Context-Aware Deep Model for Entity Recommendation System in Search Engine at Alibaba

Qianghuai Jia, Ningyu Zhang, Nengwei Hua
Alibaba Group
Hangzhou. China
qianghuai.jqh@alibaba-inc.com
ningyu.zny@alibaba-inc.com
nengwei.huanw@alibaba-inc.com

ABSTRACT: *Entity recommendation, providing search users with an improved experience via assisting them in finding related entities for a given query, has become an indispensable feature of today's search engines. Existing studies typically only consider the queries with explicit entities. They usually fail to handle complex queries that without entities, such as "what food is good for cold weather", because their models could not infer the underlying meaning of the input text. In this work, we believe that contexts convey valuable evidence that could facilitate the semantic modeling of queries, and take them into consideration for entity recommendation. In order to better model the semantics of queries and entities, we learn the representation of queries and entities jointly with attentive deep neural networks. We evaluate our approach using large-scale, realworld search logs from a widely used commercial Chinese search engine. Our system has been deployed in ShenMa Search Engine [1] and you can fetch it in UC Browser of Alibaba. Results from online A/B test suggest that the impression efficiency of click-through rate increased by 5.1% and page view increased by 5.5%.*

Keywords: Entity Recommendation, Deep Neural Networks, Query Understanding, Knowledge Graph, Cognitive Concept Graph

Received: 10 September 2019, Revised 30 November 2019, Accepted 23 December 2019

DOI: 10.6025/jmpt/2020/11/1/23-35

1. Introduction

Over the past few years, major commercial search engines have enriched and improved the user experience by proactively presenting related entities for a query along with the regular web search results. Figure 1 shows an example of Alibaba ShenMa search engine's entity recommendation results presented on the panel of its mobile search result page.

Existing studies [2, 7] in entity recommendation typically consider the query containing explicit entities, while ignoring those

[1] m.sm.cn

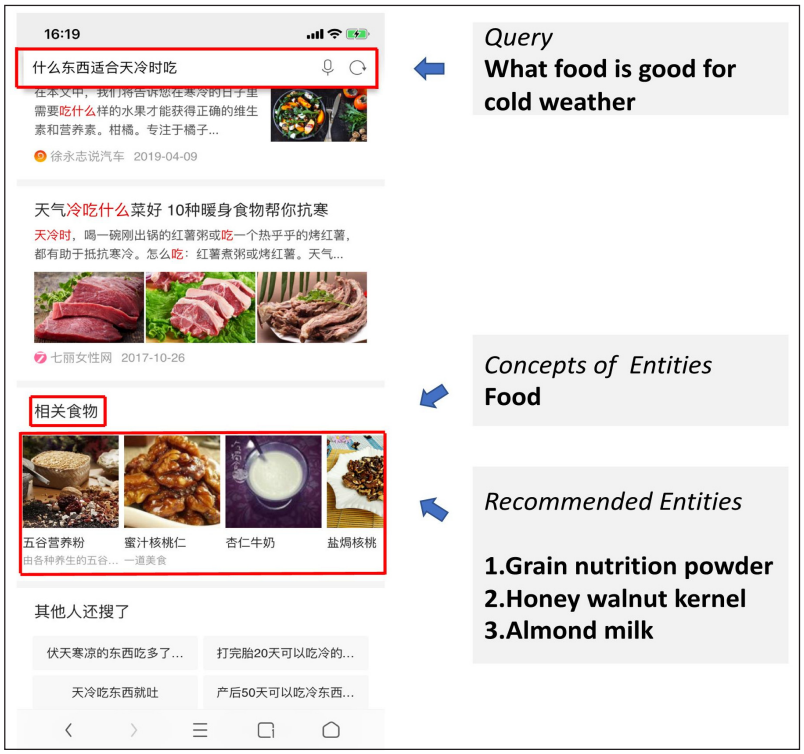

Figure 1. Example of entity recommendation results for the query "what food is good for cold weather

queries without entities. A main common drawback of these approaches is that they cannot handle well the complex queries, because they do not have informative evidence other than the entity itself for retrieving related entities with the same surface form. Therefore, existing entity recommendation systems tend to recommend entities with regard to the explicitly asked meaning, ignoring those queries with implicit user needs. Through analyzing hundreds of million unique queries from search logs with named entity recognition technology, we have found that more than 50% of the queries do not have explicit entities. In our opinion, those queries without explicit entities are valuable for entity recommendation.

The queries convey insights into a user's current information need, which enable us to provide the user with more relevant entity recommendations and improve user experience. For example, a user's search intent behind the query "what food is good for cold weather" could be a kind of food suitable to eat in cold weather. However, most of the entities recommended for the query are mainly based on entities existed in the query such as given the query "cake" and recommend those entities "cup-cakes," "chocolate" and so on, and there is no explicit entity called "good food for cold weather" at all. It is very likely that the user is interested in the search engine that is able to recommend entities with arbitrary queries.

However, recommending entities with such complex queries is extremely challenging. At first, many existing recommendation algorithms proven to work well on small problems but fail to operate on a large scale. Highly specialized distributed learning algorithms and efficient serving systems are essential for handling search engine's massive queries and candidate entities. Secondly, user queries are extremely complex and diverse, and it is quite challenging to understand the user's true intention. Furthermore, historical user behavior on the search engine is inherently difficult to predict due to sparsity and a variety of unobservable external factors. We rarely obtain the ground truth of user satisfaction and instead model noisy implicit feedback signals.

In this paper, we study the problem of context-aware entity recommendation and investigate how to utilize the queries without explicit entities to improve the entity recommendation quality. Our approach is based on neural networks, which maps both queries and candidate entities into vector space via large-scale distributed training.

We evaluate our approach using large-scale, real-world search logs of a widely used commercial Chinese search engine. Our

system has been deployed in ShenMa Search Engine and you can experience this feature in UC Browser of Alibaba. Results from online A/B test involving a large number of real users suggest that the impression efficiency of click-through rate (CTR) increased by 5.1% and page view (PV) increased by 5.5%.

The main contributions of our paper are summarized as follows:

• To the best of our knowledge, we are the first approach to recommend entities for arbitrary queries in large-scale Chinese search engine.

• Our approach is flexible capable of recommending entities for billions of queries.

• We conduct extensive experiments on large-scale, real-world search logs which shows the effectiveness of our approach in both offline evaluation and online A/B test.

2. Related Work

Previous work that is closest to our work is the task of entity recommendation. Entity recommendation can be categorized into the following two categories: First, for query assistance for knowledge graphs [16, 17], GQBE [9] and Exemplar Queries [13] studied how to retrieve entities from a knowledge base by specifying example entities. For example, the input entity pair {Jerry Yang, Yahoo!} would help retrieve answer pairs such as {Sergey Brin, Google}. Both of them projected the example entities onto the RDF knowledge graph to discover result entities as well as the relationships around them. They used an edge-weighted graph as the underlying model and subgraph isomorphism as the basic matching scheme, which in general is costly.

Second, to recommend related entities for search assistance. [2] proposed a recommendation engine called Spark to link a user's query word to an entity within a knowledge base and recommend a ranked list of the related entities. To guide user exploration of recommended entities, they also proposed a series of features to characterize the relatedness between the query entity and the related entities. [11] proposed a similar entity search considering diversity. [8] proposed to enhance the understandability of entity recommendations by captioning the results. [5] proposed a number of memory-based methods that exploit user behaviors in search logs to recommend related entities for a user's full search session. [7] propose a model in a multi-task learning setting where the query representation is shared across entity recommendation and context-aware ranking. However, none of those approaches take into account queries without entities.

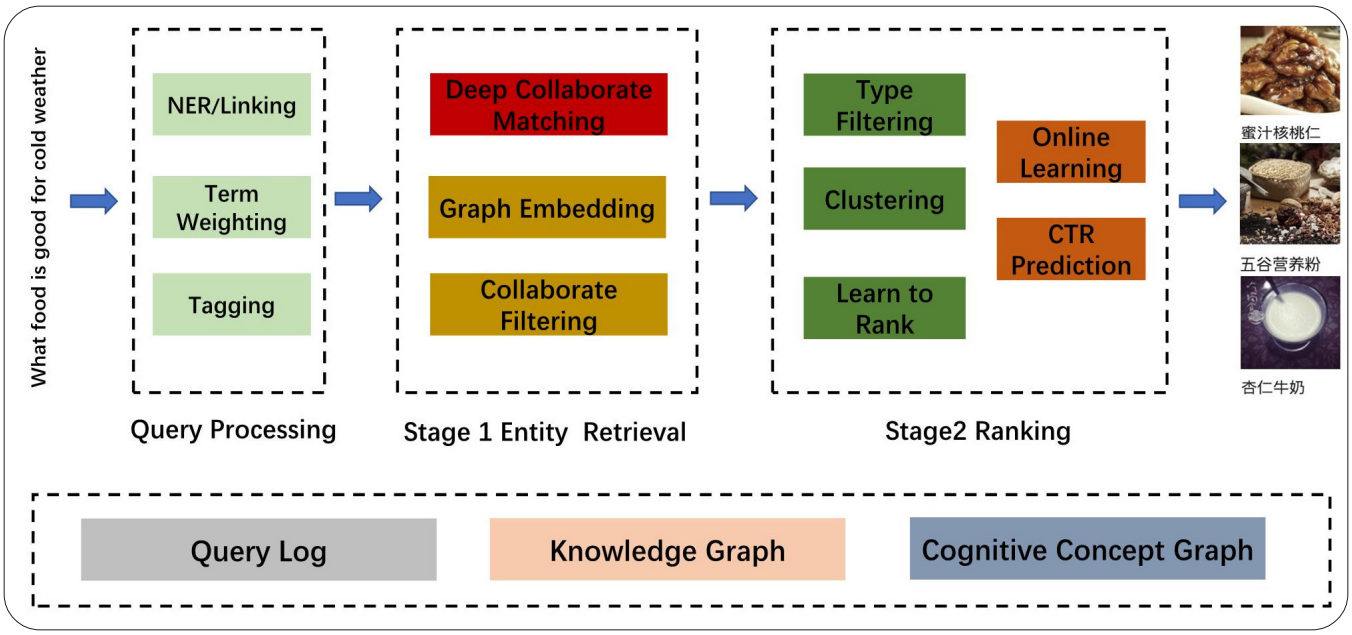

Figure 2. System overview of entity recommendation in ShenMa search engine at Alibaba, the red part is the focus of this paper

Our objective is to infer entities given diverse and complex queries for search assistance. Actually, there are little research papers that focus on this issue. In industry, there are three simple approaches to handle those complex queries. One is tagging the query and then recommend the relevant entities based on those tags. However, the tagging space is so huge that it is difficult to cover all domains. The second method is to use the query recommendation algorithm to convert and disambiguate the queries into entities, ignoring effect of error transmission from query recommendation. The last approach is to recall entities from the clicked documents. However, not all queries have clicked documents. To the best of our knowledge, we are the first end-to-end method that makes it possible to recommend entities with arbitrary queries in large scale Chinese search engine.

3. System Overview

The overall structure of our entity recommendation system is illustrated in Figure 2. The system is composed of three modules: query processing, candidate generation and ranking. The query processing module at first preproceses the queries, extract entities (cannot extract any entities for complex queries) and then conceptualize queries. The candidate generation module takes the output of query processing module as input and retrieves a subset (hundreds) of entities from the knowledge graph. For a simple query with entities, we utilize heterogeneous graph embedding [6] to retrieve relative entities. For those complex queries with little entities, we propose a deep collaborative matching model to get relative entities. These candidates are intended to be generally relevant to the query with high recall. The candidate generation module only provides broad relativity via multi-criteria matching. The similarity between entities is expressed in terms of coarse features. Presenting a few "best" recommendations in a list requires a fine-level representation to distinguish relative importance among candidates with high precision. The ranking module accomplishes this task by type filtering, learning to rank, and click-through rate estimation. We also utilize online learning algorithm, including Thompson sampling, to balance the exploitation and exploration in entity ranking. In the final product representation of entity recommendation, we utilize the concept of entities to cluster the different entities with the same concept in the same group to represent a better visual display and provide a better user experience. In this paper, we mainly focus on candidate generation, the first stage of entity recommendation and present our approach (red part in Figure 2), which can handle complex queries.

4. Preliminaries

In this section, we describe the large knowledge graph that we use to retrieve candidate entities and cognitive concept graph that we use to conceptualize queries and entities.

4.1 Knowledge Graph

Shenma knowledge graph[2] is a semantic network that contains ten million of entities, thousand types and billions of triples. It has a wide range of fields, such as people, education, film, tv, music, sports, technology, book, app, food, plant, animal and so on. It is rich enough to cover a large proportion of entities about worldly facts. Entities in the knowledge graph are connected by a variety of relationships.

4.2 Cognitive Concept Graph

Based on Shenma knowledge graph, we also construct a cognitive concept graph which contains millions of instances and concepts. Different from Shenma knowledge graph, cognitive concept graph is a probabilistic graph mainly focus on the Is-A relationship. For example, "robin" is-a bird, and "penguin" is-a bird. Cognitive concept graph is helpful in entity conceptualization and query understanding.

5. Deep Collaborative Match

In this section, we first introduce the basics of the deep collaborative match and then elaborate on how we design the deep model architecture.

5.1 Recommendation as Classification

Traditionally, major search engines recommend related entities based on their similarities to the main entity that the user searched. [7] have detailed explained the procedure of entity recommendation in the search engine, including entity linking,

[2] kg.sm.cn

related entity discovery and so on. Unlike traditional methods, we regard recommendation as large-scale multi-classification where the prediction problem becomes how to accurately classify a specific entity e_i among millions of entities from a knowledge graph V based on a user's input query Q,

$$P(e_i|Q) = \frac{u_i\, q}{\Sigma_{j \in V}\, u_j\, q}$$

where $q \in \mathbb{R}^N$ is a high-dimensional "embedding" of the user's input query, $u_j \in \mathbb{R}^N$ represents each entity embedding and V is the entities from knowledge graph. In this setting, we map the sparse entity or query into a dense vector in $\mathbb{R}^N$. Our deep neural model try to learn the query embedding via the user's history behavior which is useful for discriminating among entities with a softmax classier. Through joint learning of entity embeddings and query embeddings, the entity recommendation becomes the calculation of cosine similarity between entity vectors and query vectors.

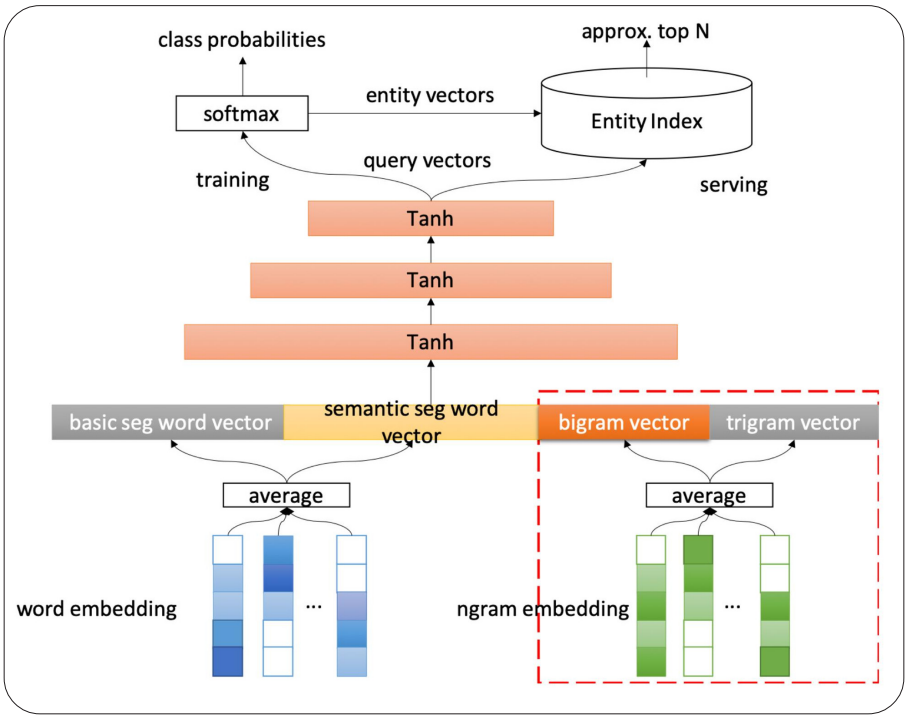

Figure 3. Base deep match model

5.2 Base Deep Match Model

Inspired by skip-gram language models [12], we map the user's input query to a dense vector representation and learn high dimensional embedding for each entity in a knowledge graph. Figure 3 shows the architecture of the base deep match model.

Input Layer. Input layer mainly contains the features from the input query, we first use word segmentation tool[3] to segment queries, then fetch basic level tokens and semantic level tokens[4], and finally combine all the input features via the embedding technique, as shown below:

• **Word Embedding:** Averaging the embedding of both the basic level tokens and semantic level tokens, and the final embedding dimension is 128.

[3] AliWS, which is similar to jieba segmentation tool and uses CRF and user-defined dictionary to segment queries.

[4] Tokens that in the same entity or phrase will not be segmented.

• **N-gram Embedding:** Inspired by fasttext [10], we add ngram ($n = 2, 3$) features to the input layer to import some local temporal information. The dimension of ngram embedding is also 128.

Fully-Connected Layer. Following the input layer, we utilize three fully connected layers (512-256-128) with tanh activation function. In order to speed up the training, we add batch normalization to each layer.

Softmax Layer. To efficiently train such a model with millions of classes, we apply sampled softmax [1] in our model. For each example, the cross-entropy loss is minimized for the true label and the sampled negative classes. In practice, we sample 5000 negatives instances.

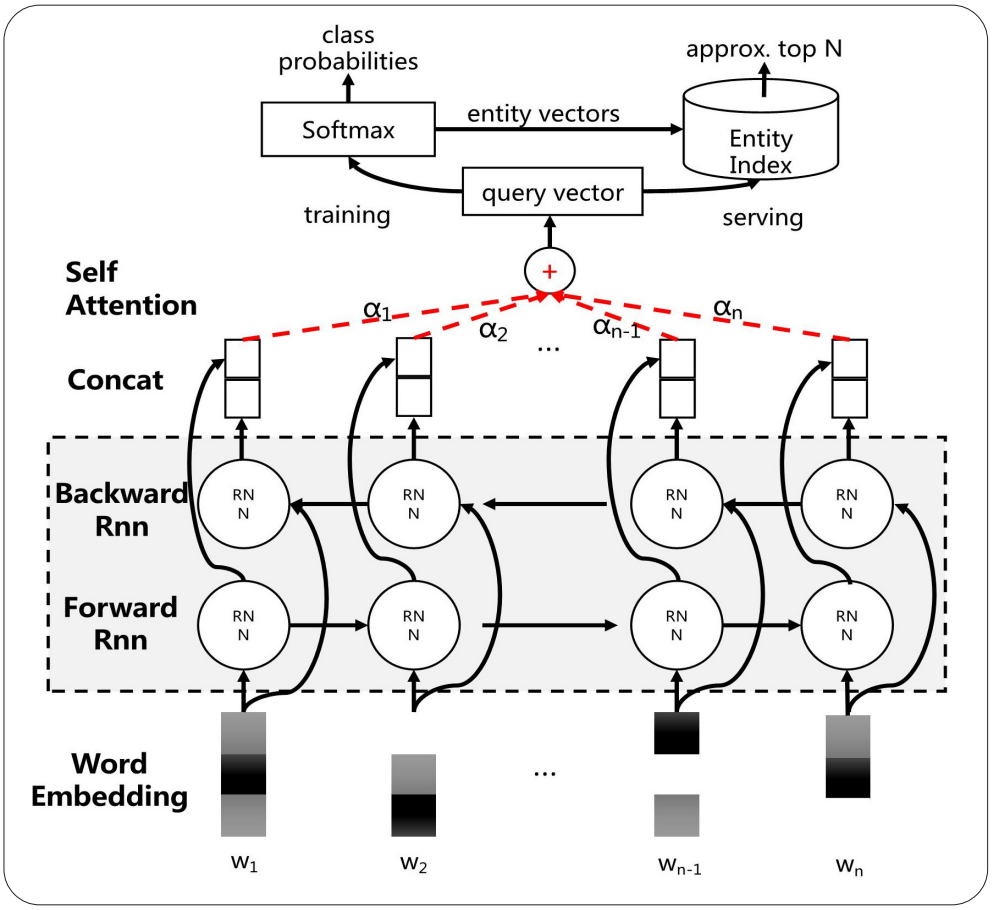

Figure 4. Enhanced deep match model

Online Serving. At the serving time, we need to compute the most likely K classes (entities) in order to choose the top K to present to the user. In order to recall the given number of entities within ten milliseconds, we deploy the vector search engine[5] under the offline building index. In practice, our model can generate query embedding within 5ms and recall related entities within 3ms.

5.3 Enhanced Deep Match Model

The above base model also remains two problems of on the semantic representation of the input query: 1) ignoring the global temporal information, which is important for learning query's sentence-level representation; 2) different query tokens contribute equally to the final input embedding, which is not a good hypnosis. For example, the entity token should be more important than other tokens such as stop words.

[5] The vector search engine is similar to the facebook's faiss vector search engine, and optimized in the search algorithm.

To address the first issue, we adopt the Bi-directional LSTM model to encode the global and local temporal information. At the same time, with the attention mechanism, our model can automatically learn the weights of different query tokens. Figure 4 shows the enhanced deep match model architecture.

The proposed model consists of two parts. The first is a Bidirectional LSTM, and the second is the self-attention mechanism, which provides weight vectors for the LSTM hidden states. The weight vectors are dotted with the LSTM hidden states, and the weighted LSTM hidden states are considered as an embedding for the input query. Suppose the input query has n tokens represented with a sequence of word embeddings.

$$Q = (w1, w2, ..., w_{n-1}, w_n)$$

where $w_i \in \mathbb{R}^d$ is the word embedding for the i-th token in the query. $Q \in \mathbb{R}^{n \times d}$ is thus represented as a 2-D matrix, which concatenates all the word embeddings together. To utilize the dependency between adjacent words within a single sentence, we use the Bidirectional LSTM to represent the sentence and concatenate h_{if} with h_{ib} to obtain the hidden state h_i:

$$h_i = [h_{if}, h_{ib}]$$

The number of LSTM's hidden unit ism. For simplicity, we concatenate all the hidden state h_i as $H \in \mathbb{R}^{n \times 2m}$. $H = [h_1, h_2, ..., h_{n-1}, h_n]$ With the self-attention mechanism, we encode a variable length sentence into a fixed size embedding. The attention mechanism takes the whole LSTM hidden states H as input, and outputs the weights $\alpha \in \mathbb{R}^{1 \times k}$:

$$\alpha = softmax(U \tanh(WH^T + b))$$

where $W \in \mathbb{R}^{k \times 2m}$, $U \in \mathbb{R}^{1 \times k}$, $b \in \mathbb{R}^k$. Then we sum up the LSTM hidden states H according to the weight provided by α to get the final representation of the input query.

$$q = \sum_{n=1}^{n} \alpha_i h_i$$

Note that, the query embeddings and entity embeddings are all random initialized and trained from scratch. We have huge amounts of training data which is capable of modeling the relativity between queries and entities.

6. Experiments

6.1 Data Sets

In this section, we illustrate how to generate the training samples to learn the query-entity match model. Training samples are generated from query logs and knowledge graph, which can be divided into four parts as shown below:

• **Query-Click-Entity:** Given a query, choose the clicked entities with relatively high CTR. In practice, we collect thousand millions of data from the query logs in the past two months.

• **Query-Doc-Entity:** We assume that high clicked doc is well matched to the query and the entities in title or summary are also related to the query. The procedure is 1) we first fetch the clicked documents with title and summary from the query log; 2) extract entities from title and summary via name entity recognition; 3) keep those high-quality entities. At last, we collect millions of unique queries.

• **Query-Query-Entity:** Given the text recommendation's well results, we utilize the entity linking method to extract entities from those results. We also collect millions of unique queries.

• **Query-Tag-Entity:** As to some specific queries, we will tag entity label to them and generate query-entity pairs. Here, we define hundreds of entity tags in advance.

After generating of query-entity pairs, we adopt the following data prepossessing procedures:

• **Low-quality Filter:** We filter low-quality entities via some basic rules, such as blacklist, authority, hotness, importance and so on.

• **Low-frequency Filter:** We filter low-frequency entities.

Figure 5. The top-N similar entities for given entities via entity embedding

• **High-frequency Sub-sampling:** We make sub-sampling to those high-frequency entities.

• **Shuffle:** We shuffle all samples.

Apart from user clicked data, we construct millions of query entity relevant pairs at the semantic level, which are very important for the model to learn the query's semantic representation. Finally, we generate billions of query-entity pairs and about one thousand billion unique queries.

Method	P@1	P@10	P@20	P@30
DNN	6.53	28.29	38.83	53.79
+ngram	7.25	30.76	41.57	56.49
att-BiLSTM	7.34	30.95	41.56	56.02

Table 1. The offline comparison results of different methods in large-scale, real-world search logs of a widely used commercial web search engine

6.2 Evaluation Metric

To evaluate the effectiveness of different methods, we use *Precision@M* following [18]. Derive the recalled set of entities for a query u as P_u ($|P_u| = M$) and the query's ground truth set as G_u. *Precision@M* are:

$$Precision@M(u) = \frac{|P_u \cap G_u|}{M}$$

6.3 Offline Evaluation

To evaluate the performance of our model, we compare its performance with various baseline models. From unseen and real online search click log, we collect millions of query-entity pairs as our test set (ground truth set). The evaluation results are shown in Table 1: **DNN** [3] is the base method with a DNN encoder; **+ngram** is method adding ngram features; **att-BiLSTM** is our method with BiLSTM encoder with attention mechanism. The DNN [3] is a very famous recommendation baseline and we re-implement the algorithm and modify the model for entity recommendation setting. Note that, there are no other baselines of entity recommendation for complex queries with no entities at all. **att-BiLSTM** is slightly better than **+ngram**. The reasons are mainly that a certain percentage of queries is without order and ngram is enough to provide useful information.

Our approach achieves the comparable results in the offline evaluation. These results indicate that our method benefits a lot from joint representation learning in queries and entities. Note that, we learn the embedding of queries and entities with random initialization. We believe the performance can be further improved by adopting more complex sentence encoder such as BERT[4] and XLNet[15] and inductive bias from structure knowledge[14] to enhance the entity representation, which we plan to address in future work.

6.4 Online A/B Test

We perform large-scale online A/B test to show how our approach on entity recommendation helps with improving the performance of recommendation in real-world applications. We first retrieve candidate entities by matching queries, then we rank candidate entities by a click-through rate (CTR) prediction model and Thompson sampling. The ranked entities are pushed to users in the search results of Alibaba UC Browser. For online A/B test, we split users into buckets. We observe and record the activities of each bucket for seven days.

Figure 6. Entity recommendation results from complex and diverse queries

We select two buckets with highly similar activities. For one bucket, we perform recommendation without the deep collaborative match model. For another one, the deep collaborative match model is utilized for the recommendation. We run our A/B test for seven days and compare the result. The page view (PV) and click through rate (CTR) are the two most critical metrics in real-world application because they show how many contents users read and how much time they spend on an application. In the online experiment, we observe a statistically significant CTR gain (5.1%) and PV (5.5%). These observations prove that the deep collaborative match for entity recommendation greatly benefits the understanding of queries and helps to match users with their potential interested entities better. With the help of a deep collaborative match, we can better capture the contained implicit user's need in a query even if it does not explicitly have an entity. Given more matched entities, users spend more times and reading more articles in our search engine.

6.5 Qualitative Analysis

We make a qualitative analysis of the entity embeddings learned from scratch. Interestingly, we find that our approach is able to capture the restiveness of similar entities. As Figure 5 shows, the entities "Beijing University," "Fudan University" are similar to

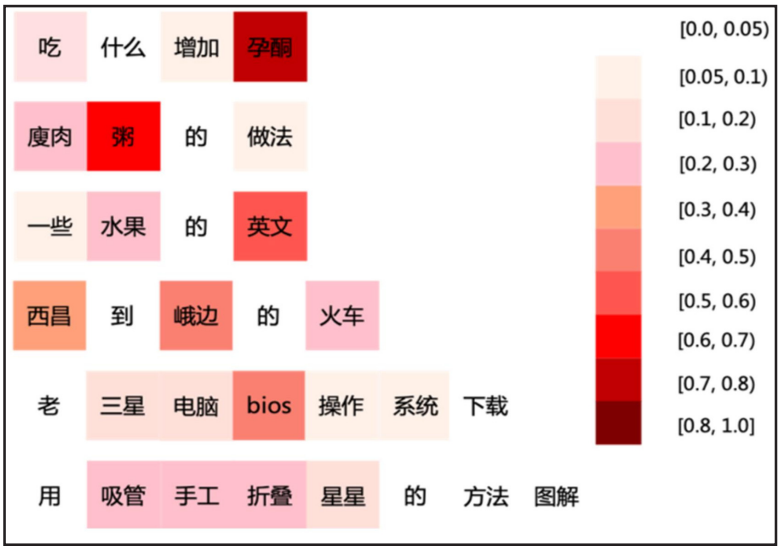

Figure 7. Attention weights visualization of six random queries from search log

the entity "Tsinghua University." Those results demonstrate that our approach's impressive power of representation learning of entities[6]. It also indicates that the text is really helpful in representation learning in knowledge graph.

We also make a qualitative analysis of the query embeddings. We find that our approach generates more discriminate query embedding for entity recommendation due to the attention mechanisms. Specifically, we randomly selected six queries from the search log and then visualize the attention weights, as shown in Figure 7. Our approach is capable of emphasizing those relative words and de-emphasizing those noisy terms in queries which boost the performance.

6.6 Case Studies

We give some examples of how our deep collaborative matching takes effect in entity recommendation for those complex queries. In Figure 6, we display the most relative entities that are retrieved from the given queries. We observe that (1) given the

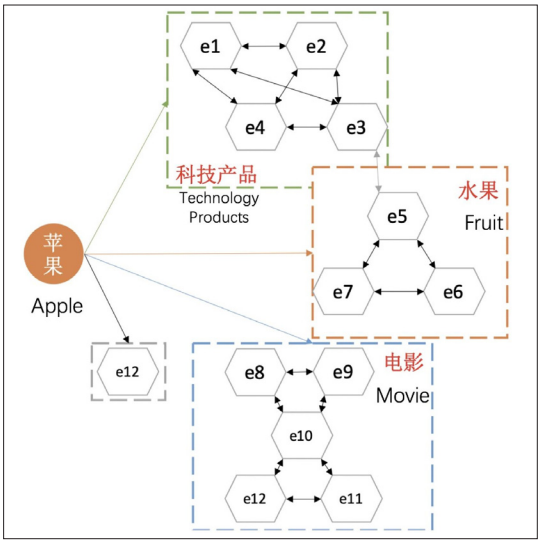

Figure 8. Multiple concepts of an entity

[6] We do not have ground truth of similar entities so we cannot make quantitative analysis

Figure 9. Conceptualized multi-dimension entity recommendation

interrogative query "what food is good for cold weather", our model is able to understand the meaning of query and get the most relative entities "Grain nutrition powder", "Almond milk"; (2) our model is able to handle short queries such as "e52640 and i73770s" which usually do not have the syntax of a written language or contain little signals for statistical inference; (3) our model is able to infer some queries such as "multiply six by the largest single digit greater than fourth" that need commonsense "number" is "mathematical terms" which demonstrate the generalization of our approach; (4) our approach can also handle multi-modal queries "the picture of baby walking feet outside" and get promising results although in recent version of our model we do not consider the image representation in entity recommendation, which indicates that our approach can model the presentation of queries which reveal the implicit need of users. We believe the multi-modal information (images) will further boost the performance which will be left for our future work.

6.7 Conceptualized Entity Recommendation

In the entity recommendation system, each entity may have different views. For example, when recommending entities relative to "apple", it may represent both "fruits" and "technology products" as the Figure 8 shows. Actually, different users have different intentions. To give a better user experience, we develop the conceptualized multi-dimensional recommendation shown in Figure 9. To be specific, we utilize the concepts of candidate entities to cluster the entities in the same group to give a better visual display. Those concepts are retrieved from our cognitive concept graph. Online evaluation shows that conceptualized multi-dimensional recommendation has the total coverage of 49.8% in entity recommendation and also achieve more than 4.1% gain of CTR.

7. Conclusion

In this paper, we study the problem of context modeling for improving entity recommendation. To this end, we develop a deep collaborative match model that learns representations from complex and diverse queries and entities. We evaluate our approach using large-scale, real-world search logs of a widely used commercial search engine. The experiments demonstrate that our approach can significantly improve the performance of entity recommendation.

Generally speaking, the knowledge graph and cognitive concept graph can provide more prior knowledge in query understanding and entity recommendation. In the future, we plan to explore the following directions: (1) we may combine our method with

structure knowledge from knowledge graph and cognitive concept graph; (2) we may combine rule mining and knowledge graph reasoning technologies to enhance the interpretability of entity recommendation; (3) it will be promising to apply our method to other industry applications and further adapt to other NLP scenarios.

Acknowledgments

We would like to thank colleagues of our team - Xiangzhi Wang, YulinWang, Liang Dong, Kangping Yin, Zhenxin Ma, YongjinWang, Qiteng Yang, Wei Shen, Liansheng Sun, Kui Xiong, Weixing Zhang and Feng Gao for useful discussions and supports on this work. We are grateful to our cooperative team - search engineering team. We also thank the anonymous reviewers for their valuable comments and suggestions that help improve the quality of this manuscript.

References

[1] Blanc, Guy., Rendle, Steffen. (2017). Adaptive sampled softmax with kernel based sampling. arXiv preprint arXiv:1712.00527.

[2] Blanco, Roi., Barla Cambazoglu, Berkant., Mika, Peter., Torzec, Nicolas. (2013). Entity recommendations in web search. *In*: *International SemanticWeb Conference*. Springer, 33–48.

[3] Covington, Paul., Adams, Jay., Sargin, Emre. (2016). Deep neural networks for youtube recommendations. *In*: Proceedings of the 10th ACM conference on recommender systems. ACM, 191–198.

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

[5] Fernández-Tobías, Ignacio., Blanco, Roi. (2016). Memory-based recommendations of entities for web search users. *In*: Proceedings of the 25th ACM International on Conference on Information and Knowledge Management. ACM, 35–44.

[6] Grover, Aditya., Leskovec, Jure. (2016). node2vec: Scalable feature learning for networks. *In*: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 855–864.

[7] Huang, Jizhou., Zhang, Wei., Sun, Yaming., Wang, Haifeng., Liu, Ting. (2018). Improving Entity Recommendation with Search Log and Multi-Task Learning. In IJCAI. 4107–4114.

[8] Huang, Jizhou., Zhao, Shiqi., Ding, Shiqiang., Wu, Haiyang., Sun, Mingming., Wang, Haifeng. (2016). *Generating Recommendation Evidence using Translation Model*. In IJCAI. 2810–2816.

[9] Jayaram, Nandish., Gupta, Mahesh., Khan, Arijit., Li, Chengkai., Yan, Xifeng., Elmasri, Ramez. (2014). GQBE: Querying knowledge graphs by example entity tuples. *In*: 2014 IEEE 30th International Conference on Data Engineering. IEEE, 1250–1253.

[10] Joulin, Armand., Grave, Edouard., Bojanowski, Piotr., Douze, Matthijs., Jégou, Hérve., Mikolov, Tomas. (2016). Fasttext. zip: Compressing text classification models. arXiv preprint arXiv:1612.03651.

[11] Metzger, Steffen., Schenkel, Ralf., Sydow, Marcin. (2013). Qbees: query by entity examples. *In*: Proceedings of the 22nd ACM international conference on Information & Knowledge Management. ACM, 1829–1832.

[12] Mikolov, Tomas., Sutskever, Ilya., Chen, Kai., Greg S Corrado, Dean, Jeff. (2013). Distributed representations of words and phrases and their compositionality. *In*: *Advances in Neural Information Processing Systems*. 3111–3119.

[13] Mottin, Davide., Lissandrini, Matteo., Velegrakis, Yannis., Palpanas, Themis. (2014). Exemplar queries: Give me an example of what you need. *In*: Proceedings of the VLDB Endowment 7, 5 (2014), 365–376.

[14] Wang, Hongwei., Zhang, Fuzheng., Xie, Xing., Guo, Minyi. (2018). DKN: Deep knowledge-aware network for news recommendation. *In*: Proceedings of the 2018 World WideWeb Conference. *International World Wide Web Conferences Steering Committee*, 1835–1844.

[15] Yang, Zhilin., Dai, Zihang., Yang, Yiming., Carbonell, Jaime., Salakhutdinov, Ruslan., Quoc V Le. (2019). XLNet: Generalized Autoregressive Pretraining for Language Understanding. arXiv preprint arXiv:1906.08237.

[16] Zhang, Ningyu., Deng, Shumin., Sun, Zhanlin., Chen, Xi., Zhang, Wei., Chen, Huajun. (2018). Attention-based capsule networks with dynamic routing for relation extraction. arXiv preprint arXiv:1812.11321.

[17] Zhang, Ningyu., Deng, Shumin., Sun, Zhanlin., Wang, Guanying., Chen, Xi., Zhang, Wei., Chen, Huajun. (2019). Long-tail Relation Extraction via Knowledge Graph Embeddings and Graph Convolution Networks. arXiv preprint arXiv:1903.01306.

[18] Zhu, Han., Li, Xiang., Zhang, Pengye., Li, Guozheng., He, Jie., Li, Han., Gai. (2018). Learning Tree-based Deep Model for Recommender Systems. *In*: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. ACM, 1079–1088.