Learning to Extract Conditional Knowledge for Question Answering using Dialogue

Pengwei Wang
South China University of Technology
Guangzhou, China
w.pengwei@mail.scut.edu.cn

Lei Ji
Microsoft Research Asia
Beijing, China
leiji@microsoft.com

Jun Yan
Microsoft Research Asia
Beijing, China
junyan@microsoft.com

Lianwen Jin
South China University of Technology
Guangzhou, China
lianwen.jin@gmail.com

Wei-Ying Ma
Microsoft Research Asia
Beijing, China
wyma@microsoft.com

ABSTRACT

Knowledge based question answering (KBQA) has attracted much attention from both academia and industry in the field of Artificial Intelligence. However, many existing knowledge bases (KBs) are built by static triples. It is hard to answer user questions with different conditions, which will lead to significant answer variances in questions with similar intent. In this work, we propose to extract conditional knowledge base (CKB) from user question-answer pairs for answering user questions with different conditions through dialogue. Given a subject, we first learn user question patterns and conditions. Then we propose an embedding based co-clustering algorithm to simultaneously group the patterns and conditions by leveraging the answers as supervisor information. After that, we extract the answers to questions conditioned on both question pattern clusters and condition clusters as a CKB. As a result, when users ask a question without clearly specifying the conditions, we use dialogues in natural language to chat with users for question specification and answer retrieval. Experiments on real question answering (QA) data show that the dialogue model using automatically extracted CKB can more accurately answer user questions and significantly improve user satisfaction for questions with missing conditions.

Keywords

Knowledge based question answering; conditional knowledge base; dialogue

1. INTRODUCTION

Automatic question answering (QA) [9, 21, 7, 8] is becoming an important task in Artificial Intelligence. Compared with traditional manual QA services, automatic solutions can save both service cost and user time. Among various QA systems, knowledge based question answering (KBQA) [15, 1, 31] plays an important role. However, classical KBQA generally provides the same answer to questions having similar intent but with different conditions. For example, if a user question is "*How to free upgrade to windows 10?*", the answer will condition on the current product version specified by user since the answers for "*windows 8*" and "*windows xp*" are significantly different.

Inspired by this, we propose an end-to-end framework for extracting the conditional knowledge base (CKB) from the QA pairs. The format of the traditional knowledge base (KB) is generally a set of triples represented as (*subject*, *predicate*, *object*). The subject and object can be entities, and the predicate is the relation between subject and object. In comparison, our CKB is to add condition to typical facts triples, which is represented as (*subject*, *predicate*, *object* | *condition*). Specially, we define terms of this paper as follows:

- ***Subject***: It can refer to all possible entities.
- ***Predicate***: A relation (or intent) between the *subject* and *object*. In this work, a *predicate* is extracted from the patterns since a pattern can potentially reflect the intent of a question.
- ***Object***: The general attribute value of the *subject* and *predicate*. In this work, the *object* is an answer or solution given *subject* and *predicate*.
- ***Condition***: Some important information that can lead to different *objects* given same *subject* and *predicate*. The condition can refer to everything including the entities as long as they can lead to answer variances.

From the definition we can see that the conditions are obtained after processing the whole QA pairs. In other words, our model cannot recognize the condition from an input question without the learned CKB. For example, the condition cannot be recognized if we just input a question "*how to from windows xp free upgrade windows 10?*". After learning the CKB from the "*how to from windows xp free upgrade windows 10? – Answer 1*" and "*how to from windows 8 free upgrade windows 10? – Answer 2*", we can find that the "*windows xp*" and "*windows 8*" are the important conditions.

Table 1: Condition mining process.

Source	Number of Questions	Percentage of Questions with Missing Conditions
Software	864K	33.17%
Furniture	8K	30.86%
Entire dataset	5515K	28.16%

CIKM'16 , October 24-28, 2016, Indianapolis, IN, USA
© 2016 ACM. ISBN 978-1-4503-4073-1/16/10. . . $15.00
DOI: http://dx.doi.org/10.1145/2983323.2983777

Table 2: Condition mining process.

Original question: How to change login screen password on windows 10?				
Subject	Find another question with different answers	**Object**	**Predicate**	**Condition**
windows_10	***Original question***	answer1	change login screen	password
	How to change login screen background on *windows_10*?	answer2		background
login_screen_password	***Original question***	answer3	change	windows 10
	How to change *login_screen_password* on windows xp?	answer4		windows xp

Table 3: Conditional Knowledge Base.

Subject	**Predicate**	**Object(Answer)**	**Condition**
Windows 10	change_login_screen	press Ctrl+Alt+Delete, and then click Change...	password|pwd|pin
Windows 10	change_login_screen	Find the "disable lock screen" option first, then...	background|wallpaper
Office 365	download	download office 365 on the official website...	windows_8|win7
Office 365	download	download office 365 from apple app store	iphone|iphone_6|iphone_5
Bed	wash_surface	clear the surface with a damp cloth	baking_varnish|painting
Bed	wash_surface	better to use a dry cloth to scrub or ask for...	stickers|paper

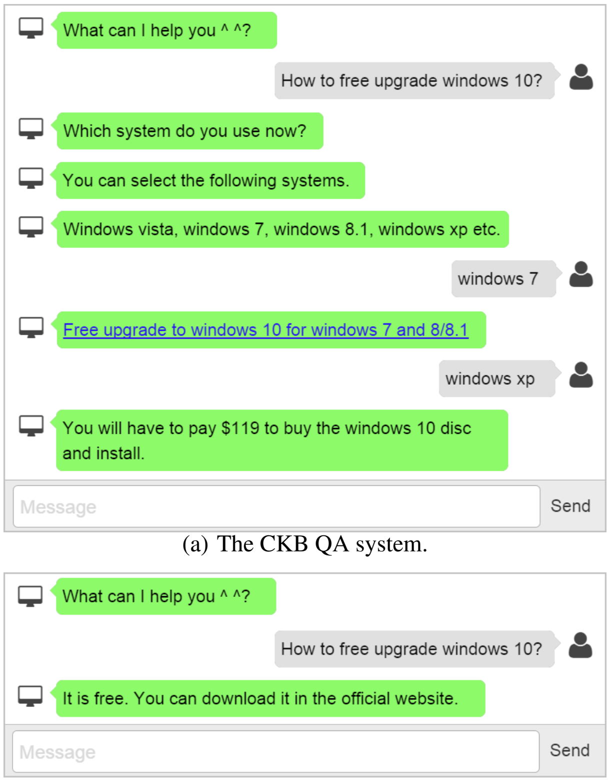
In our data study, many questions posted on search engines, forums and consultation websites frequently lack important information, which leads to uncertain answers. Table 1 gives a short summary of the statistics we did on QA datasets including the randomly selected software domain, furniture domain and the entire dataset. We learn the conditions from the QA pairs and count the question ratio that misses important conditions. From this statistic, we can see that many questions miss the conditions when users request answers. The details of the dataset and statistic will be introduced in experiments.

As shown in Table 2, the different tagged subjects of the same "*original question*" will lead to obtaining different CKBs. Thus, we propose to extract the CKB of one entity (subject) at a time. The proposed system first uses NER tool to tag entities in QA data with some labeled data. Notice the fact that if the data is too sparse, any algorithm will fail to mine knowledge from it. Thus, we select top fifty percent of the entities as the subjects according to the frequency in the QA data. The intermediate results of previous subjects can be used to improve the results of current subject's CKB extraction. Once a subject is selected, any other entities would be candidate conditions in CKB extraction of the current subject.

For each subject, we first mine the conditions and patterns in a bootstrapping manner. The position of a condition in the pattern is commonly defined as slot. For example, in the input question "*how to change login screen password on windows 10*", a pattern it satisfies can be "*change login screen SLOT0 on windows_10*" if the "windows 10" is tagged as the subject. Here, the "*SLOT0*" takes the condition of "*password*". After processing other similar questions, the "*SLOT0*" in above pattern is added with other conditions such as "*background*", "*passwd*", "*pin*", etc. Next, we propose an ***embedding based co-clustering*** algorithm to simultaneously group the similar conditions and patterns. Finally, the CKB can be constructed using the condition clusters and pattern clusters, as shown in Figure 5. Some representative words selected from a pattern cluster are regarded as the predicate. With a dialogue model using natural language, user will be asked to fill in the missing information if the user question lacks important conditions.

Figure 1 (a) shows an end-to-end dialogue example using CKB. In comparison, Figure 1 (b) is a frequently asked questions (FAQ) retrieval system result, and it only returns one answer every time no matter what the user condition is. In Figure 1 (a), when the user question misses the important condition, our dialogue model will chat with user and ask him to provide more information until this course ends.

Table 3 gives several examples of the CKB elements. Each line represents a knowledge record. The object is different based on different conditions. In this paper, we mainly introduce one type of conditional knowledge for "***how to***" questions as showcase, to verify the value of this research challenge and will expand it to general KB in future work. We only focus on the single condition missing task in this work, and will extend the multi-conditions missing task in future work.



What can I help you ^ ^?

How to free upgrade windows 10?

Which system do you use now?

You can select the following systems.

Windows vista, windows 7, windows 8.1, windows xp etc.

windows 7

Free upgrade to windows 10 for windows 7 and 8/8.1

windows xp

You will have to pay $119 to buy the windows 10 disc and install.

(a) The CKB QA system.

What can I help you ^ ^?

How to free upgrade windows 10?

It is free. You can download it in the official website.

(b) FAQ retrieval system.

Figure 1: Question answering systems examples

The contribution of this paper is ***conditional knowledge base construction*** using the ***question and answer pairs***. At the same time, we also ***construct a dialogue model*** to chat with user for specifying the missing condition. In our experiments, we mainly focus on the "***how to***" questions in our dataset as showcase. We test the proposed system on two datasets, which are query log files of a commercial search engine and Shopping QA dataset (Amazon and Walmart). The experimental results show that our proposed system can effectively learn the CKB from QA pairs for better question answering using dialogue than traditional QA with missing conditions.

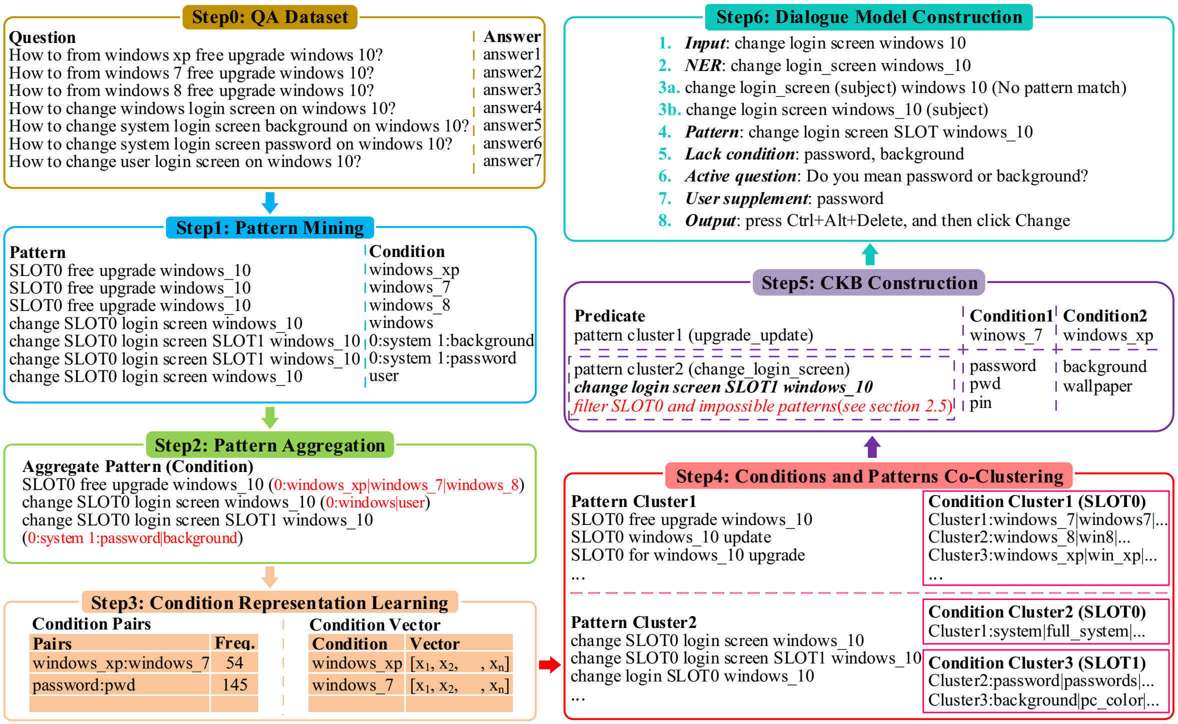

Step0: QA Dataset
Question
How to from windows xp free upgrade windows 10?
How to from windows 7 free upgrade windows 10?
How to from windows 8 free upgrade windows 10?
How to change windows login screen on windows 10?
How to change system login screen background on windows 10?
How to change system login screen password on windows 10?
How to change user login screen on windows 10?
Answer
answer1
answer2
answer3
answer4
answer5
answer6
answer7
Step1: Pattern Mining
Pattern
SLOT0 free upgrade windows_10
SLOT0 free upgrade windows_10
SLOT0 free upgrade windows_10
change SLOT0 login screen windows_10
change SLOT0 login screen SLOT1 windows_10
change SLOT0 login screen SLOT1 windows_10
change SLOT0 login screen windows_10
Condition
windows_xp
windows_7
windows_8
windows
0:system 1:background
0:system 1:password
user
Step2: Pattern Aggregation
Aggregate Pattern (Condition)
SLOT0 free upgrade windows_10 (0:windows_xp|windows_7|windows_8)
change SLOT0 login screen windows_10 (0:windows|user)
change SLOT0 login screen SLOT1 windows_10
(0:system 1:password|background)
Step3: Condition Representation Learning
Condition Pairs
Pairs
Freq.
windows_xp:windows_7
54
password:pwd
145
Condition Vector
Condition
Vector
windows_xp
$[x_1, x_2, \quad , x_n]$
windows_7
$[x_1, x_2, \quad , x_n]$
Step4: Conditions and Patterns Co-Clustering
Pattern Cluster1
SLOT0 free upgrade windows_10
SLOT0 windows_10 update
SLOT0 for windows_10 upgrade
...
Condition Cluster1 (SLOT0)
Cluster1:windows_7|windows7|...
Cluster2:windows_8|win8|...
Cluster3:windows_xp|win_xp|...
...
Pattern Cluster2
change SLOT0 login screen windows_10
change SLOT0 login screen SLOT1 windows_10
change login SLOT0 windows_10
...
Condition Cluster2 (SLOT0)
Cluster1:system|full_system|...
Condition Cluster3 (SLOT1)
Cluster2:password|passwords|...
Cluster3:background|pc_color|...
Step5: CKB Construction
Predicate
pattern cluster1 (upgrade_update)
pattern cluster2 (change_login_screen)
change login screen SLOT1 windows_10
filter SLOT0 and impossible patterns(see section 2.5)
Condition1
winows_7
password
pwd
pin
Condition2
windows_xp
background
wallpaper
Step6: Dialogue Model Construction
1. Input: change login screen windows 10
2. NER: change login_screen windows_10
3a. change login_screen (subject) windows 10 (No pattern match)
3b. change login screen windows_10 (subject)
4. Pattern: change login screen SLOT windows_10
5. Lack condition: password, background
6. Active question: Do you mean password or background?
7. User supplement: password
8. Output: press Ctrl+Alt+Delete, and then click Change

Figure 2: Our proposed system flow chart with true examples.

2. CONDITIONAL KNOWLEDGE BASED QUESTION ANSWERING

As introduced before, a *subject* is selected from the top fifty percent of the entities tagged by a NER tool[1][10]. Once a *subject* is selected, any other entities would be candidate conditions in CKB extraction of the current *subject*. Then for a single subject, our conditional knowledge based question answering (CKBQA) works as follows: 1. pattern mining; 2. pattern aggregation; 3. condition and pattern representation learning; 4. conditions and patterns clustering; 5. CKB construction; 6. dialogue model construction.

The steps (from the step 1 to step 5) are focus on single subject, and the step 6 is suitable for all subjects using the CKB generated from all subjects. If the user question is very clear, our system will give the answer directly. Otherwise, our system will chat with the user to clarify the conditions. Figure 2 shows the whole flowchart of our system with some real examples. In the following sub-sections, we will introduce each part of our system in detail.

2.1 Pattern Mining

To learn the conditional knowledge, we need to extract the conditions from a set of user questions. In this work, we use question pattern mining as the solution for condition learning. This step takes all questions with the same subject as input, and outputs question patterns and conditions in CKB. Bootstrapped systems are commonly used to learn entities [22, 11]. Our learning framework is based on the system SPIED-Learn [12], which builds upon previous bootstrapped pattern-learning work [22]. It learns conditions from unlabeled text via bootstrapping from seed dictionaries which contain the seeds mined in current subject, the seeds used in tagging subject process and the conditions mined from previous subjects. Patterns are learned using labeled conditions, and conditions are learned based on the extraction of learned patterns. The process is performed iteratively until no more patterns or conditions can be learned. For example, there is an input "*windows_xp free upgrade windows_10*". Meanwhile, the "*windows_10*" is selected as a subject and "*windows_xp*" is in the candidate conditions. Then, a pattern "*SLOT0 free upgrade windows_10*" is generated, where the "*SLOT0*" denotes the "*windows_xp*". After obtaining the pattern, a candidate condition "*win7*" can be extracted from the following sentence "*win7 free upgrade windows_10*" using the extracted pattern "*SLOT0 free upgrade windows_10*".

Next, we briefly introduce some operations in mining initial seeds. Our algorithm first preprocesses all the questions by removing the questioning words such as "*how to*", "*how do I*" etc. These phrases have no use in pattern mining. And then we use special type of words for question chunking. Here, special words include prepositions, copulas, interrogatives, conjunctions, modal verbs, personal pronouns, verbs and some stop words. Stanford POS tagger[2][26] is used to do the part-of-speech tagging. Finally, the remaining parts of a question are added into the seed dictionary.

After pattern mining, the conditions in the learned pattern are replaced by "*SLOT*" added by index. As shown in Step1 of Figure 2, our system can learn the pattern "*SLOT0 free upgrade windows_10*" and the condition "*windows_xp*" from the question "*How to from windows xp free upgrade windows 10*". The "*windows xp*" is replaced by the "*SLOT0*".

2.2 Pattern Aggregation

As shown in Step1 of Figure 2, some patterns have the same format but different conditions after pattern mining, we aggregate the same patterns with different conditions together. As shown in the example of Step2 of Figure 2, the pattern "*SLOT0 free upgrade windows_10*" should be aggregated, where the "*SLOT0*" contains three conditions: "*windows_xp*", "*windows_7*" and "*windows_8*".

2.3 Condition and Pattern Representation Learning

After learning patterns and conditions, we can get many patterns with many conditions in each pattern. As shown in Figure 3, our

[1] http://nlp.stanford.edu/software/CRF-NER.shtml

[2] http://nlp.stanford.edu/software/tagger.shtml

goal is to cluster the patterns which reflect the same user intent and cluster the conditions which have similar answers in a pattern cluster. Otherwise, we need to save the repeated patterns and conditions in each pattern.

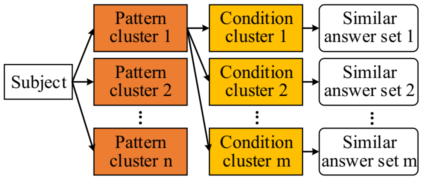


Figure 3: The tree structure of the clustering result.

To efficiently leverage semantic information of the patterns, conditions and answers to the following pattern and condition clustering stage, we propose an embedding based co-clustering algorithm. Thus, in this step we first focus on learning the embedding representation of patterns and conditions. This step takes the patterns, conditions and answers as input, and outputs vector representations of the patterns and conditions.

To encode the relation between question and answer into the embedding learning process, we find that if two different conditions in the same slot of the same pattern has similar answers, the two conditions should be semantically similar. Thus, we can obtain all the condition pairs and the corresponding co-occurrence frequencies under this assumption after processing all patterns and conditions, as shown in condition pairs part of Step3 in Figure 2. Similarly, we can also obtain the pattern pairs and the corresponding co-occurrence frequencies under the assumption that the two different patterns, which have same answers, should be semantically similar.

Notice the fact that some general conditions (e.g., prices) can occur in the questions of different subjects. To efficiently leverage these knowledge, we add the condition pairs mined from previous subjects into the current pairs set. If a previous pair has occurred in current pairs set, we just sum their frequencies. Otherwise, we add it as a new pair to current pairs set.

In addition, to leverage the co-occurrence relationships between conditions and patterns, we propose to jointly embed patterns and conditions into the same continuous vector space instead of training separately. We name the algorithm as ***patterns and conditions jointly embedding algorithm*** (**PCJE**). The co-occurrence relationship is used as the alignment of the pattern and condition.

Condition Embedding Model: In this model, we mainly encode the information of the raw input questions, answers and condition pairs into the embedding representation of conditions and other vocabularies of questions and answers. We use the *Skip-gram* [17] to model question and answer information:

$$J(\theta) = \frac{1}{T}\sum_{t=1}^{T} \sum_{-l\leq j\leq l, j\neq 0} logp(w_{t+j}|w_t) \tag{1}$$

$$p(w_{t+j}|w_t) = \frac{exp(\mathbf{w}_{t+j}^{\mathrm{T}}\mathbf{w}_t)}{\sum_{k=1}^{N} exp(\mathbf{w}_k^{\mathrm{T}}\mathbf{w}_t)} \tag{2}$$

where w_t and w_{t+j} represent the target word and contextual word respectively, and l is the size of context window centered at the target word w_t. $\mathbf{w}_{t+j}$, $\mathbf{w}_t$, $\mathbf{w}_k$ denote the vector of the w_{t+j}, w_t, w_k. N is the number of words in the vocabulary and T is the length of a question or answer. To leverage the condition pairs, a regularization function derived from the condition pairs information is added to the *Skip-gram* objective function. We use the following joint probability between the condition c_k and c_m to constraint the "*similar*"relation in condition pair (c_k, c_m).

$$p(c_k, c_m) = \frac{1}{1 + \exp(-\mathbf{c}_k^T \cdot \mathbf{c}_m)} \tag{3}$$

where $\mathbf{c}_k$ and $\mathbf{c}_m$ denote the embedding representation of the c_k and c_m. Then we encode the condition pairs information using a regularization function E_c:

$$E_c = \sum_{(k,m)\in P_c} w_{km}^c logp(c_k, c_m) \tag{4}$$

where P_c denotes the condition-condition pairs set and w_{km}^c denotes the co-occurrence frequency of the c_k and c_m. After combining the *Skip-gram* objective function and the regularization function, the goal of condition embedding model is to maximize the objective function J_c:

$$J_c = J(\theta) + \beta E_c \tag{5}$$

where β is the combination coefficient, which is set to 0.02 according to the validation set.

Pattern Embedding Model: In the pattern embedding model, we only have the pattern pairs information. Thus, we use the similar method as above to model the information of pattern pairs using the following equation:

$$J_p = \sum_{(k,m)\in P_p} w_{km}^p logp(v_k, v_m) \tag{6}$$

$$p(v_k, v_m) = \frac{1}{1 + \exp(-\mathbf{v}_k^T \cdot \mathbf{v}_m)} \tag{7}$$

where (v_k, v_m) denotes a pattern pair, P_p denotes the pattern- pattern pairs set and w_{km}^p denotes the co-occurence frequency of the v_k and v_m. $\mathbf{v}_k$ and $\mathbf{v}_m$ denote the embedding representation of the v_k and v_m. The goal of pattern embedding model is to maximize the objective function J_p.

Alignment Model: Now the condition embedding and pattern embedding are in distinct space, any computing between them is meaningless. Thus, we use the co-occurrence relation of pattern and condition to align the two distinct spaces, and model this relation using the following equation:

$$J_a = \sum_{(k,m)\in P_a} w_{km}^a logp(c_k, v_m) \tag{8}$$

$$p(c_k, v_m) = \frac{1}{1 + \exp(-\mathbf{c}_k^T \cdot \mathbf{v}_m)} \tag{9}$$

where (c_k, v_m) denotes a condition-pattern pair, P_a denotes the condition-pattern pairs set and w_{km}^a denotes the co-occurrence frequency of the c_k and v_m. $\mathbf{c}_k$ and $\mathbf{v}_m$ denote the embedding representation of the c_k and v_m. The goal of alignment model is to maximize the objective function J_a.

Considering the above three component models together, our goal is to maximize the following objective J, which can be optimized using back propagation neural networks:

$$J = J_c + J_p + J_a \tag{10}$$

According to validation set, the dimension of the condition embedding and the pattern embedding are set to 300, the windows size l in *Skip-gram* part is set to 5, and the learning rate is set to 0.01. Step3 in Figure 2 shows some examples of condition pairs and vector representation. In order to better represent the semantic relationship of the condition embedding, we show the condition word cloud in two dimensions space in Figure 4 using the t-SNE tool[3][27], which is a technique for dimensionality reduction. From Figure 4, we can see that different conditions with similar semantic meanings are grouped together.

2.4 Conditions and Patterns Clustering

Since the learned patterns can share the same intent and some different conditions will lead to the same answer, we propose a embedding based co-clustering approach to group similar patterns and conditions, which are co-correlated, simultaneously. This step takes patterns, conditions and corresponding embedding representations as input. It outputs pattern clusters and condition clusters.

[3] http://lvdmaaten.github.io/tsne/

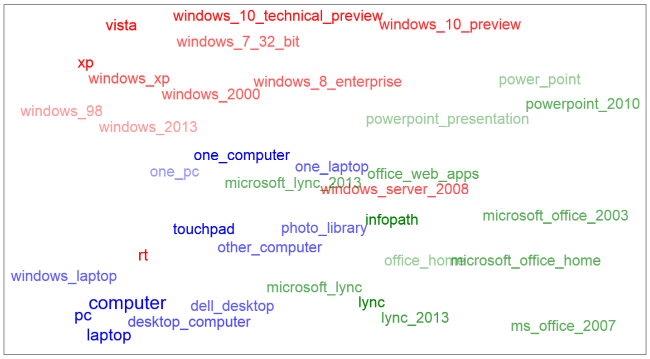


Figure 4: Condition word cloud.

Our embedding based co-clustering is based on the clustering algorithm described in X-Sim [4]. We select the hierarchical clustering algorithm as our base clustering solution in the embedding based co-clustering framework.

Algorithm 1 Embedding Based Co-clustering Algorithm

```
Input: Data matrix $\mathbf{M}$, condition prior similarity matrix $\mathbf{OC}$, pattern prior
    similarity matrix $\mathbf{OR}$, maximum number of generations $T$
Output: Ultimate condition similarity matrix $\mathbf{SC}$, ultimate pattern similar-
    ity matrix $\mathbf{SR}$
Initialize the $\mathbf{SR_0}$ and $\mathbf{SC_0}$ with the identity matrix $\mathbf{I}$;
for $t = 1$ to $T$ do
    $\mathbf{SR}_t = \alpha_1 \mathbf{MSC}_{t-1}\mathbf{M}^T \cdot \mathbf{NR} + (1-\alpha_1)\mathbf{OR}$, $nr_{i,j} = \frac{1}{|m_i| \cdot |m_j|}$;
    $\mathbf{SC}_t = \alpha_2 \mathbf{M}^T \mathbf{SR}_{t-1}\mathbf{M} \cdot \mathbf{NC} + (1-\alpha_2)\mathbf{OC}$, $nc_{i,j} = \frac{1}{|m^i| \cdot |m^j|}$;
    Set the diagonal of $\mathbf{SR}_t$ and $\mathbf{SC}_t$ to 1;
end for
```

Algorithm 1 summarizes our embedding based co-clustering procedure. Here, we use the classical notations: matrices (in uppercase letters) and vectors (in lower-case letters) are in bold, where all variables (in lower-case letters) are in italics. Let $\mathbf{M}$ be the data matrix representing a pattern-condition with n rows (patterns) and m columns (conditions and vocabulary of patterns); m_{ij} corresponds to the cosine similarity between the j^{th} condition embedding (and vocabulary of patterns) and the i^{th} pattern embedding; $m_i = [m_{i1}, ..., m_{im}]$ is the row vector representing the pattern i and $m^j = [m_{1j}, ..., m_{nj}]$ is the column vector representing the condition j or pattern vocabulary j of a pattern. $\mathbf{SR}$ and $\mathbf{SC}$ represent the square and symmetrical row similarity and column similarity matrices of size $n \times n$ and $m \times m$ respectively with $sr_{ij} \in [-1, 1]$, $1 \le i, j \le n$ and $sc_{ij} \in [-1, 1]$, $1 \le i, j \le m$. $\mathbf{OR}$ represents the pattern prior similarity matrix of size $n \times n$ with $or_{ij} \in [-1, 1]$, $1 \le i, j \le n$ denoting the cosine similarity between i^{th} pattern embedding and j^{th} pattern embedding. $\mathbf{OC}$ represents the condition prior similarity matrix of size $m \times m$ with $oc_{ij} \in [-1, 1]$, $1 \le i, j \le m$ denoting the cosine similarity between i^{th} condition embedding and j^{th} condition embedding. The parameter α is used to balance the importance of pattern-condition relation and external embedding. In our experiment, both α_1 and α_2 are set to 0.5 according to the validation set.

After obtaining the ultimate condition similarity matrix and pattern similarity matrix, the hierarchical clustering algorithm is used to cluster the conditions and patterns according to the ultimate similarity matrix. From the Step4 in Figure 2, we can see that the similar patterns and conditions are grouped together.

2.5 Conditional Knowledge Base Construction

This step takes pattern clusters and condition clusters as input. It outputs a set of triples with conditions in CKB. After clustering step, we can obtain the pattern clusters and condition clusters. The patterns in the same cluster will share the same intent, which is known as predicate in classical Knowledge Base. Before moving to next step, we define the missing percentage of slot as follows.

DEFINITION 1 (MISSING PERCENTAGE OF SLOT). *As an example, given a pattern "SLOT0 free upgrade windows_10", a query "free upgrade windows_10" can also match this pattern except for missing "SLOT0". That means this query lacks the "SLOT0" condition.*

Thus, we can calculate the missing percentage of each slot in pattern, with results shown in Table 1. Next, we filter out following three types of slots, whose existence has no effect on the answer selection.

- The slots containing only one cluster will be directly filtered out. As the example in Step4 of Figure 2, the "*SLOT0*" in the pattern, "*change SLOT0 login screen SLOT1 windows_10*", only contains Cluster1 in Condition Cluster2.
- The slots that almost no user will ignore although they can significantly affect the answers. As an example "*install SLOT0 on windows_10*", where "*SLOT0*" can be replaced by "*edge*", "*office*", etc. The "*SLOT0*" is a very important slot leading to different answers. However, almost no user will miss them in questions through our statistics.
- The slots that few users will care about although they seem very important. For instances, "*install windows_10 SLOT0*". The "*SLOT0*" can be replaced by "*enterprise*" or other suffixes. However, only few questions contain this condition.

The first type of slot can be directly filtered out. The two latter types of slots can be filtered out according to the missing percentage which is greater than 0.7 or less than 0.3. The 0.7 and 0.3 are selected based on the experiment results in this work. Finally, the remaining slots are regarded as very important slots.

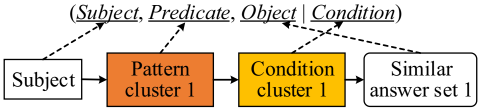


Figure 5: The map from cluster result to one term in CKB.

Finally, as shown in Figure 5, all parts of the CKB are extracted. The ***predicate*** of an item in CKB contains several patterns in a same cluster, and the ***condition*** of the item contains several conditions in a same cluster, and we select the most similar answer in the similar answer set according to the cosine similarity with the average embedding of the pattern cluster as the ***object*** of the item. To better represent the intent of the *predicate*, we select some words or phrases with high frequency in pattern cluster as the representative words of *predicate*. Here, we still use the pattern to match with question. A principled way to match the question is to use the representative words of *predicate*, which we leave as future work. Some examples are shown in Table 3. As we can see from the Step5 in Figure 2, the "*update windows_10*" is conditioned on "*windows 7*" or "*windows xp*".

2.6 Dialogue Model Construction

In this step, our goal is to match a pattern and figure out the missing condition given an input question. Thus, the dialogue model construction contains two steps:

1. Match the suitable pattern given the input question. If the suitable pattern has been matched and the important condition is missing, the algorithm will go to the step 2.
2. Our dialogue model will post a question to ask user to provide the condition in a natural language, and return a candidate condition list to user according to the recognized pattern.

For the first step, we first tag the entities in the question. Then, some candidate entities belonging to the subjects set are selected from the tagged entities. After that, for each candidate, we fix it as the subject, and match the suitable ***pattern***. If there is condition missing under this pattern, the pattern will be added into the pattern queue. Finally, the pattern queue will have the following states:

- The queue is empty. This shows that the question does not lack conditions or there is no pattern that can be matched. In this case, we will use the traditional retrieval based method [17, 19] to retrieve the similar question or best answer to the given question.
- There is only one pattern in the queue. In this case, the model will directly go to the step 2 of dialogue model construction.
- There are several patterns in the queue. In this case, we rank the patterns in the queue according to the similarity of pattern embedding and question embedding. Then, the best one is selected to be used in step 2.

For the second step, our dialogue model actively asks a question to let user provide the condition. It is unrealistic to show all the conditions in each condition cluster to user. Thus, we propose to show some representative concept names of the clusters and append few examples at the same time. For example, if the condition cluster contains "*windows 7*", "*windows xp*", etc., the representative concept name to describe them are "*system*". In some cases, there will obtain several concept names due to the multi condition clusters in a pattern cluster, as shown in Figure 3. In this paper, the concept names are also ranked according to the sum of condition frequency in each corresponding condition cluster. Our model will select the concept name with the highest frequency.

A web-based knowledge base, *Probase*[4] [30], is used to find the general concept name according to the items in cluster in this work. Here, the natural language is generated by some simple rule based patterns, such as "*which concept_name do you use now?*", "*do you mean concept_name_1 or concept_name_2?*" etc. We plan to find more patterns in future work.

After that, if the user can fill the condition, our model will return the corresponding answer in CKB to user. Otherwise, our model will still use the traditional retrieval based method [17, 19] to retrieve the similar question or best answer to the given question. As shown in the Step6 of Figure 2, our dialogue can help user to supply "password" to the original question, and output the correct answer. More dialogue examples will be introduced in experiments.

3. EXPERIMENTS

3.1 Data Collection

In this work, we use two datasets covered both explicit and implicit QA pairs. The former is the Amazon and Walmart Product Customer Questions Answers data[5] named as Shopping data and the latter is the queries with frequently clicked page titles from query log of a commercial search engine over one month period.

Shopping QA data contains 54,256 "*how to*" question answer pairs of which the "*Furniture*" category data size is 8,264. The length of the answer varies considerably, ranging from 2 to 61. In contrast, the average question length is 24. Spelling errors are very common in this collection.

The search query Log is obtained from one month query log files of a commercial search engine. We take the query and the corresponding title in the most frequently clicked page as the implicit QA pairs. In particular, we only select those queries that start with "*how to*". The query log data contains 5,461,268 "*how to*" QA pairs of which the "*Software*" category data size is 864,781. The average query length is 14, and the average title length is 26.

3.2 Results

3.2.1 CKB Quality Measure

Since the question answering using dialogue is based on the extracted CKB, we will firstly show the coverage and precision of the CKB we extracted as its quality measure.

The Coverage of CKB.

Intuitively, if most of the real questions could be covered by our top predicates in the CKB, it means our extracted CKB can handle most of questions in real QA scenarios. Here, we use the whole dataset to do this statistic. We first calculate the number of questions triggered by each predicate in the CKB, and then rank the predicates in descending order according to the size. The question coverage of the top n predicates is calculated as:

$$coverage@n = \frac{1}{N}\sum_{i=1}^{n} C_i \qquad (11)$$

where N denotes the number of questions in QA dataset, and C_i denotes the number of questions triggered by the i^{th} predicate. The result containing all the questions is shown in Figure 6. From the result, we can see that half of the predicates can cover most of the question, which means that the CKB can be used in real QA environment.

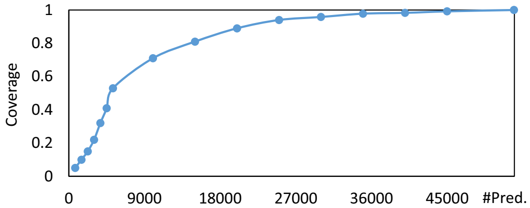


Figure 6: Question coverage of CKB.

The Precision of CKB.

To evaluate the precision of CKB, we randomly select 100 predicates from four sources to do manually labeling, which are *software*, *furniture*, *entire Shopping data* (random$_a$) and *entire search query data* (random$_s$). In addition, we randomly select 100 predicates from the entire data including the Shopping data and Search query data as the validation data set. Since one predicate may have multiple conditions, the number of knowledge triples obtained from each entity is given in Table 4. Given a triple in CKB, an annotator is asked to label it with 2-level score, 0 and 1. If the value in the triple is actually affected by the corresponding condition, the annotator will label it as 1; otherwise, the label is 0. Two annotators are involved in the annotation process. If a conflict happens, a third person will make judgement for the final result. The precision is calculated as:

$$precision = \frac{1}{N}\sum_{i=1}^{N} f(i) \qquad (12)$$

where $f(i)$ denotes the score of the i^{th} triple, and N denotes the total number of triples in each entity.

Table 4: Number of predicates and triples.

Entity	#Predicate	#Triples with conditions
Software	100	615
Furniture	100	498
Random$_a$	100	345
Random$_s$	100	597
Validation	100	563

The result of the CKB precision is shown in Table 5. We compare different approaches used in CKB extraction. The HAC(LINE) denotes that the hierarchical clustering algorithm is used to group the

[4] http://probase.msra.cn/Default.aspx

[5] http://www.amazon.com/ http://www.walmart.com/

condition and pattern separately. The feature of condition and pattern is generated by LINE [25] which takes the pairs of condition and pattern as input, and outputs the embedding of condition and pattern. The X-sim approach [4] is an original co-clustering algorithm, which just considers the co-occurrence relation between the condition and pattern. The Co-Clustering(Skip-gram) and Co-Clustering(LINE) are similar to our approach Co-Clustering(PCJE) except that the embedding of conditions and patterns are learned use the Skip-gram and LINE methods. The skip-gram embedding of pattern is obtained by splitting the pattern into words and summing the word skip-gram embedding. The last approach in the table denotes that the CKB extraction results of previous subjects are added into the current process. From the results we can see that our proposed method, Co-Clustering(PCJE), outperforms other previous methods and the result can be improved by adding previous mining results.

Table 5: The comparison of different approaches on CKB precision.

Domain	Software	Furniture	Random$_a$	Random$_s$
Metric	Prec.	Prec.	Prec.	Prec.
HCA(LINE)	59.67%	52.41%	48.7%	51.26%
X-sim	62.11%	51.2%	49.86%	50.42%
Co-Clustering(Skip-gram)	68.13%	60.64%	58.26%	60.8%
Co-Clustering(LINE)	70.89%	66.47%	64.06%	64.49%
Co-Clustering(PCJE)	80.16%	78.51%	82.32%	76.88%
Co-Clustering(PCJE)+Prev.	**89.92%**	**88.15%**	**88.99%**	**85.26%**

To better reflect the results affected by previous subject results, Figure 7 gives a plot of precision versus number of previous subjects results added into CKB extraction of current subject. From the result, we can see that the previous results can efficiently improve the performance of the current CKB extraction in the beginning of increasing the number of subjects. The precision increase flatten out after the number of subjects exceeds 900. This also shows that previous results provide lots of general pairs, which improve performance in the beginning. And the domain specific pairs are needed to further improve the performance after the number of subjects exceed 900.

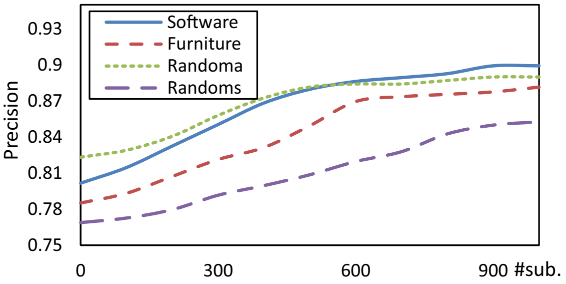


Figure 7: Precision versus number of previous subjects results.

3.2.2 Multiple QA Algorithms Comparison

In order to show the effectiveness of our proposed conditional knowledge based question answering (CKBQA), we compare it with other Question Answering system algorithms, including knowledge based question answering (KBQA), bag-of-word (BOW) based retrieval model and Skip-gram based retrieval model [17, 19].

Test Data Preparation.

Firstly, we access the queries with top 10 frequently clicked pages titles from query log files of a commercial search engine over recent one day period, whose period has no overlap with the previous 3 months of the training data. Then we randomly select 90 titles from the training dataset as the negative samples. Due to the fact that there is always only one answer corresponding to the question in the Shopping data, it is unfair to use this dataset directly. Thus, we propose to throw the question into the search engine, and take the top 9 titles returned by the search engine as the positive answers. Similarly, we use the same way to generate 90 negative samples. We name the 100 answers of a test question as candidate answers. The Shopping test set also has no overlap with previous training data. At last, there are 15,223 "*how to*" questions in the query log and 3,745 "*how to*" questions in the Shopping data.

Evaluation Protocol.

BOW: we directly use the question to rank the candidate answers according to the cosine similarity calculated by BOW vector. We use one-hot code as the question and answer BOW vector. The cosine similarity is calculated as:

$$similarity = \frac{\mathbf{A} \cdot \mathbf{B}}{\parallel \mathbf{A} \parallel \parallel \mathbf{B} \parallel} \tag{13}$$

where **A** and **B** are question vector and answer vector represented by the BOW model. The "·" denoted the vector dot product and the $\|\|$ denoted the magnitude of a vector. After ranking the candidate answer, the precision of the top n answers is calculated as:

$$precision@n = \frac{\mid \mathbf{A}_n \cap \mathbf{A}^* \mid}{\mid \mathbf{A}_n \mid} \tag{14}$$

where $||$ denotes set size, the $\mathbf{A}^*$ denotes the correct answer set, and $\mathbf{A}_n$ denotes the top n answer set.

Skip-gram: A skip-gram (word2vec[6][17]) model is trained on the whole training set and the default parameter of the model is adopted. The question and answer vector are generated by mean summation of each word vector. We also directly use the question to rank the candidate answers according to above cosine similarity between question vector and answer vector. After ranking the candidate answer, the precision@n is measured in the same way as BOW model.

KBQA: We construct the knowledge base without conditions using the following way. After learning CKB, the *object* whose condition cluster has the highest frequency is select as the unique object in term of one *subject* and *predicate*. Then we take the *subject*, *predicate* and the *unique object* as a triple in the knowledge base by ignoring the condition term. As shown in Figure 3, if the *condition cluster 1* has the highest frequency against other condition clusters, the *object* generated by *similar answer set 1* is select as the *unique object* in term of *subject* and *pattern cluster 1*. After that, the corresponding object is selected to rank the candidate answers using the BOW feature if a suitable pattern and subject in CKB can be matched by the input question. Otherwise, the question is directly used to rank the candidate answers. After ranking the candidate answer, the precision@n is measured in the same way as BOW model.

CKBQA: If a suitable pattern in CKB can be matched by the input question and a condition belonging to this pattern exists in the input question, the corresponding object of this condition is used to rank the candidate answers according to the BOW feature. In some cases, if there are several conditions existing in the question, the condition with the highest frequency is selected. If there is no condition of the pattern existing in the input question, we will use the same way as KBQA to rank the candidate answers. After ranking the candidate answer, the precision@n is measured in the same way as BOW model.

Results. The result is shown in Figure 8. From the figure, it is clear to see that our proposed model consistently outperforms other three question answering algorithms. It also shows that if the question contains the condition, our CKB model can provide

[6] https://code.google.com/p/word2vec/

more accurate answer. In the next section, we will show that our CKB model can also provide the accurate answer even though the question lacks conditions.

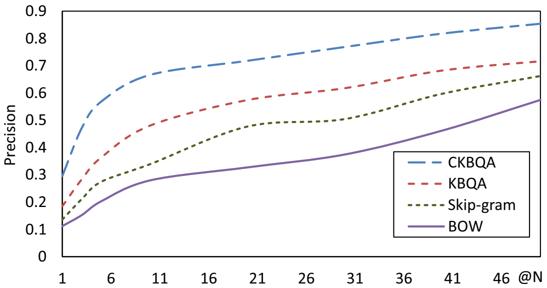


Figure 8: Answer precision in Question Answering system using different model.

3.2.3 *Answering Question without Condition Experiment*

To verify that our dialogue model using CKB can help user to select more accurate answer than general QA retrieval answer given question without condition, we design the following experiment to automatically simulate the dialogue environment.

Test Data Preparation.

We select 1000 QA pairs that are not used in the learning stage from above four sources (*software*, *furniture*, $random_a$ and $random_s$). Then we randomly delete some conditions in the questions manually. We name these questions as unconditional questions. To evaluate automatically, we consider the following method to let our dialogue model to generate supplementary information. If the supplementary condition list contains the corresponding deleted condition, we will take it as the supplementary condition. Otherwise, we take the most similar condition in the condition list with the deleted condition. We name the newly generated question as conditional question. Therefore, unconditional and conditional question share original answer. As an example the original question is "*how to change login screen password on windows 10*" and has a corresponding answer. The unconditional question after deleting condition is "*how to change login screen on windows 10*". Because the "*password*" is in our condition cluster, the conditional question generated by above method is "*how to change login screen password on windows 10*". The two questions should share the answer. At the same time, we use this correct answer and other 99 answers randomly selected from the whole answer set to construct a test answer set.

Evaluation Protocol.

Next, we use the conditional and unconditional questions to rank the answers in test answer set using bag-of-word feature. If the conditional question's result is better than the unconditional question's, it indicates the dialogue model generate useful information to the unconditional question. The metric is f-score, and it is calculated as:

$$f-score@n = \frac{prec.@n \times rec.@n}{prec.@n + rec.@n} \tag{15}$$

$$prec.@n = \frac{\mid \mathbf{A}_n \cap \mathbf{A}^* \mid}{\mid \mathbf{A}_n \mid} \qquad rec.@n = \frac{\mid \mathbf{A}_n \cap \mathbf{A}^* \mid}{\mid \mathbf{A}^* \mid} \tag{16}$$

where $||$ denotes set size, the $\mathbf{A}^*$ denotes the correct answer set, and $\mathbf{A}_n$ denotes the top n retrieval answer set.

Results. Figure 9 shows the results. Because there is only one right answer, the bigger the n, the lower is the precision. Thus, the f-score and the f-score differences between unconditional and conditional question become small as n increases. In the Figure 9, the unconditional and the conditional curve mean the unconditional and conditional question respectively. From the results, we can see that our dialogue model can help user to get more accurate answer, especially when n is small.

3.2.4 *User Satisfaction Survey*

On the other hand, to verify whether this CKB based dialogue can improve the user experiences, the user satisfaction is evaluated by manual user survey work. The satisfaction means whether the user feels the system help them find the right answer easily when compare with the results directly retrieved. We find 100 users to try the system. Each of them can ask 50 questions, and the answer result label is on 2-level score, 1 and 0, where score 1 means dialogue answer is better than general QA with condition, and score 0 means general QA is better than dialogue answer. The satisfaction is calculated as:

$$satisfaction = \frac{1}{N}\sum_{k=1}^{N} f(k) \tag{17}$$

where $f(k)$ denotes the score of the k^{th} question, N denotes the total number of question. The result is shown in Figure 10. We calculate the user percentage as:

$$Perc._{(a,b]} = \frac{1}{M}\sum_{i=1}^{M} I_{(a,b]}(\frac{1}{N}\sum_{k=1}^{N} f_i(k)) \tag{18}$$

$$I_{(a,b]}(v) = \begin{cases} 1 & \text{if } v \in (a,b] \\ 0 & \text{else} \end{cases} \tag{19}$$

where M denotes the total number of user, N denotes the total number of question, and $f_i(k)$ denotes the score of the k^{th} question in i^{th} user. From the figure, we can see that most of users are satisfactory with our system. This also shows that our dialogue with CKB can help people select more accurate answer.

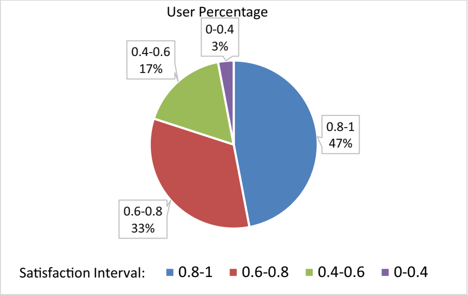
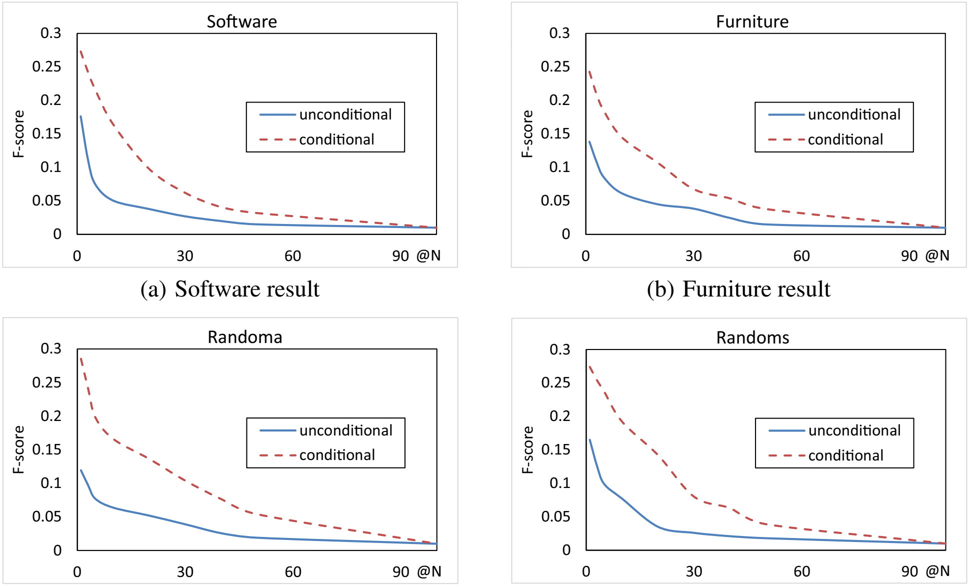


Figure 10: User satisfaction statistic.

3.2.5 *CKB Dialogue Examples*

Finally, we use some examples of the CKB dialogue between users and the system for insights. Sample conversations involving the three considered entities are given below, where **Machine** denotes our dialogue model, and **Human** denotes the human user interacting with it.

Conversation 1:
Machine: *What can I help you?*
Human: *How can I download the Office 365?*
Machine: *Which system or device do you use now?*
Machine: *You can select the following systems or devices.*
Machine: *Windows vista, windows 7, windows 8, windows 8.1, windows xp, ios, iphone 6, iphone 5 or others.*
Human: *Windows 8*
Machine: *Download office 365 on the official website. http://products.office.com/en-us/home*
Human: *iPhone 6*
Machine: *Download office 365 from apple app Store.*



(a) Software result

(b) Furniture result

(c) Random$_a$ result

(d) Random$_s$ result

Figure 9: Four datasets retrieval results using unconditional and conditional questions

Conversation 2:
Machine: *What can I help you?*
Human: *How can I wash the bed?*
Machine: *What is the material of the bed?*
Machine: *You can select the following materials.*
Machine: *Baking varnish, painting, stickers, paper, or others.*
Human: *Baking varnish.*
Machine: *Clear the surface with a damp cloth.*
Human: *What about stickers?*
Machine: *Better to use a dry cloth to scrub or ask for a professional worker to clean.*

Conversation 3:
Machine: *What can I help you?*
Human: *How to create a variable in computer programming?*
Machine: *Which programming language do you use?*
Machine: *You can select the following systems.*
Machine: *C#, C++, Python or others.*
Human: *Python*
Machine: *Python variables do not need explicit declaration to reserve memory space. The declaration happens automatically when you assign a value to a variable*

4. RELATED WORK

In this section, we briefly review the work that is close to this paper. We organize the related work in two subsections: knowledge based question answering and dialogue model.

Knowledge Extraction and Knowledge Based Question Answering: Thanks to the long-term efforts made by experts, crowdsourcing and even machine learning techniques, several web-based knowledge repositories have been built, such as WordNet, Freebase. Besides, the automatic knowledge extraction [22] is also popular, such as Probase [30] and NELL [6, 20]. Knowledge based Question Answering (KBQA) has attracted increasing interests from both industry and academia [15, 1, 5]. There are many Question Answering (QA) systems [13, 9, 7, 8], such as IBM's Watson, which is highly optimized for vertical domains [9] and Stanford's semantic parsing via paraphrasing [2, 3]. Large scale structured KBs are widely adopted in modern web search engines such as Bing and Google. However, typical KBQA focuses on simple questions that can be answered with facts [14], and cannot handle questions without conditions well. Another drawback of existing KBs is that people cannot construct dialogue model with them. In this paper, we aim to construct CKB to answer conditional questions using dialogue. Note detailed discussion for question understanding and KB matching are out of the scope of this paper.

Dialogue Model: Dialogue model [16, 29, 14] has been pursued by many researchers over the last decades, e.g., Microsoft XiaoIce and Apple Siri. However, frame, namely the conditions in our model, always need to be generated manually in most of these systems. The recent success of neural network modeling [18, 24, 28, 23] shows that recurrent neural networks are rather effective models for dialogue model. However, this straightforward model needs a large conversational training data if it generates simple conversations. Large scale of QA pairs on the web cannot be used directly for dialogue training, which is different from the scope of this paper.

5. CONCLUSION AND FUTURE WORK

In conclusion, we presented a novel end-to-end system for conditional knowledge base learning. The system learns conditions and patterns in bootstrapping manner, and uses co-clustering algorithm to group the conditions and patterns by adding the answer as supervised information. Finally, we generated the predicates and conditions from pattern clusters and condition clusters to build a dialogue model for automatic question answering. The dialogue with CKB can help people select more accurate answer.

In next step work, we will handle multi-conditions missing problems instead of handling one condition missing problems now. At the same time, we will combine our conditional feature with other QA algorithms. We believe that the performance can be potentially improved. Besides, in this paper, we only define a simple type of conditional knowledge to verify our assumption. We will expand to more facts types and generate the general KB in our next work.

6. ACKNOWLEDGMENTS

This research is supported in part by NSFC (Grant No.: 61472144), the National Key Research & Development Plan of China (Grant No.: 2016YFB1001405), GDSTP (Grant No.: 2013B010202004, 2015B010131004)

7. REFERENCES

[1] J. Bao, N. Duan, M. Zhou, and T. Zhao. Knowledge-based question answering as machine translation. *In Proceedings of the 52nd Annual Meeting of the ACL*, 2014.

[2] J. Berant, A. Chou, R. Frostig, and P. Liang. Semantic parsing on freebase from question-answer pairs. *In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1533–1544, 2013.

[3] J. Berant and P. Liang. Semantic parsing via paraphrasing. *In Proceedings of the 52nd Annual Meeting of the ACL*, 2014.

[4] G. Bisson and F. Hussain. X-sim: a new similarity measure for the co-clustering task. *In Proceedings of the 2008 Seventh International Conference on Machine Learning and Application-Volume(ICMLA)*, pages 211–217, 2008.

[5] A. Bordes, J. Weston, and S. Chopra. Question answering with subgraph embeddings. *In Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2014.

[6] A. Carlson, J. Betteridge, J. Kisiel, B. Settles, E. H. Jr, and T. Mitchell. Towards an architecture for never-ending language learning. *In Proceedings of the 2010 Conference on Association for the Advancement of Artificial Intelligence (AAAI)*, 2010.

[7] A. Fader, L. Zettlemoyer, and O. Etzioni. Paraphrase-driven learning for open question answering. *In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1608–1618, 2013.

[8] A. Fader, L. Zettlemoyer, and O. Etzioni. Open question answering over curated and extracted knowledge bases. *In Proceedings of the 20th conference on Knowledge discovery and data mining (SIGKDD)*, pages 1156–1165, 2014.

[9] D. Ferrucci, E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. Kalyanpur, A. Lally, J. Murdock, E. Nyberg, J. Prager, N. Schlaefer, and C. Welty. Building watson: An overview of the deepqa project. *In Proceedings of the 2010 Conference on Association for the Advancement of Artificial Intelligence (AAAI)*, 2010.

[10] J. R. Finkel, T. Grenager, and C. Manning. Incorporating non-local information into information extraction systems by gibbs sampling. *Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 363–370, 2005.

[11] S. Gupta and C. Manning. Improved pattern learning for bootstrapped entity extraction. *In Proceedings of the Eighteenth Conference on Computational Natural Language Learning (CoNLL)*, pages 68–77, 2014.

[12] S. Gupta and C. Manning. Spied: Stanford pattern-based information extraction and diagnostics. *In Proceedings of the ACL 2014 Workshop on Interactive Language Learning, Visualization, and Interfaces (ACL-ILLVI)*, 2014.

[13] X. Hao, X. Chang, and K. Liu. A rule-based chinese question answering system for reading comprehension test. *In proceedings of IEEE third International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIHMS)*, pages 325–329, 2007.

[14] D. Jurafsky and J. Martin. Speech and language processing. pearson international. 2009.

[15] O. Kolomiyets and M. Moens. A survey on question answering technology from an information retrieval perspective. *Information Sciences*, 2011.

[16] J. Lester, K. Branting, and B. Mott. Conversational agents. *In Handbook of Internet Computing*, 2004.

[17] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *In ICLR Workshop*, 2013.

[18] T. Mikolov, M. Krarafiat, L. Burget, J. Gernocky, and S. Khudanpur. Recurrent neural network based language model. *In Proceedings of INTERSPEECH*, pages 1045–1048, 2010.

[19] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of phrases and their compositionality. *In Proceedings of the 27th Annual Conference on Neural Information Processing Systems (NIPS)*, pages 3111–3119, 2013.

[20] T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. Never-ending learning. *In Proceedings of the 2015 National Conference on Association for the Advancement of Artificial Intelligence (AAAI)*, 2015.

[21] J. Moré, S. Climent, and M. Coll-Florit. An answering system for questions asked by students in an e-learning context. *Universities and Knowledge Society Journal (RUSC)*, pages 229–239, 2012.

[22] E. Riloff. Automatically generating extraction patterns from untagged text. *In Proceedings of the 13th National Conference on Association for the Advancement of Artificial Intelligence (AAAI)*, pages 1044–1049, 1996.

[23] L. Shang, Z. Lu, and H. Li. Neural responding machine for short-text conversation. *In Proceedings of ACL*, pages 1044–1049, 2015.

[24] I. Sutskever, O. Vinyals, and Q. Le. Sequence to sequence learning with neural networks. *In Proceedings of the 2014 Annual Conference on Neural Information Processing Systems (NIPS)*, 2014.

[25] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. Line: Large-scale information network embedding. *In Proceedings of the 24th International Conference on World Wide Web (WWW)*, pages 1044–1049, 2015.

[26] K. Toutanova, D. Klein, C. Manning, and Y. Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. *In Proceedings of HLT-NAACL*, pages 252–259, 2003.

[27] L. van der Maaten. Accelerating t-sne using tree-based algorithms. *Journal of Machine Learning Research*, pages 15(1):3221–3245, 2014.

[28] O. Vinyals and Q. Le. A neural conversational model. *In Proceeding of the 31st International Conference on Machine Learning, JMLR*, 2015.

[29] T. Will. Creating a dynamic speech dialogue. *VDM Verlag Dt*, 2007.

[30] W. Wu, H. Li, H. Wang, and K. Zhu. Probase: A probabilistic taxonomy for text understanding. *In Proceedings of the 2012 International Conference on Management of Data (SIGMOD)*, pages 481–492, 2013.

[31] X. Yao and B. Durme. Information extraction over structured data: Question answering with freebase. *In Proceedings of the 52nd Annual Meeting of the ACL*, 2014.