach to first reduce the search space for each query through a shortlisting scheme and then identify the matching concepts using pre-trained language models, which are meta-tuned to our query-concept matching task. Our shortlisting scheme involves using a GRU-based Relevant Words Generator (RWG) to first expand and complete the context of the given query and then shortlisting the candidate concepts through a scoring mechanism based on word overlaps. To accurately identify the most appropriate matching concepts for a query, even when the concepts may have zero verbatim overlaps with the query, we meta-fine-tune a BERT pairwise text-matching model under the Reptile meta-learning algorithm, which achieves zero-shot transfer learning on the conceptualization problem. Our two-stage framework can be trained with data completely derived from a search click graph, without requiring any human labelling efforts. For evaluation, we have constructed a large click graph based on more than 7 million instances of the click history recorded in Tencent QQ browser and performed the query conceptualization task based on a large ontology with 159, 148 unique concepts. Results from a range of evaluation methods, including an offline evaluation procedure on the click graph, human evaluation, online A/B testing and case studies, have demonstrated the superiority of our approach over a number of competitive pre-trained language models and fine-tuned neural network baselines.

KDD '20, August 23–27, 2020, Virtual Event, CA, USA

ACM ISBN 978-1-4503-7998-4/20/08…$15.00
https://doi.org/10.1145/3394486.3403357

CCS CONCEPTS

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

KEYWORDS

Information retrieval; query analysis; conceptualization; meta-learning

ACM Reference Format:
Fred X. Han, Di Niu, Haolan Chen, Weidong Guo, Shengli Yan, and Bowei Long. 2020. Meta-Learning for Query Conceptualization at Web Scale. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '20), August 23–27, 2020, Virtual Event, CA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3394486.3403357

1 INTRODUCTION

Teaching machines to understand search queries and interpret the user intents behind them is essential to building a more intelligent web. To fully "understand" a search query means that a system is not only capable of acquiring relevant knowledge about the terms in the query but is able to extrapolate beyond these terms to discover high-level user intents, which often reveal additional hidden interests of the user. Query understanding is thus of profound significance to many downstream applications such as content recommendation, intent classification, and user-profiling.

In this paper, we study the problem of *search query conceptualization*, which is to find one or multiple matching concepts for a given query from a large pool of pre-defined concepts. A concept is a short text sequence that could be a phrase or a sentence. It associates various text entities under the *isA* relation. For example, *Toyota RAV4 isA* real world entity under the concept *fuel-efficient SUV*. The major benefit of query conceptualization is that concepts create a tractable abstraction for the fine-grained knowledge in queries from the open domain.

We showcase how conceptualization may help query understanding through an example in Fig. 1, where the task is to recommend queries after a user finishes reviewing a page of search results. We present the top-3 related queries for the input query *Toyota RAV4* recommended by Google, Yahoo and Bing. Most of the recommended queries are more detailed versions of the original query. While these can certainly be helpful to the user, in this paper, we are interested in a different recommendation task, which is to discover queries that have fewer or even no common words with the input query but may characterize the query on a higher conceptual level (e.g., fuel-efficient SUVs). The ability of conceptualization, although

native to human intelligence, is what most search engines currently lack and can benefit from.

We define that a concept is a match to a search query only if there exists an *isA* relation between them, i.e., the knowledge conveyed in the query *isA* instance of the concept. Prior research on structured knowledge base and taxonomy construction, such as DBPedia [3], YAGO [35], Probase [43], ConcepT [22], have provided abundant sources to create the pool of concepts.

We propose an efficient solution to the query conceptualization problem based on deep neural networks and meta-learning, which can effectively learn generalizable knowledge from a large amount of *unlabelled* click logs (i.e., the click graph) and transfer it to the query-concept matching problem. Our approach follows a coarse-to-fine strategy in a two-stage framework. Specifically, we introduce a reliable shortlisting scheme to reduce the search space for concepts, where a GRU-based Relevant Words Generator (RWG) takes in a search query and produces the most relevant concept words, which helps to reveal the its surrounding context.

To narrow the shortlist down to the most appropriate matching concepts, in the second stage, we meta-train a pairwise neural text-matching model, where each candidate concept is matched against the input query, and the result indicates if there is indeed an *isA* relation among the pair. The main challenge here is that it is impossible to manually label an unbiased training set due to the large number of candidate concepts at the web scale. Realizing that many concepts are present in and mined from the click graph (i.e., most concepts are a part of a searched query or document title), we believe the problem of query-concept matching shares the same underlying distribution with other similar natural language matching tasks sampled from the click graph, which naturally links to the idea of meta-learning [9, 25]. Therefore, we derive four text-matching tasks by mining the click graph and utilize Reptile [25], which is a type of Model-Agnostic Meta-Learning (MAML) algorithm, to meta-tune a BERT [7] model on these tasks.

To train and evaluate our framework, we have collected over 7 million click logs from Tencent QQ Browser mobile app. Each log contains a search query and the document title a user clicked after he/she issued that query. We then construct a large click graph and derive the training data needed by each stage. For the concept pool, we adopt a version of the ontology presented in [22], which contains 159, 148 user-centered concepts mined from the web click logs.

We have performed extensive comparisons with several baselines based on pre-trained word embeddings, contextualized word representations, and conventionally fine-tuned BERT models. As a labelled test set is costly to acquire, we have designed an evaluation procedure using the click graph, which enables us to report averaged F1 scores for the matching results and indicates the closeness of a query and its matching concept(s) on the click graph. We also hired human judges to examine and score the results returned by each method under a blind policy. Finally, we performed online A/B tests on QQ Browser to verify the effectiveness of the proposed approach in a production environment for a query recommendation task.

The remainder of the paper is organized as follows. We present the detailed methodology in Sec. 2 and describe how to mine the click graph to collect necessary training data in Sec. 3. Sec. 4 presents the experiment setup and results, while we discuss their implications and conduct further analysis in Sec. 5. Finally, we review related work in Sec. 6 and conclude the work in Sec. 7.

2 FRAMEWORK

We begin by formalizing the learning objective. 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, assume that there exists a non-empty set $C_t^q \subset C$, where each concept in C_t^q is a match to q. Our goal is to learn a model f to predict C_t^q for every query with the following objective,

$$\max_f \sum_{q \in Q} \sum_{c_i \in C_t^q} \log p(c_i | q, C;\ f), \tag{1}$$

where Q represents the set of search queries available for training. The dependence on C indicates that the cost of learning is proportional to m. Since we lack a large amount of labelled training query-concept pairs, supervised classification is not feasible when m is large. A shortlisting mechanism is thus needed to reduce the concept search space for each query.

Let us define a shortlisting model g, which takes in q and the complete concept set C, and then outputs a new set $C_s^q \subset C$. The purpose of g is to constrain the search space for each query from C to C_s^q. The original objective from Eq. 1 then becomes more tractable since $|C_s^q| \ll |C|$, and it is expressed as

$$\max_f \sum_{q \in Q} \sum_{c_i \in C_t^q} \log p(c_i | q, C_s^q;\ f). \tag{2}$$

To transform Eq. 2 into a text-matching problem, we define the matching degree between a concept and a query as r, where r is a binary label of either 0 or 1, and the objective becomes

$$\max_f \sum_{q \in Q} \sum_{c_i \in C_{+,-}^q} \log p(r_{q,c_i} | q, c_i;\ f), \tag{3}$$

where $C_{+,-}^q$ represents the set of positive (matching) and negative (non-matching) concepts for q. Fig. 2 depicts the overall training and testing procedure. We provide a more detailed description of each stage in the following subsections.

2.1 Shortlisting by Relevant Words

Simple ideas for shortlisting include comparing the mean word embedding of a query with that of each concept or comparing the embeddings (encoded with some pre-trained language models) of the query and each concept directly. However, pre-trained word or query embeddings suffer from the Lexical Chase problem [28]– the surrounding context of a search query cannot be accurately established by considering only the words in the query. Therefore, retrieving a shortlist of concepts according to only the information in query words verbatim is not sufficient and tends to neglect truly relevant concepts. To address this challenge, inspired by [13], we propose to use a Relevant Words Generator (RWG) as the very first step in our shortlisting stage to expand the set of terms conveyed by a query.

We denote a concept c as a sequence of words (tokens), i.e., $c = \{w_1^c, w_2^c, ...\}$. Similarly, a query q is denoted by $q = \{w_1^q, w_2^q, ...\}$. Let V_C be a vocabulary of all the unique words that appear in the

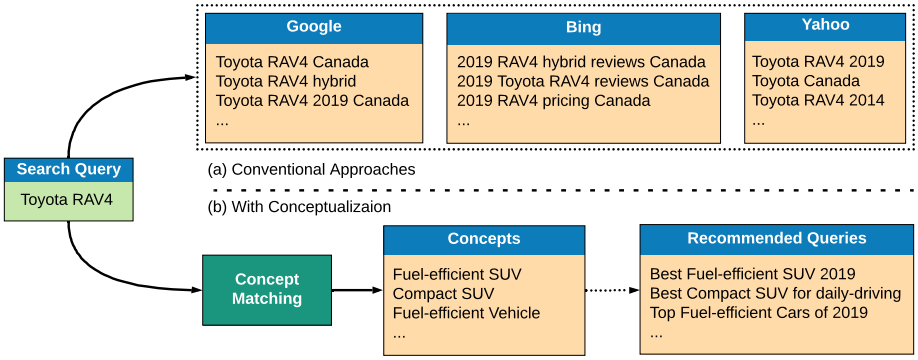

Figure 1: A comparison of the related search queries recommended a) by popular search engines and b) with concept matching. Note that even though the results of b) have fewer word overlaps with the original query, they are also likely to be attractive to the user as they match the user intent on a *conceptual* level.

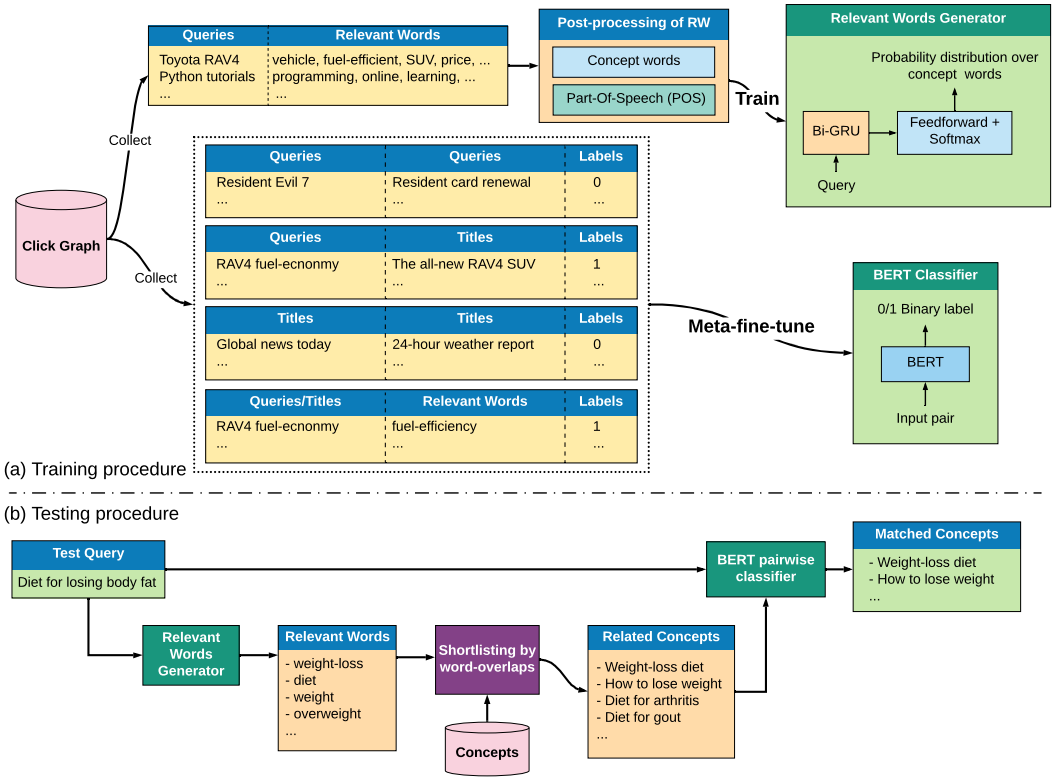

Figure 2: The training a) and testing b) procedures for our query-concept matching framework.

concepts. In reality, we normally have $m > |V_C|$, since the words in V_C can be combined to generate a large number of concepts. In other words, the size of V_C does not increase linearly with the number of concepts m. Therefore, as opposed to directly shortlisting candidate concepts from C, for a given input query, we first generate (select) the most relevant words in the vocabulary V_C, which is precisely what the RWG model achieves. The selected relevant words can then be used to fast retrieve a shortlist of candidate concepts.

Given an input query q and V_C, we learn a model θ such that the log probabilities of choosing the words, which are relevant to q, are maximized. Suppose that the set R^q contains all the target relevant words of q, our objective then becomes

$$\max_{\theta} \sum_{q \in Q} \sum_{w_c \in R^q} \log p(w_c|q;\ \theta), \tag{4}$$

where the generated relevant words $w_c \in V_C$. The RWG model [13] includes Bi-directional Gated Recurrent Unit (GRU) [6] that encodes a query word-by-word to learn a contextual representation, followed by a fully-connected layer, a Dropout layer [34], and a Softmax operation. The output is a probability distribution over all words in V_C.

Let us define $\hat{R}_k^q$ 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 $\hat{C}_k^q$ as the set of all concepts where each concept has at least one word that is in $\hat{R}_k^q$. Finally, we define $\hat{C_s^q}$ as the output set of candidate concepts, where $\hat{C_s^q} \subseteq \hat{C}_k^q$. $\hat{C_s^q}$ is obtained as follows:

$$\hat{C_s^q} = \{c \mid c \in \hat{C}_k^q,\ score(c) \geq \gamma\}, \tag{5}$$

where for each concept $c \in \hat{C}_k^q$, we compute an overlap score as the percentage of unique words in c that are also in $\hat{R}_k^q$, the set of predicted relevant words for query q, i.e.,

$$score(c) = \frac{1}{|\{c\}|} \sum_{w_c \in \{c\}} \lambda(w_c, \hat{R}_k^q), \tag{6}$$

where $\{c\}$ denotes the set of unique words in c and $\lambda(w_c, \hat{R}_k^q)$ is an indicator function that returns 1 if w_c is in $\hat{R}_k^q$ and 0 otherwise.

2.2 Meta-Learned Matching Model

Without a large amount of labelled training data, we could not directly learn the query-concept matching model f under Eq. 3. Fortunately, since a significant portion of concepts is originated and extracted from click histories [22], we assume that there exist distributional similarities between the task of query-concept matching and other pairwise text-matching tasks that can be derived from a large click graph.

For this reason, we take a meta-learning approach in the second stage to train f. A key assumption behind optimization-based meta-learning algorithms [9, 25] is that in a machine learning problem, there exists a distribution of tasks $p(\mathcal{T})$. According to the idea of Model-Agnostic Meta-Learning (MAML) [9], it is possible for a model ϕ to learn to adapt to $p(\mathcal{T})$ as opposed to a sampled task $\mathcal{T}_i$ by minimizing the following loss function:

$$\min_{\phi} \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}(\phi - \alpha \nabla_{\phi} \mathcal{L}(\phi, \mathcal{D}_i^{train}), \mathcal{D}_i^{test}), \tag{7}$$

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. 7 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 for 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. 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* [9], where

$$\hat{\phi} = \arg\max_{\phi} \log p(\phi | \mathcal{D}_j^{fine-tune}, \phi). \tag{8}$$

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

Instead of performing the second-order differentiation in Eq. 7, we utilize Reptile [25], a first-order meta-learning algorithm to learn ϕ. Reptile converts the outer loss function into a simple step in the direction of $\nabla_{\phi} \mathcal{L}(\phi, \mathcal{D}_i^{train})$ after K batches of training on $\mathcal{T}_i$. 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, and then 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

$$\phi \leftarrow \phi + \alpha(\tilde{\phi} - \phi), \tag{9}$$

where $\tilde{\phi}$ is the copied model after the task-learning phase and α is the meta-learning rate. Under the definition of Reptile, the objective for query-concept matching becomes the objective for the task-learning phase,

$$\max_{\tilde{\phi}} \sum_{u_i, u_j \in U} \log p(r_{u_i,u_j} | u_i, u_j;\ \tilde{\phi}), \tag{10}$$

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

$$\max_{\phi} \log p(\phi | \mathcal{D}^{meta}), \tag{11}$$

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

To establish a good starting point for learning ϕ, we utilize the pre-trained BERT model [7], which is shown to capture generalizable semantic knowledge that could easily be transferred to other NLP tasks through fine-tuning. BERT contains 12 layers of Transformer [39] 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. 10 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. We *meta-train* BERT under the Reptile-learning procedures defined by Eq. 10 and 11.

3 GENERATING DATA FROM CLICK GRAPHS

All necessary training data are automatically mined from a click graph without any human labelling efforts. A click graph is a bipartite graph that records the click histories of many users in a search engine and is easy to retrieve. Each vertex in a click graph corresponds to either a search query or a document title. An edge between a query vertex and a title vertex indicates that the user who issued the query clicked through the corresponding title. We believe a click graph is an excellent source of collective intelligence, where the click behavior naturally reflects the relations between various entities in the open domain. Therefore, we generate the training data by following a k-hop breadth-first 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 then denote the k-*hop neighbors set* for a vertex v in a click graph $\mathcal{G}$

by $\mathcal{S}^{v,\mathrm{k}}_{[.]}$, where [.] indicates the type(s) of vertices to be included. For example, $\mathcal{S}^{v,\mathrm{k}}_{[Q]}$ includes only neighboring queries while $\mathcal{S}^{v,\mathrm{k}}_{[Q,T]}$ includes both queries and document titles. Regardless of whether v is a query or 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 query-title edge in $\mathcal{G}$ is also weighted, where the weight reflects the number of click-throughs (by any user) from the query to the title.

3.1 Generating Relevant Words Data

Let $\mathcal{R}^q_t$ denote the set of target relevant words for a query q. For each query vertex q, we first retrieve its 3-hop neighboring queries set $\mathcal{S}^{q,3}_{[Q]}$. Next, we iterate through every query $\hat{q}$ in $\mathcal{S}^{q,3}_{[Q]}$ and perform Part-Of-Speech (POS) tagging using the StanfordCoreNLP tagger [23]. For every word w in $\hat{q}$, we add it to $\mathcal{R}^q_t$ only if it satisfies the following conditions:

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

Once $\mathcal{R}^q_t$ is formed in the above way, we add the pair $(q, \mathcal{R}^q_t)$ as a new sample into the RWG dataset, if $\mathcal{R}^q_t$ is non-empty. We repeat the above process for every query vertex q in the click graph to derive the complete dataset, from which we then split into train, dev and test sets.

3.2 Generating Data for Meta-learning

Meta-learning requires a set of tasks that are sampled from the same underlying task distribution. Based on Eq. 10, we derive and sample the following tasks from the click graph:

- **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 from $\mathcal{S}^{q,2}_{[Q]}$. For the next two tasks, positive instances are created in the same fashion, though with different inputs and 2-hop neighbor sets.
- **Query-to-title (Q2T)**, where the goal is to classify whether a query is related to a document title.
- **Title-to-title (T2T)**, where we decide whether two document titles are related.
- **Query/title-to-word (QT2W)**. The first input is a sequence, either a query or title, and the second input is a word. We create positive instances by pairing an input vertex v to all the relevant words in $\mathcal{S}^{v,3}_{[Q,T]}$, following the same procedure in Sec. 3.1. Yet, the relevant words here are no longer constrained by Condition (1) from Sec. 3.1.

We sample negative instances by randomly selecting other inputs of the same type, i.e., for each positive pair (q, q^+) from the Q2Q task, we randomly select a q^- from the set of all unique queries Q. The same procedure applies 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.

Table 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		

4 EXPERIMENTATION

4.1 Datasets

Before deriving any data from the click graph, we randomly sample 398, 447 queries from it as a held-out test set for query-concept matching. 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 procedures in Sec. 3.

Table 1 and 2 provide useful statistical information on the datasets. To ensure an even contribution from every task when meta-learning the text-matching model, we constrain the train/dev set of each task to have the same size by randomly pruning larger datasets.

4.2 Baseline Models

We compare our 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 [32]. We set a cosine similarity threshold of 0.75 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 0.75. We select this threshold because on the test set, it results in a final coverage similar to our proposed framework.

ELMo-pre-train. As a contextualized representation, ELMo [26] incorporates the context of the sequence when generating word embeddings. We use the pre-trained Chinese ELMo model with a hidden size of 1024, which is provided by [5, 8]. The procedure to select the matching concepts is the same as the MoWE baseline.

BERT-pre-train. 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-train baseline.

RW. The Relevant Words (RW) baseline is only our proposed first stage. We 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-ft. We fine-tune a pre-trained BERT-base model as a matching model on only the Q2Q task. We then pair it with a

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

Table 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

Table 3: Top-N recall scores on the RWG test set.

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

MoWE shortlisting scheme, which is constructed the same way as the MoWE baseline except that we do not set a similarity threshold.

RW-BERT-ft. We adopt the same fine-tuned BERT model from MoWE-BERT-ft and instead pair it with our RW first stage.

We name our proposed two-stage framework **RW-BERT-meta**. We refer interested reader to the supplementary information section for more details about the overall training setup.

4.3 Offline Evaluation

4.3.1 Evaluating the RWG model. 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 pre-trained representations. Therefore, we report the top-N recall scores on the test set of the RWG model. For our model, the top-N recall score as the percentage of truth relevant words that appear in the top-N predicted words. 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 3 reports the results of this experiment. Some coverages are not 1.0 because we reject inputs that contain Out-Of-Vocabulary (OOV) words.

4.3.2 Offline evaluation using click graphs. In the absence of a labelled test set, we evaluate the results of query-concept matching using our click graph. Considering the fact that many concepts are also popular queries, we thoroughly examine our click graph and find that it contains 2997 concepts as query vertices. Under our previous assumption, the hop distance between two vertices is a rough estimate for their relatedness. Therefore, we propose 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. 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 H hops. Note that the click graph used for evaluation here is the original graph containing all the test queries. With the derived truth labels, we could then compute the macro-averaged F1 score on the query-concept matching results. The definition of this metric is introduced in the supplementary section. We experiment with several H values and report the F1 scores in Table 4.

Table 4: F1 scores with different hop limits (H).

	$H = 6$	$H = 8$	$H = 10$	$H = 12$	$H = 14$
MoWE	0.702	0.657	0.627	0.605	0.589
ELMo-pre-train	0.750	0.699	0.626	0.561	0.521
BERT-pre-train	0.827	0.755	0.691	0.637	0.608
RW	0.494	0.497	0.494	0.491	0.491
MoWE-BERT-ft	0.595	0.600	0.608	0.611	0.631
RW-BERT-ft	0.682	0.709	0.725	0.738	0.743
RW-BERT-meta	**0.873**	**0.866**	**0.854**	**0.847**	**0.842**

Table 5: Human evaluation results on 500 randomly selected test instances.

	# of 1s assigned	# of 2s assigned	Total score
MoWE	436	81	598
ELMo-pre-train	199	58	315
BERT-pre-train	86	32	150
RW	532	62	656
MoWE-BERT-ft	285	29	343
RW-BERT-ft	435	31	497
RW-BERT-meta	**621**	**117**	**855**

4.4 Human Evaluation

Since the offline evaluation does not explicitly reflect which model is better at predicting *isA* relations, we further conduct human evaluations. Specifically, we randomly select 500 test instances and take the top-3 matched concepts of every model [2]. We then hire 2 human judges through crowdsourcing and ask each one to evaluate half of the instances according to the following criteria,

- For a test instance, the judge reviews the top-3 concepts matched by each model without given any knowledge about the models and assigns a score for each concept.
- A concept receives a score of 2 if there exists an *isA* relation between it and the input query. If a concept only matches part of the query, 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 5. In Sec. 5, We discuss the implications behind these results and conduct case studies to get a more intuitive visualization of the matching results.

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

4.5 Online A/B Testing

We conduct online A/B testing on the Tencent QQ browser to evaluate the usefulness of our framework in a production environment. For a user's search query, we discover its matching concepts and present them at the end of the article which the user clicked on, i.e., we directly recommend the matched concepts to the user. Fig. 3 reports the results of the A/B test. The control group here is the MoWE baseline. We sample 50 hours of click behavior from real users, where each point in the plot represents the average number of clicks on the recommended concepts in an hour. We also report the interquartile range box-plot with the 25th to the 75th percentiles.

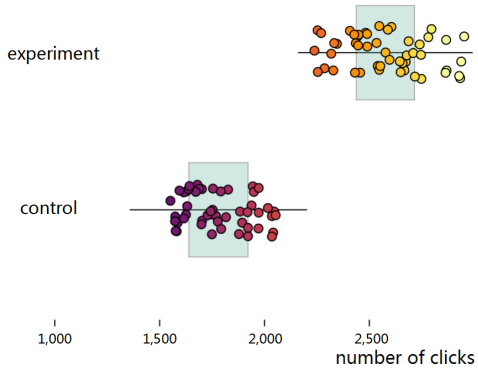

Figure 3: Average number of clicks per hour on the recommended concepts. The experiment groups is our framework and the control group is the baseline.

5 RESULT ANALYSIS & DISCUSSION

Table 3 suggests that the RWG model easily outperforms the other two pre-trained word representations. This indicates that the word-relatedness information derived from a click graph is different from the word similarities learned by pre-trained embeddings. For the results of query-concept matching, we observe in Table 4 that our two-stage setup outperforms all baselines by a large margin regardless of the hop limit H. Specifically, by adding a second stage text-matching model to the RW baseline, the performance increases significantly.

We believe that while the imaginative nature of the RWG model enables us to discover additional relevant words, it could "wander-off" too far and match completely irrelevant concepts. Therefore, another stage of quality assurance is crucial, and our meta-learned BERT text-matching model is a better candidate for this role compared to the simple fine-tuned version (RW-BERT-ft). According to the human evaluation results in Table 5, our framework scores highest in every category. In particular, it receives more scores of 2 than the baselines, which proves its superiority in discovering matching *isA* relations between queries and concepts. We observe the same pattern in the A/B testing results, where the concepts matched by our framework consistently attract more clicks than the baseline.

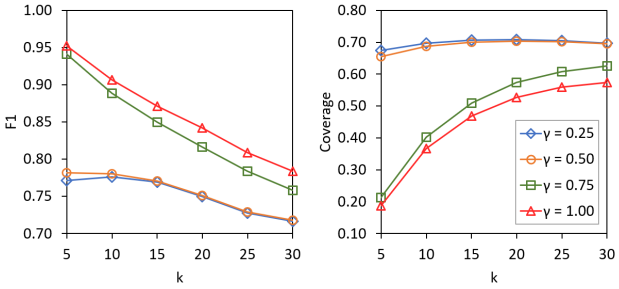

Figure 4: F1-score and coverage results produced with different combinations of k and γ values.

5.1 Hyper-parameter Sensitivity

We run the offline evaluation procedure with $H = 10$ to observe how the performance changes under different hyper-parameter settings. We test several values of k, which controls how many top relevant words to take from the output of the RWG model, and γ, which determines the percentage threshold of word-overlaps between the candidate concepts and the predicted relevant words. We report the changes in F1 scores and coverage, i.e., the percent of test instances that we could find at least one matching concept.

From Fig. 4, we first observe that a larger value of γ produces higher F1 scores and lower coverage. This makes sense because a lower γ means we allow a candidate concept to contain more words that are not from the predicted relevant words, which increases the chance of mismatches. Another trend we notice is that decreasing k values often leads to an increase in F1 scores. However, this does not mean that we should prefer a smaller k because smaller k values also produce a much lower coverage. In a real-world application, we want our framework to handle as many queries as possible while maintaining decent performance metrics. We set $k = 15$ to strike a balance between F1 score and coverage.

5.2 Case Study

In Table 6, we show a representative example of the top-10 relevant words generated by our RWG model and compare with the top-10 most similar words found by the Tencent AI Lab (TAL) word embeddings. In general, the RWG model discovers more words related to the input query on a higher conceptual-level, especially the highlighted ones. For instance, 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, taking the average of word embeddings for the query cannot capture its overall meaning and the majority of related words are only similar to "screen".

In Table 7, we show the top-3 matched concepts found by our framework and competitive baselines for a test query. Overall, our framework is superior at both finding potential candidates and picking out the best matching concepts. Here, Omen and Legion are gaming laptop models from HP and Lenovo, respectively. Clearly, the MoWE baseline does not have this knowledge and base its matching solely on the word "Legion". In contrast, our proposed RW stage captures this relation. Then, the second stage reliably filters out non-matching concepts such as "Laptop keyboard".

Table 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.

Table 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-ft	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.

6 RELATED WORK

6.1 Search Query Understanding

We review related works related to search query understanding from the perspectives of query expansion, query reformulation, query generation and query-concept matching. Query expansion tackles the Lexical Chase problem. [10] performs expansions through a relation graph. [11, 12, 28, 29] approach the problem from the perspective of statistical machine translation.

Query reformulation re-writes a search query such that it is easier to process, while maintaining the original meaning. Early methods either delete unnecessary terms in a query [18] or substitute ambiguous terms [37]. [45] proposes an active-learning-based method. SimRank and SimRank++ [2, 16] compute similarities between queries using the links in a click graph. More recently, methods based on machine translation [28] or recurrent neural networks [15] further improves the quality of the reformulated queries.

When generating search queries directly using Sequence-to-Sequence models, the process of query understanding is implicitly captured by the encoder part of the model. [41] considers the compression of E-commerce product titles and the generation of search queries in a multi-task learning setup. [15, 44] perform generative query re-writing, while [14] represents documents as graphs and reverse-engineers the most appropriate queries from them. Finally, [13] proposes a two-stage generative framework for query recommendation, where the Relevant Words Generator model is proven to be extremely useful in mitigating the Lexical Chase problem.

Query-concept matching is a relatively unexplored topic due to the need of a pre-defined concept set. [42] proposes a Bayesian inference mechanism for query-concept matching. Specifically, the matching degree between a query and a concept is derived from the conditional probabilities between their related instances and attributes. [33] refines the Probase [43] ontology by adding more relations extracted from web documents, then presents a random walk strategy to discover the matching concepts. [22] proposes a probabilistic approach according to the co-occurrence of context words as well as a similarity-based method using TF-IDF vectors.

6.2 Meta-Learning

Meta-learning studies the problem of learning how to learn [24, 38]. 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 [4, 30]. [1] transfers this idea to neural networks and proposes an optimizer-optimizee setup, where each component is learned with an iterative gradient-descent procedure. [20] 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. [27] proposes an LSTM meta-learner to learn an optimization procedure for few-shot image classification. [21] develops an SGD-like meta-learning process and experiment on few-shot regression and reinforcement learning problems. MAML [9] is another popular approach that does not impose a constraint on the architecture of the learner. Finally, Reptile [25] 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. [19] meta-learns a Siamese network to classify whether two images are from the same class. Matching networks [40] 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 [31] learn an entirely new metric space for comparing the similarities between inputs. [36] proposes that separating embedding learning and relation matching further improves the accuracy of meta-learned few-shot classification models.

7 CONCLUSION

In this paper, we propose a meta-learned neural framework for matching a search query to its high-level concepts. We begin by discovering potentially matching candidates with a novel shortlisting scheme, where a recurrent Relevant Words Generator performs context-completion for a query. To ensure that each concept is indeed a match to the query, we meta-fine-tune a BERT pairwise text-matching model with the Reptile meta-learning algorithm, which performs binary classification on a query-concept pair to determine the final matching degree. We train our framework with data derived from a large click graph that contains over 7 million click histories, without any manual labelling. Through offline evaluations, human assessments, online tests and case studies, we have verified the effectiveness of our proposed framework, especially its

ability to infer and discover higher-level concepts for search queries, as compared to a range of competitive baselines. We conclude that a combination of deep neural language models and meta-learning may open up avenues for natural language conceptualization and inference, an ability that is native to human intelligence and is critical to enhancing human-computer interaction through search engines.

REFERENCES

[1] Marcin Andrychowicz, 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*. 3981–3989.

[2] Ioannis Antonellis, Hector Garcia Molina, and Chi Chao Chang. 2008. Simrank++: query rewriting through link analysis of the click graph. *Proceedings of the VLDB Endowment* 1, 1 (2008), 408–421.

[3] Sören Auer, 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*. Springer, 722–735.

[4] Samy Bengio, 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, 6–8.

[5] Wanxiang Che, 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, 55–64.

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

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

[8] Murhaf Fares, 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, 271–276.

[9] Chelsea Finn, 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, 1126–1135.

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

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

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

[13] Fred X Han, Di Niu, Haolan Chen, Kunfeng Lai, Yancheng He, and Yu Xu. 2019. A deep generative approach to search extrapolation and recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM.

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

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

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

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

[18] Rosie Jones 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, 435–436.

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

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

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

[22] Bang Liu, Weidong Guo, Di Niu, Chaoyue Wang, Shunnan Xu, Jinghong Lin, Kunfeng Lai, and Yu Xu. 2019. 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*. ACM.

[23] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*. 55–60.

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

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

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

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

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

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

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

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

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

[33] Yangqiu Song, Haixun Wang, Zhongyuan Wang, Hongsong Li, and Weizhu Chen. 2011. Short text conceptualization using a probabilistic knowledgebase. In *Twenty-Second International Joint Conference on Artificial Intelligence*.

[34] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* 15 (2014), 1929–1958.

[35] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*. ACM, 697–706.

[36] Flood Sung, 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*. 1199–1208.

[37] Egidio Terra 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, 50–58.

[38] Sebastian Thrun and Lorien Pratt. 2012. *Learning to learn*. Springer Science & Business Media.

[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

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

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

[42] Zhongyuan Wang, Kejun Zhao, Haixun Wang, Xiaofeng Meng, and Ji-Rong Wen. 2015. Query understanding through knowledge-based conceptualization. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

[43] Wentao Wu, Hongsong Li, Haixun Wang, and Kenny Q Zhu. 2012. Probase: A probabilistic taxonomy for text understanding. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*. ACM, 481–492.

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

[45] Wei Vivian Zhang, 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, 853–854.

8 SUPPLEMENTARY INFORMATION

8.1 Training Setup

To pre-processing the click graph, we first prune any edges that do not appear more than once to avoid noise caused by mis-clicks. We then keep the top-3 weighted outgoing edges of each query vertex, because we found that over 90% of vertices have less than 3 outgoing edges.

We utilize the FAISS [17] tool for fast similarity searches. 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 classifiers is 0.5.

We implement both stages of our model using Pytorch 0.4 and train with the Adam optimizer. 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 1. We initialize the embedding layer with the Tencent AI Lab 200d pre-trained word embeddings. 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 probability 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 possible learn-rate is 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.

In the task-learning phase of Reptile, we minimize the Cross-Entropy loss on the predicted labels. We set $K = 10$. 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.

8.2 Evaluation Setup

We derive the macro-averaged F1 score from Mean Average Precision (MAP) and Mean Average Recall (MAR) as,

$$\text{MAP} = \frac{1}{\mathcal{N}} \sum_{i=1}^{\mathcal{N}} \frac{TP_i}{TP_i + FP_i}, \tag{12}$$

$$\text{MAR} = \frac{1}{\mathcal{N}} \sum_{i=1}^{\mathcal{N}} \frac{TP_i}{TP_i + FN_i}, \tag{13}$$

$$\text{F1} = 2 \cdot \frac{\text{MAP} \cdot \text{MAR}}{\text{MAP} + \text{MAR}}, \tag{14}$$

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