Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)

Efficiently Answering Technical Questions — A Knowledge Graph Approach

Shuo Yang,[§*] Lei Zou,[§†] Zhongyuan Wang,[‡†] Jun Yan,[‡] Ji-Rong Wen[#]
[§] Peking University [†] Beijing Institute of Big Data Research
[‡]Microsoft Research Asia
[#]Renmin University of China & Beijing Key Laboratory of Big Data Management and Analysis Methods
{yangshuo1991,zoulei}@pku.edu.cn; wzhy@outlook.com; junyan@microsoft.com; jrwen@ruc.edu.cn

Abstract

More and more users prefer to ask their technical questions online. For machines, understanding a question is nontrivial. Current approaches lack explicit background knowledge. In this paper, we introduce a novel technical question understanding approach to recommending probable solutions to users. First, a knowledge graph is constructed which contains abundant technical information, and an augmented knowledge graph is built on the basis of the knowledge graph, to link the knowledge graph and documents. Then we develop a light weight question driven mechanism to select candidate documents. To improve the online performance, we propose an index-based random walk to support the online search. We use comprehensive experiments to evaluate the effectiveness of our approach on a large scale of real-world query logs. Our system outperforms main-stream search engine and the state-of-art information retrieval methods. Meanwhile, extensive experiments confirm the efficiency of our index-based online search mechanism.

Introduction

An emerging trend is the growth need to ask questions online. It has become a popular part of Web search services on main-stream search engines. Some promising intelligent assistants on mobile devices, such as Siri and Cortana, provide similar question answering services. In this paper, we focus on answering "technical questions". Given a large technical corpus (such as HELP documents) and a natural language question, our goal is to return some relevant documents with regard to the technical question. For example, users usually confront some technical problems when using Microsoft products, such as "how to uninstall office" and "how can I check the edition and version of the SQL server". They post their questions on search engines or technical forums. Then, search engines or experts will recommend relevant technical documents to them. However, results from search engines usually turn out to have low accuracy and experts cannot reply immediately. Taking these circumstances into consideration, if we can improve the precision of search results on the search engines, it will provide convenience not only for users but also for companies that provide technical supports. However, understanding technical questions is not trivial for machines. There are several inherent challenges posed:

1. ***Lack of background knowledge***
   It is difficult for machines to understand a question for which they lack the technical knowledge required. For example, if we don't know that the *"outbox"* is the place that stores emails users have sent, when a question "sent e-mails disappear when using Outlook" comes, machines cannot recognize it as an error about *outbox* in Outlook.
2. ***Hard to understand query intention***
   We cannot understand the query intention according to surface meaning of the words. For example, question "unable to view security event logs" asks for the "access control" rules of the security event logs. i.e. "view" means "access" or "permission" rather than "see".

Traditional information retrieval methods only focus on keywords appearing in a question, which cannot really understand user query intentions. To understand a question, humans usually resort to their knowledge in the technical area. For example, when one tries to understand the question "some specific custom forms get stuck in outbox when users send it", she will first focuses on the "custom form", "outbox" and words describing phenomena like "stuck" and "send". She also knows that "custom form" and "outbox" are components of "Outlook", a mailbox product. Then she deduces that the question is about an error of *Outlook*.

As shown by the analysis above, the background knowledge plays a critical role in understanding questions for humans. Intuitively, we need a background knowledge base for machines to understand user technical questions. Many existing knowledge bases are about lexical knowledge or about open domain facts. A typical example of the former is WordNet (Miller 1995), a lexical knowledge base for the English language. The latter examples focus on general knowledge (Auer et al. 2007; Suchanek, Kasneci, and Weikum 2007; Wu et al. 2012). In this paper, we build a large scale technique knowledge base covering the full series of Microsoft products. To improve the precision of answering technical questions, we propose a novel two-step framework—*candidates selection and re-ranking*. In the first step, a light weight question-driven method is applied to candidate doc-

[*]This work was partially done when Shuo visited MSRA.
[†]Zhongyuan is currently with Facebook Inc. at Menlo Park, USA. His email is zhy@fb.com.

uments selection. The first step has a good performance in time and the results of it is low in precision but high in recall. Therefore, we can apply it to select the candidates and re-rank the top-k results in the next step. In the second step, we re-rank the candidate documents based on the technique knowledge graph that is built offline. Due to the large size of the knowledge graph, we propose a novel index that speeds up online query processing. Generally speaking, our method outperforms the state-of-the-arts by 5%-15% in accuracy and also achieves the real time query response times in answering technical questions.

Hierarchical Technical Knowledge Graph

As mentioned earlier, none of existing knowledge graph are designed for technical question answering. Therefore, we study how to build a technical knowledge graph by making use of technical corpus.

Graph Definition

A technical question usually consists of 3 parts: product, component and event words, even though some are not explicitly stated (In fact, more than 90% questions satisfy our assumption). Generally speaking, it has four levels: category, product, component and event. Each node of the knowledge graph belongs to one level.

1. ***Category Level:*** A node in this level denotes a category, which is a group of some products sharing the same or similar functions. A category also can be a sub-category of another one.
2. ***Product Level:*** This level contains all products and attributes of products, which are essential in our technical knowledge graph. Each node in this level represents a product or an attribute value. We pre-define three attributes with regard to products: version, language and environment.
3. ***Component Level:*** More specifically, an error usually belongs to a component of a product. Thus, a node in this level corresponds to a component.
4. ***Event Level:*** When products and components are identified, the essence of answering users' questions is to understand "error phenomena".

A sample of a knowledge graph is shown in Figure 1.

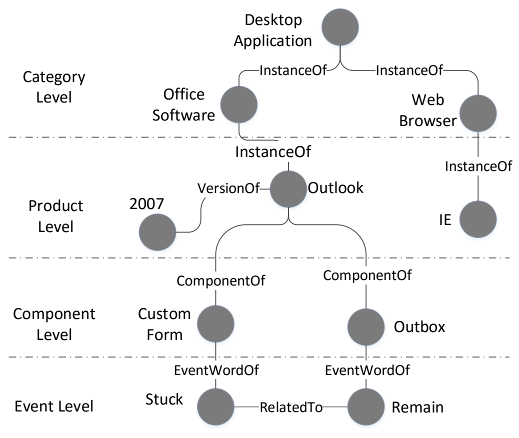


Figure 1: A sample of knowledge graph

Knowledge Graph Building

We are to build a knowledge graph based on the technical corpus. We describe the detailed approaches in each level.

Category & Product Level

We extract categories and products from product information. There are 6052 products and they are grouped into 214 categories in our knowledge graph. We also identify the attributes of each product based on rules. For example, "Office Pro Win32 IT" represents this product is "office" and its version is "Professional", language is "Italian" and it is installed on the "32bit Operating System".

Component Level

We propose to build component levels automatically based on the technical corpus and users query logs. First, we extract some frequent technical noun phrases by using the sequence labelling method (Lafferty, McCallum, and Pereira 2001), which are regarded as the components in technical domain. These extracted noun phrases are added to the set of component nodes in this level. Then we extract "ComponentOf" relations based on co-occurrence of components and products in the technical corpus. Specifically, considering a component c and a product p. Pointwise Mutual Information (PMI) (Manning and Schütze 1999) is a common measure of the strength of association between two terms, we define that c is a component of p if $PMI(c,p)$ is larger than a threshold α, where PMI is define as:

$$PMI(c,p) = log\frac{P(c,p)}{P(c)P(p)} \tag{1}$$

where $P(c)$, $P(p)$ and $P(c,p)$ are calculated by:

$$P(c) = \frac{\#(c)}{\#(sen)} \quad P(p) = \frac{\#(p)}{\#(sen)} \quad P(c,p) = \frac{\#(c,p)}{\#(sen)} \tag{2}$$

$\#(c)$ ($\#(p)$) represents the number of sentences containing component words c (product words p), while $\#(c,p)$ represents the number of sentences containing both c and p. $\#(sen)$ denotes all sentences in the corpus.

Event Level There are two kinds of edges in this level: "EventWordOf" and "RelatedTo". We discuss them separately. We adopt a similar solution with extracting "ComponentOf" to find "EventWordOf" relations between products (or components) with event words. Users often use verbs, adjectives and nouns to describe error phenomena. Given a large technical corpus, we expect to exact the relations between components (or products) and events from these sentences. Then all verbs, adjectives and nouns in these sentences are identified as candidate event words. For example, "cannot forward email in Outlook" contains the product "Outlook" and verbs "forward" and noun "email", where "forward" and "email" are regarded as event words. We also use PMI to extract event words with regard to some component or product. We only keep the event words if their PMI values are larger than a threshold. Secondly, we want to connect two event words if they share similar semantic. We assume that two questions are semantically similar if they can

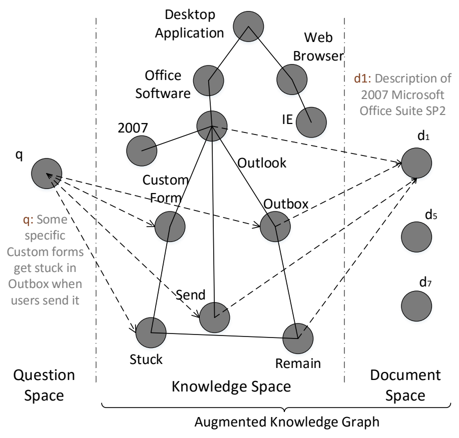

Desktop Application
Web Browser
Office Software
d1: Description of 2007 Microsoft Office Suite SP2
IE
2007
d1
q
Outlook
Custom Form
Outbox
d5
q: Some specific Custom forms get stuck in Outbox when users send it
d7
Send
Stuck
Remain
Question Space
Knowledge Space
Document Space
Augmented Knowledge Graph

Figure 2: Augmented knowledge graph

be solved by the same document. Because experts usually answer the technical questions in the forum, a document can be regarded as a solution of a question only when its URL appear in the same page of the question. These question-document pairs are high in precision as they are manually labeled. For example, the technical document d_0 can solve three questions $\{q_2, q_9, q_{15}\}$, where

I. q_2: Outlook 2007 gets **frozen**.

II. q_9: Outlook sending status **remains** for hours.

III. q_{15}: Emails get **stuck** in outbox.

So it is assumed that event words “frozen”, “remain” and “stuck” are similar in semantics, and “RelatedTo” edges are added between each other. The key advantage of this approach is addressing the “data-sparsity” issue. The word “frozen” is infrequent in the data set, so when it appears in another question, it’s probable that the target document cannot be discovered. When these edges are added, “frozen” will be transferred to another word like “remain” or “stuck”.

Augmented knowledge graph

Since our goal is to return some relevant technical documents with regard to user questions, we need to link the knowledge graph that is constructed above with the technical corpus (Figure 2). We call the link result as “Augmented knowledge graph” (“A-knowledge graph”). The relations in the knowledge graph nodes and the corresponding documents are built based on user question logs. We collect questions from the technical forum. In the question logs, each document solves some of questions. Specifically, for document d, we denote $QL(d)$ as a list of questions that can be answered with the document d.

Edge Weight. In the knowledge space, for edges linking two category nodes or a category node and a product node, we define the edge weight as the same. And for the other edges, and the weight of edge (x, y) is defined as the conditional probability of y that given x, i.e.

$$w(x, y) = P(y|x) = \frac{\#(x, y)}{\#(x)} \tag{3}$$

where $\#(x, y)$ denotes the number of sentences containing both node x and y. To measure the weight of edges between knowledge graph nodes and document nodes, we define the $w(x, d)$ as the appearance probability of node x:

$$w(x, d) = P(x|d) = \frac{|q \in QL(d) \wedge x \in q|}{|QL(d)|} \tag{4}$$

where the numerator is the number of questions that can be solved by d and contain the word x. $|QL(d)|$ is the number of questions can be solved by d.

Candidate Answer Selection

As shown in Figure 2, our mechanism aims at linking a question to the augmented knowledge graph, and then calculates the relevance of the question and all the document nodes. However, the number of nodes in the document space is so large that re-ranking all of them is time-consuming. So we need a lightweight approach to selecting candidate documents for a user’ question, to support the online search.

Traditional document retrieval methods only consider similarities between questions and documents (Celikyilmaz, Hakkani-Tur, and Tur 2010; Jeon, Croft, and Lee 2005b), which are called the document-driven approach. However, user questions and the corresponding technical documents may use different wording. Therefore, in this sec tion, we propose to employ a question-driven solution. Specifically, we first find the similar questions in the log with regard to a user issued question. Based on the question similarities, we deduce the relevance between a user question with regard to the documents. Assume there is a question log C that is a collection of question and document pairs. As shown in Figure 3, document d_5 solves question $q004$, thus, there is a pair $(q004, d_5)$ in the log C. For a user question q, we find the most similar m questions from C based on the similarity $score(q, q_i)$, where $score(q, q_i)$ is calculated by a traditional information retrieval method. These selected questions q_i together with their corresponding technical documents are denoted as C_0. Let $C_0 = \{(q'_0, d'_0), \cdots, (q'_m, d'_m)\}$. We assume that (q'_i, d'_i) are ranked according to $score(q, q'_i)$ in C_0. We compute the relevance between the user’s question q with document d as follows:

$$score(q, d) = log\#(d, C) \times \sum_{(q'_i, d) \in C_0} \frac{\#(d, C_0)}{\#(d, C) \times i} \times score(q'_i, q) \tag{5}$$

where $\#(d, C)$ denotes the number of questions in the question logs set C that can be solved by d and $\#(d, C_0)$ is analogue to $\#(d, C)$. $(q'_i, d) \in C_0$ means that d can solve the question q'_i in C_0 and $score(q, q'_i)$ is ranked at i-th in C_0. Since d may solve multiple questions in C_0, we need to consider all of them. Meanwhile, we combine the score of the document content and the history question log via linear weighting, in case a new document has no logs.

Extensive experiments (in experiment part) show that the first step have a good recall in the top-100 answers; but the most relevant documents are usually out of the top-10 answers for users’ questions. In other words, we need a sophisticated approach to re-rank the candidates and provide more precise results on the document ranking. That is based on the technical knowledge graph.

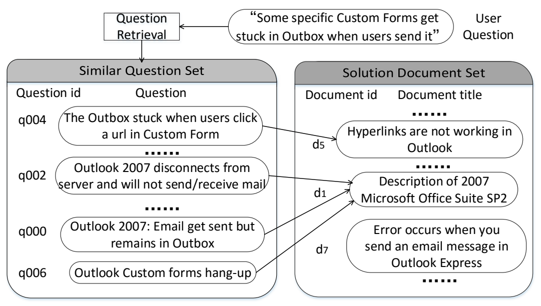

Question Retrieval
"Some specific Custom Forms get stuck in Outbox when users send it"
User Question
Similar Question Set
Question id
Question
q004
The Outbox stuck when users click a url in Custom Form
q002
Outlook 2007 disconnects from server and will not send/receive mail
q000
Outlook 2007: Email get sent but remains in Outbox
q006
Outlook Custom forms hang-up
Solution Document Set
Document id
Document title
d5
Hyperlinks are not working in Outlook
d1
Description of 2007 Microsoft Office Suite SP2
d7
Error occurs when you send an email message in Outlook Express

Figure 3: Select candidates by a question-driven method

Candidates Re-ranking

Given a user's question, we have obtained the top-k candidate documents. In this section, we propose a knowledge graph-based approach to re-rank these candidates.

Candidate Documents Re-ranking

In previous sections, a light weight approach is proposed to select candidates from a large scale of technical documents. In this section, we link user question to the A-knowledge graph. Then a random walk based approach is applied to calculate the similarity between user's online question and help documents.

Let us recall the augmented knowledge graph (A-knowledge graph for short) in Figure 2. Given a user question, we link it to the corresponding nodes in the A-knowledge graph.

As we know, *random walk* (Pearson 1905) is an effective measure to evaluate the similarity between two nodes in a graph. Generally speaking, if we start at a question node and walk to another randomly, the probability we reach each document node is regraded as the similarity between the question and the corresponding document node. It's defined as:

$$s(x,y) = \sum\nolimits_{x' \in N(x)} \left(T(x,x') \times s(x',y)\right) \qquad (6)$$

where $s(x,y)$ is the random walk-based similarity between nodes x and y, $N(x)$ denotes all out neighbors of x and $T(x,x')$ is the transfer probability from node x to y. Here, we define the $T(x,x')$ by normalizing the edge weights.

In computing random walk-based similarity, we only have the edges from knowledge graph nodes to document nodes, but the reverse direction is not allowed. This is the same for edges from query nodes to knowledge graph nodes.

Index-based Online Search

The random walk similarity between query node q and document d can be computed by two approaches. The first approach is the sampling-based solution. We start at the query node q and walk to another randomly according to the transfer probability. Assume that we perform N random walks starting from query node q and there are r times that the random walk ends at document d. Then, the similarity

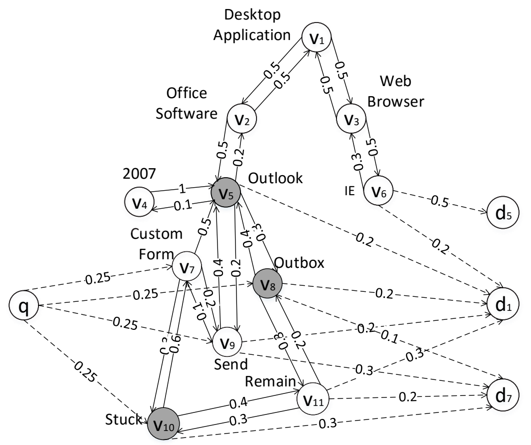


Figure 4: An example of index-based random walk

$s(q,d)$ is estimated as $\frac{r}{N}$. We found that the estimated value becomes convergent when N is larger than 4 million. More details are found in our experiment section (Figure 6(a)). It means that it is time-consuming to employ the sampling-based solution in online phase, since we need the real-time response for user questions. The second approach is that we build a linear equation group based on the random walk definition (see Equation 6). Let us recall Figure 2. In order to compute $s(q,d_1)$, we have the following linear equation group.

$$\begin{cases} s(q,d_1) &= 0.25 \times s(v_7,d_1) + 0.25 \times s(v_8,d_1) + \\ & \quad 0.25 \times s(v_9,d_1) + 0.25 \times s(v_{10},d_1) \\ s(v_1,d_1) &= 0.5 \times s(v_2,d_1) + 0.5 \times s(v_3,d_1) \\ s(v_2,d_1) &= 0.5 \times s(v_1,d_1) + 0.5 \times s(v_5,d_1) \\ s(v_3,d_1) &= 0.5 \times s(v_1,d_1) + 0.5 \times s(v_6,d_1) \\ s(v_4,d_1) &= s(v_5,d_1) \\ s(v_5,d_1) &= 0.2 \times s(v_1,d_1) + 0.1 \times s(v_4,d_1) + \\ & \quad 0.3 \times s(v_8,d_1) + 0.2 \times s(v_9,d_1) + 0.1 \\ s(v_6,d_1) &= 0.3 \times s(v_3,d_1) + 0.2 \\ s(v_7,d_1) &= 0.5 \times s(v_5,d_1) + 0.2 \times s(v_9,d_1) + 0.3 \times s(v_{10},d_1) \\ s(v_8,d_1) &= 0.4 \times s(v_5,d_1) + 0.3 \times s(v_{11},d_1) + 0.2 \\ s(v_9,d_1) &= 0.4 \times s(v_5,d_1) + 0.1 \times s(v_7,d_1) + 0.2 \\ s(v_{10},d_1) &= 0.6 \times s(v_7,d_1) + 0.4 \times s(v_{11},d_1) \\ s(v_{11},d_1) &= 0.3 \times s(v_{10},d_1) + 0.2 \times s(v_8,d_1) + 0.3 \end{cases} \qquad (7)$$

However, the time complexity for solving the linear equation group is $O(n^3)$ via Gaussian elimination method (n is the number of unknowns). Therewith, the prohibitive computing cost forbids us using the approach in online processing. Obviously, the fast online computing $s(q,d)$ is to materialize the similarities between all knowledge graph nodes v_i and all document nodes d_j. The straightforward solution leads to the expensive space cost. To provide a good trade-off between the space cost and online response time, we pre-calculate the similarity between some selected knowledge graph nodes and document nodes. For example, we select three nodes (v_5, v_8, v_{10}) as the materialized nodes. Specifically, we record the similarities between v_5, v_8 and v_{10} with all document nodes at offline phase. Given the same question q, the linear equation group is shrunk as follows (since $s(v_5,d_1) = 0.701, s(v_8,d_1) = 0.668, s(v_{10},d_1) = 0.642$):

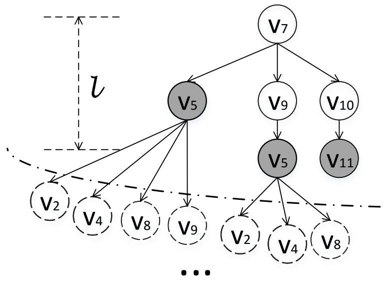


Figure 5: Path tree rooted at v_7

$$\begin{cases} s(q,d_1) &= 0.25\times s(v_4,d_1)+0.25\times s(v_7,d_1)+0.25\times \\ & \quad 0.668+0.25\times s(v_9,d_1) \\ s(v_7,d_1) &= 0.5\times 0.701+0.2\times s(v_9,d_1)+0.3\times 0.642 \\ s(v_9,d_1) &= 0.4\times 0.701+0.1\times s(v_7,d_1)+0.2 \end{cases}$$

The number of unknows in Equation 8 is 3 (11 if no materialized node), which leads to significant query performance improvement. The format of the index of a materialized node x is a list of doubles, represented as: $Index(x) = s(x,d_0), s(x,d_1), \cdots, s(x,d_m)$, where m is the number of documents.

From the analysis described above, our task is to build indices on a set of materialized nodes X in the knowledge graph G. The remaining set is $Y = V(G)\backslash X$, where $V(G)$ denotes all knowledge graph nodes. Given a set of materialized nodes X, for each node $y \in Y$, for any path starting from y with length no shorter than l, we ensure that the path must contain at least one materialized node $x \in X$. Figure 5 illustrates the path tree rooted at vertex v_7. The subtree rooted at v_7 is surrounded by some materialized nodes (illustrated in black ones). In order to compute $s(v_7, d_i)$, where d_i is a document node, the number of unknowns in the equation group equals to the number of (white) nodes surrounded by some materialized nodes (black noes) in Figure 5. Obviously, the number of unknowns (white nodes) is related to the controlled path length l. Therefore, in order to select materialized nodes, we propose the following LPCP problem in Definition 0.1. Unfortunately, this is a NP-hard problem.

Definition 0.1 (LPCP: l-length path cover problem)**.** *For a given controlled path length l, the LPCP problem is to select a set of materialized nodes $X \subseteq V(G)$, for each simple path p that is no less than l, there is at least one node x in p, and $x \in X$.*

Theorem 0.1. *For a LPCP, choosing the minimum number of nodes to build index is a NP-hard problem.*

Proof. The set covering problem is known to be a NP-hard problem. For a given set U which contains all universe elements, and a set of n sets S, each element in S is a set which is composed by elements in U. We can replace each element in U with a path in the set of all l-length path, and each s_i can be regarded as the paths that pass the i-th node in G. And the set covering problem can be reduced to the LPCP in polynomial time. Since the set covering problem is NP-hard, choosing the minimum number of nodes for LPCP is also a NP-hard problem. □

In this paper, we propose a greedy approach to choose the materialized nodes set X. Our greedy algorithm selects *popular* nodes, where popular nodes are the nodes that are more likely to be appear in l-length paths. When the popular nodes are selected, more paths in G will be covered. In our algorithm, the node with largest (in-degree$\times$out-degree) are considered as *popular nodes*. We generate the materialized nodes set X iteratively. In each iteration, the node v with the maximum popularity is selected, then v is added to X and deleted from Y. Then popularity of each node in Y is recalculated. These operations proceed repeatedly until there is no simple path longer than l.

Experiment

We conduct comprehensive experiments on real-world dataset to evaluate the effectiveness and efficiency of our approach to answering technical questions.

Benchmark

We aim to provide natural language question-based technical services for the full series of Microsoft products, which covers 214 categories and 6052 products.

Dataset. We collected 1,176,328 uses' question together with 111,062 different help documents that are clawed from the technical forum[1]. We consider a document as a solution of a question if someone answers the question with the document. To estimate the effectiveness of our approach, we use two different sources of testing data.

Evaluation Sets. The first dataset is user questions that are clawed from the technical forum. We randomly select 60,000 questions in EVAL1. A document is considered as a solution of a question if it appears in the same page with the question in the technical forum. Specifically, we randomly select 100 user questions and ask domain experts to assign several technical documents for each question (EVAL2). We use this dataset because question logs cannot find all the answers of a question.

Effectiveness of Knowledge Graph

In this section, we evaluate the effectiveness of our constructed technical knowledge graph. In the knowledge graph, there are about 250,000 nodes that are classified into four levels: 214 categories, 6052 products, 45085 components and the rest are event words nodes. There are about 11 million relations (edges) between these nodes in the knowledge graph. We perform a case study to demonstrate the effectiveness of the knowledge graph in Table 1. ("EWO": "EventWordOf", "CPO": "ComponentOf", "PDE": "Product Description Edge", "ITO":"InstanceOf")

Effectiveness of Candidates Selection

As mentioned above, we propose a question-driven solution for candidate document selection. In this experiment, we compare our solution with the document-driven solution. For the document-driven solution, we adopt (Fox and Shaw 1994), which combines some classical information retrieval

[1]http://answers.microsoft.com/en-us/windows/forum

Table 1: Frequent Relations in Knowledge Graph

Node	Level	Neighbors and Adjcent Edges (ordered by edge weight)									
		1		2		3		4		5	
Windows	Product	error	EWO	upgrade	EWO	32bit	EO	7	VO	Desktop	CPO
Internet Explorer	Product	website	EWO	video	EWO	cookie	CPO	use	EWO	uninstall	EWO
Windows Account	Component	Windows	CPO	verify	RT	sign	RT	fail	RT	logon	EWO
Spell Checker	Component	work	EWO	office	CPO	language	RT	English	RT	Office	CPO
license	Event Word	buy	RT	subscription	RT	error	RT	Office	EWO	add	RT
frozen	Event Word	service	EWO	stop	RT	slow	RT	disconnect	RT	respond	RT

Table 2: Candidates Selection Methods on EVAL1

method	MRR	MAP	A@10	A@50	A@100
doc. title	0.09	0.07	0.15	0.22	0.27
doc. content	0.10	0.08	0.15	0.23	0.28
doc. title + content	0.12	0.10	0.18	0.27	0.32
STEP-1	**0.22**	**0.19**	**0.39**	**0.64**	**0.87**

Table 3: Result for candidates re-ranking

Method	EVAL1			EVAL2		
	A@1	A@3	A@5	A@1	A@3	A@5
TSS	0.07	0.11	0.13	0.21	0.28	0.34
Google	0.11	0.13	0.16	0.22	0.33	0.38
STEP-1	0.13	0.23	0.31	0.21	0.36	0.44
Trans	0.10	0.21	0.30	0.25	0.42	0.52
TransLM	0.12	0.26	0.36	0.27	0.44	0.55
LETOR(1:3)	0.21	0.34	0.42	0.29	0.51	0.57
LETOR(1:5)	0.19	0.32	0.38	0.27	0.47	0.52
LETOR(1:10)	0.19	0.31	0.34	0.27	0.42	0.50
Our Method	**0.22**	**0.38**	**0.46**	**0.32**	**0.55**	**0.61**

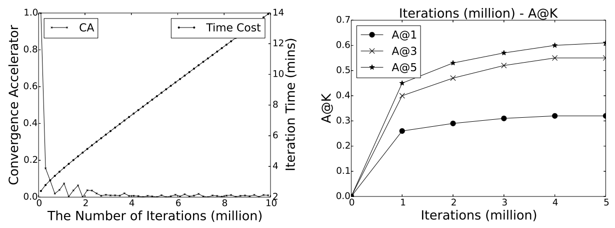
approaches, such as TF-IDF, language model, BM25, DFR (Amati and Van Rijsbergen 2002) and information-based model (Clinchant and Gaussier 2010). We test three different features in the document-driven solutions, that are title, document content and document title+content. Since a question may be solved by several technical documents, if one of them is retrieved, this question is considered to be well solved in our evaluation. We define the accuracy at top-k as the proportion of questions with at least one correct answer at top-k results, as described in Equation 8.

$$A@k = \frac{\#_k(solved\ questions)}{\#(questions)} \tag{8}$$

where $\#_k(solved\ questions)$ is the number of questions that get at least one solution document at the top k retrieval results and $\#(question)$ is the number of questions in the evaluation set. The goal of the candidates selection is to pick out the solution documents as much as possible, therefore we compute the MRR(Mean Reciprocal Rank), MAP(Mean Average Precision), accuracy at top 10, 50 and 100. And the results in Table 2 show that our query-driven solution outperforms the document-driven methods significantly. For example, our method finds solutions for 87% questions at top 100 results.

Effectiveness of Candidates Re-ranking

Although the question driven solution achieves 87% accuracy in top-100 answers, the top-1, 3, 5 answers are not good enough for answering technical questions. Their accuracies are about 13%–44%(see "STEP-1"), as reported in Table 3. Therefore, a re-ranking step is desirable. In the candidate re-ranking step, we compare our method with a tranlation-based model (Trans) (Jeon, Croft, and Lee 2005a) and translation language model (TransLM) (Xue, Jeon, and Croft 2008), which discuss the details of answering frequent asked question by making using of a large scale of question logs, and learning to rank (LETOR) approach (Liu 2009). For LETOR, we extract features of a question and two documents, that is, we use pairwise LETOR. Some pre-



(a) #(Iterations) - Performance (b) #(Iterations) - Accuracy

Figure 6: Performance of Random Walk Re-ranking

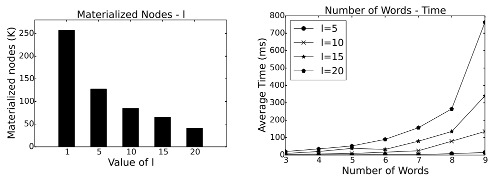
defined features provided by the offical website[2], and some other similarity features like (Clinchant and Gaussier 2010; Amati and Van Rijsbergen 2002; Fox and Shaw 1994) (features collected from both documents and similar questions) are also added. We choose negative samples randomly from the question set, the ratio of the number of positive and negative samples is varied from 1:3, 1:5 and 1:10. And the LETOR is implemented by SVM-Rank[3], the radial basis function is used as the kernel function and the parameter c is set to 5.0. Furthermore, we also including the comparison with Google and the technical support service of Microsoft (TSS). The answer are limited to the sites where reside technical documents. We report the accuracy of each solution in top-1, 3 and 5 in Table 3. LETOR has a better performance than other methods since it combines different features. And the TransLM has an improvement than Trans. Our algorithm shows better performance than other approaches because our method expands a question via a well-constructed knowledge graph, which discovers important words for a specific error phenomenon. We also list some running examples of top-1 results (Table 4).

[2]http://research.microsoft.com/en-us/projects/mslr/

[3]https://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html

Table 4: Some question-document samples

Question	Document title
Vista can't upgrade to Windows 7	Upgrading from Windows Vista to Windows 7 - Windows Help
Word 2010 Spell Checker not working on a specific document	Spell Checker does not recognize misspelled words in Word 2010
Can't right click, or open links in IE	Nothing happens when you click a link in Internet Explorer
Lost a bunch of e-mail contacts, how do I get them from backup drive?	How to manage .pst files in Microsoft Outlook
After installing IE10, performance went down and some webpages won't open	"Internet Explorer cannot display the webpage" error
How to set a desktop background on windows 8	Personalize your PC - Windows tutorial
I lost my recovery file for a document	How to recover a lost file in Word 2007 or in Word 2003
Loses email accounts for Outlook 2010	How to create profile and set up an e-mail account in Outlook



(a) materialized nodes v.s. l (b) The response time v.s. words in question

Figure 7: Time Performance of Index-Based Random Walk

Efficiency of Index-based Random Walk

To enable short query response time, we propose an index-based random walk solution. Convergence accelerator (CA) of iterations and the time cost of random walk are recorded in Figure 6(a) and CA reflects the global convergence rate of the random walk, which is defined in Equation 9 where K and D represents the knowledge graph and document nodes set, CA_n is the CA in the n-th iteration, and $s_n(x,d)$ is the similarity between x and d in the n-th iteration. We need 4 million iterations before the similarity value converges, which costs about 7 minutes. Obviously, we cannot use this approach in online answering technical questions. And as shown in Figure 6(b), as the number of iterations increases, A@1, A@3 and A@5 also have an improvement.

$$CA_n = \frac{\sum_{x \in K} \sum_{d \in D} |s_n(x,d) - s_{n-1}(x,d)|}{\sum_{x \in K} \sum_{d \in D} s_n(x,d)} \quad (9)$$

As shown in Figure 7(a), the larger the l is, the less materialized nodes we should select to build index. When l is equal to 20, we need to select about 40,000 materialized nodes. The time performance of the index-based random walk is also shown in Figure 7(b), which costs about 100-800 milliseconds.

The demo of our system can be visited via http://59.108.48.29/Search.

Related Work

A knowledge graph is an effective way to represent the knowledge of the objective world. There are many existing knowledge graphs like (Auer et al. 2007; Wu et al. 2012; Yao and Van Durme 2014; Lee et al. 2011). These knowledge graphs are open-domain and none of them aimed at the technical area. Information Retrieval (IR) approaches find similar documents from a large scale of corpus, Clinchant (Clinchant and Gaussier 2010) proposed a information-based model. Some information retrieval frameworks (Liu et al. 2007) (Fox and Shaw 1994) calculated the final ranks by combining several scores provided by other models. Query Expansion enriches the information of a question, Swapna (Gottipati and Jiang 2011) expanded a query using knowledge base, there is also work (Collins-Thompson and Callan 2005) archive QE using random wolk, Craswell (Craswell and Szummer 2007) retrieves related documents via click graph. Xu (Xu and Croft 1996) mined similar words from relative documents, and there are also some approaches (Gao and Nie 2012; Gupta et al. 2014) enrich a query via search logs. Those IR-based approaches find similar documents for a specific quesion by measuring the relavance of them direcly. Ji (Ji et al. 2012) developed a topic model in Community Question Answering, Zhou (Zhou et al. 2015) proposed an embedding-based question retrieval method in community.

Conclusion

Question understanding is an important and challenging task to a large variety of QA systems. In this paper, we propose a novel question understanding method which simulates the way humans think. We construct a hierarchical knowledge graph automatically which contains abundant technical information, to understand a question. Next, an augmented knowledge graph is used for relating the questions to documents via the knowledge graph. Then we use a question-driven method to select candidate solutions. Finally, random walk approach is applied to candidates ranking and an index-based random walk is proposed to support the online search.

Our two-step pipeline guarantees both precision and efficiency, which is also suitable for other similar tasks. The experiments provide a comprehensive evaluation from different respects, experimental data is obtained from 1.2 million real-world question logs of 111 thousand help documents. Technical knowledge graph we construct covers full series products of Microsoft and is of high quality. Our re-ranking approach outperforms main-stream search engine and other state-of-art methods. And the index-based random walk approach speeds up the online search to a large extent. In the future, we are to further evaluate our method on other specific retrieval systems such as news, research papers recommendation.

Acknowledgements

This work was supported by The National Key Research and Development Program of China under grant 2016YFB1000603,973 Program of China under grant No. 2014CB340403, and NSFC under grant 61622201, 61532010 and 61370055. Lei Zou is the corresponding author of this paper.

References

Amati, G., and Van Rijsbergen, C. J. 2002. Probabilistic models of information retrieval based on measuring the divergence from randomness. *ACM Transactions on Information Systems (TOIS)* 20(4):357–389.

Auer, S.; Bizer, C.; Kobilarov, G.; Lehmann, J.; Cyganiak, R.; and Ives, Z. 2007. Dbpedia: A nucleus for a web of open data. In *The semantic web*. Springer. 722–735.

Celikyilmaz, A.; Hakkani-Tur, D.; and Tur, G. 2010. Lda based similarity modeling for question answering. In *Proceedings of the NAACL HLT 2010 Workshop on Semantic Search*, 1–9. Association for Computational Linguistics.

Clinchant, S., and Gaussier, E. 2010. Information-based models for ad hoc ir. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, 234–241. ACM.

Collins-Thompson, K., and Callan, J. 2005. Query expansion using random walk models. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, 704–711. ACM.

Craswell, N., and Szummer, M. 2007. Random walks on the click graph. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, 239–246. ACM.

Fox, E. A., and Shaw, J. A. 1994. Combination of multiple searches. *NIST SPECIAL PUBLICATION SP* 243–243.

Gao, J., and Nie, J.-Y. 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*, 1. ACM.

Gottipati, S., and Jiang, J. 2011. Linking entities to a knowledge base with query expansion. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 804–813. Association for Computational Linguistics.

Gupta, P.; Bali, K.; Banchs, R. E.; Choudhury, M.; and Rosso, P. 2014. Query expansion for mixed-script information retrieval. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, 677–686. ACM.

Jeon, J.; Croft, W. B.; and Lee, J. H. 2005a. Finding semantically similar questions based on their answers. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, 617–618. ACM.

Jeon, J.; Croft, W. B.; and Lee, J. H. 2005b. Finding similar questions in large question and answer archives. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, 84–90. ACM.

Ji, Z.; Xu, F.; Wang, B.; and He, B. 2012. Question-answer topic model for question retrieval in community question answering. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, 2471–2474. ACM.

Lafferty, J.; McCallum, A.; and Pereira, F. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the eighteenth international conference on machine learning, ICML*, volume 1, 282–289.

Lee, T.; Wang, Z.; Wang, H.; and Hwang, S.-w. 2011. Web scale taxonomy cleansing. *Proceedings of the VLDB Endowment* 4(12).

Liu, T.-Y.; Xu, J.; Qin, T.; Xiong, W.; and Li, H. 2007. Letor: Benchmark dataset for research on learning to rank for information retrieval. In *Proceedings of SIGIR 2007 workshop on learning to rank for information retrieval*, 3–10.

Liu, T.-Y. 2009. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval* 3(3):225–331.

Manning, C. D., and Schütze, H. 1999. *Foundations of statistical natural language processing*, volume 999. MIT Press.

Miller, G. A. 1995. Wordnet: a lexical database for english. *Communications of the ACM* 38(11):39–41.

Pearson, K. 1905. The problem of the random walk. *Nature* 72(1865):294.

Suchanek, F. M.; Kasneci, G.; and Weikum, G. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, 697–706. ACM.

Wu, W.; Li, H.; Wang, H.; and Zhu, K. Q. 2012. Probase: a probabilistic taxonomy for text understanding. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, 481–492. ACM.

Xu, J., and Croft, W. B. 1996. Query expansion using local and global document analysis. In *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, 4–11. ACM.

Xue, X.; Jeon, J.; and Croft, W. B. 2008. Retrieval models for question and answer archives. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, 475–482. ACM.

Yao, X., and Van Durme, B. 2014. Information extraction over structured data: Question answering with freebase. In *ACL (1)*, 956–966. Citeseer.

Zhou, G.; He, T.; Zhao, J.; and Hu, P. 2015. Learning continuous word embedding with metadata for question retrieval in community question answering. In *Proceedings of ACL*, 250–259.