Continuous and Interactive Factual Knowledge Learning in Verification Dialogues

Sahisnu Mazumder [†], **Bing Liu** [†], **Nianzu Ma** [†], **Shuai Wang** [‡*]
[†]Department of Computer Science, University of Illinois at Chicago, USA
[‡]Amazon AI
sahisnumazumder@gmail.com, liub@uic.edu
jingyima005@gmail.com, shuaiwanghk@gmail.com

Abstract

Knowledge bases (KBs) used in applications such as dialogue systems need to be continuously expanded in order to serve the users well. This process is known as *knowledge base completion* (KBC). A piece of *knowledge* or a *fact* is often represented as a triple (s, r, t), meaning that the entity s and the entity t have the relation r or are linked by r. KBC builds a model to infer missing facts from the existing ones in a given KB. Existing KBC research typically makes the *closed-world assumption* that to infer a new fact (s, r, t), it assumes that s, r and t are already in the KB, but are not linked. Clearly, this assumption is a serious limitation. In this paper, we eliminate this assumption and allow s, r and/or t to be unknown to the KB, which we call *open-world knowledge base completion* (OKBC). We focus on solving OKBC via user interactions, which enables the proposed system to potentially serve as an engine for learning new knowledge during dialogue. Experimental results show the effectiveness of the proposed approach.

1 Introduction

Knowledge bases (KBs) are instrumental to many AI applications, e.g., question answering [5], dialogues [18, 56, 29, 59]. Although many large KBs exist like Freebase [6] and WordNet [34], they miss a large percentage of facts about common entities [52], which is also true for numerous application specific KBs. In recent years, researchers have studied ways of expanding KBs automatically. This area is called *knowledge base completion* (KBC) [24, 16]. This paper studies this problem further, but unlike most existing studies, we study the problem in the *open-world setting* (to be defined shortly).

The type of knowledge that we focus on is the fact triples commonly used in *knowledge graphs*, (s, r, t), which means that the *source entity* s and the *target entity* t can be linked by the *relation* r. For example, (*Obama*, *CitizenOf*, *USA*), meaning that *Obama is a citizen of USA*. Traditional KBC infers new facts (s, r, t) from existing ones in the KB, which is often defined as a binary classification problem: Given a (query) triple (s, r?, t), it predicts whether s and t can be linked by r, where ? indicates a query. Existing work typically makes the *closed-world assumption* that s, r and t are all *known* to exist in the KB [24, 11, 9, 37]. This paper removes this assumption and allows s, r and/or t not to exist in the KB. We call this *open-world KBC* (OKBC). The main existing approach to solving OKBC is to extract facts from documents and add them to the KB [35, 21, 3, 54, 14, 45, 44]. This paper proposes a dialogue-based approach, which makes it naturally suitable for dialogue systems.

The motivation for proposing an interactive approach to solving OKBC came from the need of dialogue systems or *chatbots* like Siri, Amazon Alexa, Google Assistant, not to mention numerous customer support bots used by large companies. Chatbots often use KBs to answer user questions.

[*]Work done while author was at the University of Illinois at Chicago.

Preprint. Under review.

USER:	(*Obama*, *CitizenOf?*, *USA*) "Is Obama a citizen of USA?"	[**Query**]
IKAI:	I do not know what "**CitizenOf**" means? Can you provide me an example?	[***Ask for Clue***]
USER:	(*David Cameron*, *CitizenOf*, *UK*). "David Cameron is a citizen of UK."	[**SF1**]
IKAI:	Got it. Can you tell me how "**Obama**" and "**Honolulu**" are related?	[***CLQ***]
USER:	(*Obama*, *BornIn*, *Honolulu*). "Obama was born in Honolulu."	[**SF2**]
IKAI:	Got it. Can you tell also me how "**Honolulu**" and "**Hawaii**" are related?	[***MLQ***]
USER:	(*Honolulu*, *CapitalOfState*, *Hawaii*). "Honolulu is the state capital of Hawaii."	[**SF3**]
IKAI:	(*Obama*, *CitizenOf?*, *USA*) holds. "Yes, Obama is a US citizen."	[**Answer**]

Figure 1: An example of interactive learning and inference. Each triple above is assumed to be extracted from the sentence after it. ***Ask for Clue***, ***CLQ*** and ***MLQ*** are interaction query types, discussed in Sec 3.

Many deep learning methods now use KBs too [18, 56, 29, 59]. However, the incompleteness [52] of the KBs limits the scope of their applications. It is desirable for a chatbot to learn new knowledge in the dialogue process to expand its KB to make it more and more knowledgeable over time [12, 27].

There are many opportunities to learn during a dialogue. For example, we can directly extract knowledge from user utterances using information extraction methods [28, 19]. We do not study that in this paper. This paper focuses on another opportunity, i.e., when the user asks the chatbot a *factual verification* (yes/no) question that the chatbot cannot answer. To learn from the user in this new setting is challenging because it not only involves extraction but also asking user questions and making inference. Note that this is the second part of our work on continuously learning factual knowledge during conversation. The first part (called CILK) is presented in [31], which studies how to interactively learn new knowledge when the user asks a *WH-question* that the system is unable to answer. A more general framework of lifelong/continual learning in dialogues is given in [27].

KBC does not form a suitable model for this dialogue-based knowledge learning because in a conversation the user may ask or say anything, which may include entities and/or relations that are not in the KB. We thus need OKBC in the dialogue or conversation context. This paper proposes a novel interactive method to solve OKBC, called *Interactive Knowledge Acquisition and Inference* (IKAI), which involves two tasks: **(1)** ***Interactive acquisition of supporting facts***, i.e., formulating an inference strategy to ask the user suitable questions to convert an OKBC query to a KBC query. Those user answers, which are *supporting facts*, are also added to the KB. **(2)** ***Knowledge inference***, building a predictive model using the (acquired) supporting facts and the (existing) knowledge in the KB to infer the answer (i.e. whether the converted KBC query is true or not) of the original user query. Fig. 1 shows an example. Here, IKAI acquires *supporting facts* SF1, SF2, SF3 to accomplish the *interactive knowledge acquisition* and then, utilizes these pieces of knowledge along with existing KB facts to *infer* over the query relation "*CitizenOf*" to answer the user query. The new knowledge (which are supporting facts) is also added to the KB for future use.

This paper only focuses on developing the core learning engine. It does not study fact or relation extraction from user utterances, entity linking, etc., which have been studied before. Thus, IKAI only works with structured *fact verification* query triples (s, r?, t), where ? indicates a query, e.g., (*Obama*, *CitizenOf?*, *USA*) meaning "*Is Obama a citizen of USA?*" and assumes that a relation extraction (e.g., OpenIE [2]) system is available to convert natural language queries or statements from the user into query triples (s, r?, t) or (supporting) fact triples (s, r, t). It is also assumed that the user has good intentions, i.e., user answers are 100% correct, but we do not assume that the user can answer all questions as opposed to the teacher-student setup, where the teacher is assumed to know everything. See [27] for strategies on how to deal with incorrect knowledge from users.

IKAI is implemented with a *Finite State Machine* (FSM) and a *Predictor*. The FSM dynamically formulates an inference strategy for a given OKBC query. While executing the strategy, IKAI processes the query and acquires the supporting facts by asking the user and thereby, transforms an OKBC problem to a KBC one (i.e., all s, r and t become *known* to the KB). The *Predictor* is invoked for knowledge inference to solve the *transformed* KBC problem. We evaluated IKAI using two real-world KBs: *Freebase* and *WordNet*. Experimental results demonstrate its effectiveness.

2 Related Work

KBC has been studied extensively [36, 13, 11, 8, 16]. The work in [36] partially deals with OKBC, but it cannot handle unknown entities. The work in [45, 44] solves OKBC using an external text corpus to perform inference on unknown entities. They cannot handle unknown relations. They or other existing methods do not perform KBC or OKBC through user interactions like IKAI.

IKAI is also related to interactive language learning (ILL) [51, 50], but ILL does not grow KB. Among dialogue systems [48, 25, 26, 10, 53, 57], the work in [26] allows the learner to ask questions. However, it learns only about *whether to ask*. The *what to ask* aspect is manually designed. More importantly, unlike IKAI, its learning is for building a dialogue system in the *Teacher-Student setup*.

The most closely related works to IKAI are those in [31, 28, 19]. [31] proposed a method CILK to acquire knowledge when a WH-question is asked by the user. CILK's method of dealing with WH-questions by trying everything in the KB is entirely different from dealing with yes/no questions in IKAI. IKAI's inference model is comparatively more interpretable and only requires learning of embeddings of relations rather than that of *both* entities and relations as in CILK and thus, do not need to acquire many facts per query entity in a given dialogue session. [28, 19] are based on extraction of facts directly from user utterances. IKAI is based on answering user queries and it formulates a strategy to ask and also perform inference. [38, 39] studied lexical knowledge acquisition in dialogues, whereas IKAI is about interactive factual knowledge learning and inference.

Recently, many conversational systems have been proposed, e.g., Conversational Recommender Systems [46, 1, 58] and Information-seeking Conversation [55, 40]. However, they do not acquire new knowledge to deal with OKBC in conversation. There are also works on KB question-answering (KBQA) [22, 7, 4, 42, 41], open-vocabulary semantic parsing [15, 47], information extraction [43, 2], and KB updating using facts extracted from the Web [35]. We do not do KBQA, semantic parsing or Web fact extraction, which is complementary to our work. Rather, we focus on solving OKBC interactively, i.e., learning through interactions with the user.

3 IKAI: The Proposed System

As discussed in the introduction, IKAI performs the following two main tasks:

Task 1 : Converting OKBC to KBC. Given an OKBC query (s, r?, t) [e.g., (*Obama*, *CitizenOf*?, *USA*) - *Is Obama is citizen of USA?*], IKAI formulates some relevant questions to ask the user (see Fig. 1) to acquire supporting facts (SFs), addition of which (in KB) convert an OKBC query (s, r?, t) to a KBC query. The KBC query is also (s, r?, t) except that all components, s, r, and t, must already exist in the current KB. OKBC allows any of s, r and t to be *unknown* or not in the KB.

The conversion involves the following two cases- **(1) If r is unknown**, IKAI asks the user to provide a ***clue*** [an example triple r]. In Fig. 1, IKAI asks the user to provide an example for the *unknown* relation "*CitizenOf*" and the user gave the clue (*David Cameron*, *CitizenOf*, *UK*), meaning *David Cameron is a citizen of UK*. With this clue added to the KB, the relation "*CitizenOf*" becomes known. If "*David Cameron*" or "*UK*" is not in the KB, the clue may need to be further processed, i.e., using the case below. **(2) If s or t is unknown**, IKAI asks the user to provide a ***link*** (relation) to connect the unknown entity s or t with an automatically selected existing entity (discussed in "Actions" in Sec. 4) in the KB. In Fig. 1, IKAI asks the user to provide a relation to connect the *unknown* entity "*Obama*" with an existing KB entity "*Honolulu*" [Can you tell me how "*Obama*" and "*Honolulu*" are related?] (discussed later). The acquired knowledge (the two user answers) reduces OKBC to KBC as all r, s and t are now in the KB. We deal with the case when the user is unable to answer later.

Task 2: Solving KBC by Building a Predictor. With the resulting KBC query, $(s, r?, t)$, IKAI uses a *path-ranking* (PR) approach [24, 23, 17, 36] to build the predictive model (Predictor) to predict if $(s, r?, t)$ is true. We choose PR due to its better performance and interpretability [49]. PR enumerates relation paths between two entities (s, t) in a KB (encoded as a multi-relation graph) and uses those relation paths as features to train the predictive model for relation r. Given a sequence of entities and relations $\langle s, r_1, e_1, r_2, e_2.., r_l, t\rangle$ linking s to t in the KB, we use the term ***path feature*** to refer to the *relation path* $\langle r_1, r_2, .., r_l\rangle$ and the full sequence $\langle s, r_1, e_1, r_2, e_2.., r_l, t\rangle$ as simply ***path*** onwards. An entity e_i appearing in the path *excluding the source and target entities* is termed as an ***intermediate*** entity for the path. *A path feature is extracted from a path by just excluding the entities in the sequence.* To enumerate a set of path features for entity pair (s, t), PR performs random walks over KB to find paths starting from s (or t) and ending at t (or s). Once relation paths (features) are extracted for a set of positively labeled (linked by r) and a set of negatively labeled (not linked by r) entity-pairs, a binary classifier (Predictor) is trained for relation r using those features. The trained classifier for r is then used to predict if a given test entity pair (not linked by r in the KB) can be linked by r.

In this work, we adopt the PR method, C-PR [30], which can leverage the **context relatedness** of the source s and target t during path enumeration to make more accurate prediction. Given s and t,

C-PR performs context-aware bidirectional random walk to enumerate path features. In particular, C-PR enumerates a forward path starting from s and a backward path starting from t such that the intermediate entities appearing in a path (forward or backward) have high contextual similarity with respect to both s and t. The contextual similarity between any two entities is computed as the cosine similarity between the corresponding entity embedding vectors learned from text documents using neural word embedding [32, 33]. C-PR treats entities in the KB as words and learns and leverages their embeddings to guide the path exploration process. If a common entity appears in both forward and backward path at any time step during exploration, C-PR stops exploration, merges the forward and backward paths into a single path from s to t, and returns the relation sequence in the merged path as a *contextual relation path* feature from s to t, which is then used for inference.

Note, in bidirectional path exploration, C-PR may get stuck with e_f (e_b) being the last forward (backward) node without being linked to form a full path. Unlike C-PR, in IKAI, we connect the forward and backward path sequence by linking e_f and e_b with a template (fake) relation "-?-" to create a full path (called a ***incomplete path***). We call "-?-" the missing link marker. For an incomplete path, IKAI asks the user to provide a relation/link to connect (e_f, e_b) for path completion. We refer to such a query as a **missing link query (MLQ)** (see Sec. 3.1 for more details).

IKAI manages the aforementioned two tasks and also data collection for building the predictive model for task 2 (discussed in Sec 3.2) using a Finite State Machine (FSM) as discussed below.

3.1 Finite State Machine of IKAI

IKAI manages everything using a Finite State Machine (FSM). It formulates and simultaneously executes an *inference strategy*. That is, at each state S_i, FSM chooses an action a_j to execute. An action can be a *processing action* or an *interaction action*. A processing action advances the reasoning (like searching query entities and relation in the KB, extracting path features etc.). An interaction action asks and acquires knowledge from the user. After an action is executed, FSM goes to a new state S_{i+1}. In the process, an OKBC problem is transformed to a KBC problem. Additionally, the process also performs path feature extraction (processing actions) for a clue triple or a query triple for training the Predictor. When FSM transits into its final state, the *Predictor* is invoked to train to solve the reduced KBC problem to answer the original user query. Table 1 gives a description of the parameters of IKAI referred on-wards.

IKAI's FSM is $(\mathcal{S}, \mathcal{A}, \mathcal{S}_0, \mathcal{S}_F, \Delta)$, where $\mathcal{S}$ is the finite set of states, $\mathcal{A}$ the set of actions, $\mathcal{S}_0$ the set of initial states, $\mathcal{S}_F$ the set of terminal states and $\Delta : (\mathcal{S}, \mathcal{A}) \rightarrow \mathcal{S}$ the state transition function. We denote a triple (s, r, t) under processing as d on-wards. Note, d can be the OKBC query (denoted as q) from the user or a clue triple (denoted as c^+) given by the user for an unknown relation (e.g., *CitizenOf* in Fig. 1). As one or both of the entities [e.g., "*David Cameron*" and/or "*UK*" in the clue (*David Cameron*, *CitizenOf*, *UK*)] may be unknown, IKAI also process clues just like OKBC queries.

Whenever IKAI receives a clue from the user, it postpones the processing of the OKBC query and initiates the strategy formulation for each clue triple consecutively. Once all clue triples are processed (i.e., strategies are formulated and executed), the unknown query relation is now in the KB and IKAI can then infer the OKBC query. Thus, the strategy formulation for OKBC query resumes. A *Processing Stack* (PS) is used to manage the process, which basically holds the triples to be processed by IKAI along with FSM state information. Whenever IKAI receives d (a query or a clue) from the user, it pushes d and the initial state (S_0) corresponding to d [see "States" below] onto PS as a pair (d, S_0). FSM processes the stack top $PS[top]$ and $PS[top]$ gets modified on a state transition due to execution of some action a_j. After the processing of d finishes, i.e., $PS[top] = (d, S_F)$ with S_F being a terminal state, the pair (d, S_F) is popped out of PS and processing of new stack top resumes.

Table 1: Parameters of IKAI.

Para.	Description
δ_p	Minimum number of complete paths in extracted feature set *per query* as a requirement for the feature set to be qualified as complete and used for training/inference.
δ_{IL}	Interaction limit, i.e., maximum number of questions that IKAI is allowed to ask the user per query.
η_p	Maximum path length for C-PR
η_w	Number of random walks per query for C-PR
l, h	Low and high contextual similarity threshold respectively
ρ	% of learned tasks (relations) selected for additional clue acquisition.

Table 2: State bits (SBs) and their meanings.

SB	Name	Description
QERS	Query entities and relation searched	Whether the query source (s) and target (t) entities and query relation (r) have been searched in KB.
SEF	Source Entity Found	Whether the source entity (s) has been found in KB.
TEF	Target Entity Found	Whether the target entity (t) has been found in KB.
QRF	Query Relation Found	Whether the query relation (r) has been found in KB.
CLUE	Clue bit set	Whether the triple (to be processed) is a clue from user.
ILO	Interaction Limit Over	Whether the interaction limit is over for the query.
PFE	Path Feature Extracted	Whether path feature extraction has been done.
CPF	Complete Path Found	Whether the extracted path features are complete.
INFI	Inference model Invoked	Whether inference model should be invoked.

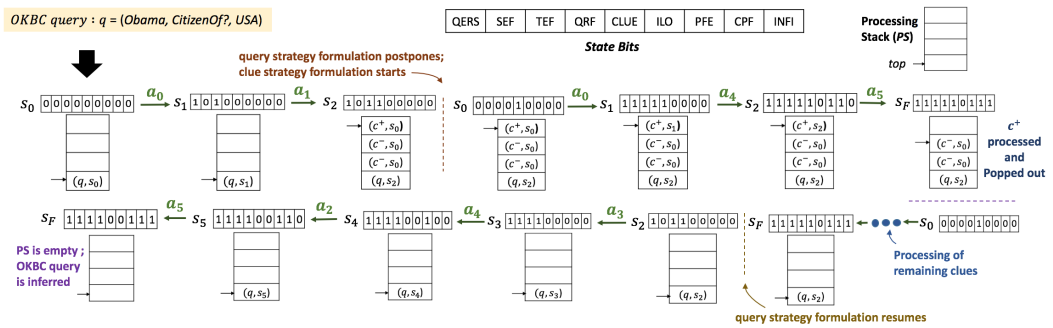

Figure 2: IKAI state transitions for the example in Fig. 1 [q is OKBC query and c^+ (c^-) is +ve (-ve) clue]

Table 3: Transition Conditions and corresponding Actions and their descriptions.

State Transition Conditions (for current state bits S_j [.])	**Action Id : Operation**
$QERS = 0$	a_0 : Search source (s), target (t) entities and query relation (r) in KB.
$ILO = 0 \wedge CLUE = 0 \wedge QERS = 1 \wedge QRF = 0$	a_1 : Ask user to provide an clue/example for query relation r.
$PFE = 1 \wedge ILO = 0 \wedge CPF = 0$	a_2 : Ask user to provide a missing link for path feature completion.
$QERS = 1 \wedge (SEF = 0 \vee TEF = 0) \wedge ILO = 0$	a_3 : Ask user to provide a connecting link to add a new entity to KB.
$QERS = 1 \wedge PFE = 0 \wedge SEF = 1 \wedge TFE = 1$	a_4 : Extract path features between source (s) and target (t) entities using C-PR.
$QRF = 1 \wedge CPF = 1$	a_5 : Invoke prediction model for inference.

Since clues are received from user after the OKBC query (q) is issued in a given dialogue session, they always become stack top and get processed first, postponing the processing of q.

States ($\mathcal{S}$): A state $S \in \mathcal{S}$ is defined by 9 bits (binary variables) (Table 2), each of which keeps track of the results of an action $a \in \mathcal{A}$ taken by IKAI and thus, records the progress of the inference. For example, $S_0 \in \mathcal{S}_0$ is the initial state with all state bits set as 0. Similarly, if $S_i[QERS] = 1$ and other bits are all 0, it denotes that IKAI has searched the KB to check for the existence of query entities (s or t) and relation (r), and found all of them as unknown. If d is a clue, the CLUE bit of S_0 is set as 1.

Actions ($\mathcal{A}$): There are six actions in IKAI: $\{a_0, \ldots, a_5\}$. They function to convert an OKBC query into a KBC query and also, enumerate relation path features for training the Predictor. We discuss the actions one by one below. Fig. 2 shows the state transitions and actions executed at various FSM states for processing the example conversation in Fig. 1 along with the changes in PS. In Fig. 2, the state bits are shown next to each state ID S_i and the content of PS is shown below S_i.

• a_0 **- Search query entities and relation.** This action searches for whether s, r, and t of d are already in the KB and then, sets appropriate bits of the current state (Table 2) and as a result, the FSM moves to a new state. For example, in Fig. 2, IKAI executes a_0 at S_0 [at $PS[top] = (q, S_0)$] and detects that entity "*Obama*" and relation "*CitizenOf*" are *unknown* and the FSM transits to S_1 [$PS[top] = (q, S_1)$]. If d is a clue and $s, t \in$ KB, a_0 also updates the KB with d and its inverse triple (t, r^{-1}, s). E.g., (*UK*, *CitizenOf*$^{-1}$, *David Cameron*) is the inverse of triple (*David Cameron*, *CitizenOf*, *UK*). Adding inverse triple helps path enumeration to collect more path features [30].

• a_1 **- Ask for a clue.** This action asks the user to provide a clue (+ve example) c^+ for an unknown relation r. For example, in Fig. 1, a_1 is executed to acquire clue (SF1) for unknown relation "*CitizenOf*" from the user, who answers (*David Cameron*, *CitizenOf*, *UK*). For learning the predictive model for unseen query relation r, a_1 also corrupts s and t of the clue triple c^+ once at a time to generate -ve clue triple(s) c^- by randomly sampling a node from the KB and pairing it with either s or t, [e.g., (*David Cameron*, *CitizenOf*, *Microsoft*) and (*Chicago*, *CitizenOf*, *UK*)] such that the generated corrupted triples c^-(s) do not exist in the current KB.

Considering the Fig. 1 example and state transitions in Fig. 2, after executing a_1 for query q, IKAI receives the +ve clue triple c^+ (*David Cameron*, *CitizenOf*, *UK*) and first pushes the generated two -ve clues (c^-, S_0) pairs and then, (c^+, S_0) into PS. Now, c^+ being $PS[top]$, strategy formulation for q postpones and that for c^+ begins. The strategy formulation for query q again resumes from "$\langle a_0, a_1 \rangle$" [at $PS[top] = (q, S_2)$] when all (+ve as well as -ve) clues finish their processing and consecutively popped out of the stack. Considering Fig. 1, the formulated strategy for each clue triple here is "$\langle a_0, a_4, a_5 \rangle$" (assuming both *David Cameron* and *UK* are already in the KB).

• a_2 **- Ask missing link query (MLQ).** This action formulates a *missing link query* (MLQ) to ask the user. That is, IKAI selects an incomplete path $\widetilde{p}$ (see Sec. 3, Task 2) from the extracted path set (by action a_4) with the highest context similarity (computed using the same embedding based method as used in C-PR (discussed in Sec. 3, Task 2) $sim(e_f, e_b)$, where e_f and e_b are the two entities

directly connected by -?- (missing link marker). A high contextual similarity indicates that e_f and e_b are more likely to possess a relation [30] and so, is a good candidate for generating a meaningful MLQ. Note, we don't want to ask the user too many questions by processing all incomplete paths.

Considering the Figure 1 example, let IKAI choose an incomplete path "$Obama - BornIn \rightarrow Honolulu - ? - Hawaii - StateOf \rightarrow USA$" from the extracted path set (see action a_4) and with missing link ("-?-") between entity pair ($Honolulu$, $Hawaii$). Then, this missing link is used to formulate a MLQ: "*Can you also tell me how "Honolulu" and "Hawaii" are related?*". On receiving the answer from the user, i.e., (*Honolulu*, *CapitalOfState*, *Hawaii*) - "*Honolulu is the state capital of Hawaii*", IKAI fills the missing link in the path as "$Obama - BornIn \rightarrow Honolulu - CapitalOfState \rightarrow Hawaii - StateOf \rightarrow USA$". This complete path is then used to extract path feature as "$\overline{BornIn \rightarrow CapitalOfState \rightarrow StateOf}$" to be used by the prediction model for inference. In Fig. 2, a_2 for query q is executed at state S_4 [i.e., when $PS[top] = (q, S_4)$]].

If the user does not respond to MLQ, a ***guessing mechanism*** is employed. Since entity pairs with high contextual similarity are likely to possess a relation, IKAI divides the similarity range [-1, 1] into three segments, using a low (l) and a high (h) similarity thresholds to replace the missing link in $\widetilde{p}$ with r_g to make it **complete** as follows- given an entity pair (s', t'), if $h \geq sim(s', t') \geq l$, r_g="*-LooselyRelatedTo-*"; else if $sim(s', t') \leq l$, r_g ="*-NotRelatedTo-*"; Otherwise, r_g="*-RelatedTo-*".

• a_3 **- Ask connecting link query (CLQ).** This action asks the user to connect unknown entities s and/or t with the KB by selecting the most contextually relevant node/entity (wrt s or t) from the KB and asking the user to provide a relation/link for s (or r) and the node. The contextual similarity of two entities is computed based on the same idea as in [30] using neural word embedding [33] and cosine similarity. In Fig. 1, IKAI asks CLQ for "*Obama*" with the chosen contextually relevant entity "*Honolulu*" and acquires SF2. In Fig. 2, a_3 for query q is executed at state S_2 [i.e., at $PS[top] = (q, S_2)$]] when the strategy formulation of q resumes after all clue triples finish their processing.

• a_4 **- Enumerate path features.** This action extracts path feature set P between (s, t) for training the Predictor. P involves both *complete* and *incomplete* paths. However, IKAI always trains the Predictor with complete path features $P_c \subseteq P$. The incomplete paths (see Sec. 3, Task 2) corresponding to the path features in $(P - P_c)$ are used as candidates for formulating and asking MLQs (see a_2) until $|P_c| = \delta_p$ (see Table 1) or δ_{IL} decrements to 0. δ_{IL} denotes the maximum number of questions that IKAI can ask the user per OKBC query and is set to limit the number of interactions with user. In Fig. 1, a_4 for query q gets executed after a_0, a_1, a_3 [i.e., at $PS[top] = (q, S_3)$ in Fig. 2].

• a_5 **- Invoke Predictor.** This action first checks whether the OKBC query has been reduced to a KBC one. The Predictor can only work, *iff* (1) all r, s and t exist in the KB (OKBC is reduced to KBC), and (2) features (relation sequences corresponding to each path in P_c) have been extracted between s and t for building the predictor. If the conditions are met, IKAI adds d (the KBC query triple or the clues) along with P_c in a data buffer $\mathcal{D}$ and invokes the Predictor for either training (on clue triples) or inference/prediction (on the reduced KBC query) by executing a_5. Note that, as clues involving query relation r get processed before the OKBC query, when the predictor is invoked as part of the inference strategy for the query (involving r), the predictor is already trained to perform inference over query triples involving r. In Fig. 2, a_5 for query q gets executed at $PS[top] = (q, S_5)$.

Considering the Figure 1 example, the formulated inference strategy by FSM for *OKBC query* q (*Obama*, *CitizenOf?*, *USA*) is: "$\langle a_0, a_1, a_3, a_4, a_2, a_5 \rangle$" [see Fig. 2].

State Transition Function (Δ): The transition function Δ is characterized by a set of unique state transition conditions specified in terms of state bits. Table 3 gives all the state transition conditions. Given a state S_i, if S_i satisfies a transition condition C, an action corresponding to C is fired. For example, if in S_i, $ILO = 0$, $CLUE = 0$, $QERS = 1$, $QRF = 0$, action a_1 is invoked and on executing a_1, bits in S_i get updated and state transition occurs. Note that, following these conditions in Table 3, only a_2 can repeat up to δ_{IL} (see Table 1) number of times. It is not meaningful to execute any other action more than once for each d (OKBC query or clue from the user) in a given trail of state transitions from the initial state to a terminal state. Thus, if the current state of the FSM remains the same for δ_{IL} times, it means that inference is in-feasible for the current query (e.g., when no complete path feature is found) and the query remains unanswered.

3.2 IKAI Relation Prediction Model

Inferring the correctness of the resulting KBC query is performed by the relation Predictor (a neural network). It learns a single model ($\mathcal{F}$) with trainable parameters Θ for all relations. The model is

trained continuously with streaming clue triples. Note, the triples (in the KB) acquired via CLQs (by a_3) and MLQs (by a_2) in past user interactions are also used as training examples, besides the clue triples obtained in the current interactive session with the user.

Given a relation r, IKAI uses the clue triples involving r (including the generated negative examples for them) stored in data buffer $\mathcal{D}$ to train $\mathcal{F}$. When a query involving r is encountered at time j, IKAI randomly samples k examples of r from $\mathcal{D}$ (we sample max. 500 examples per class label to train $\mathcal{F}$ with a very small number of training epochs (to ensure real-time training of $\mathcal{F}$). For an already learned relation, this is like a quick recollection of the past learned knowledge before the query is answered. For a new r, only the acquired clues of r take part in training, where the learned weights Θ_{j-1} of $\mathcal{F}$ at time $j-1$ acts as the prior knowledge for learning Θ_j.

Often for a (new) relation r learned recently, the number of clue triples acquired in $\mathcal{D}$ may not be sufficient to learn r. Thus, to improve $\mathcal{F}$ continuously on *poorly learned tasks* (*relations*), IKAI acquires additional clues for such r. Here, a *poorly learned task* denotes the task (relation) for which the validation performance of the *Predictor* at a given point in time is low compared to that on other tasks, as measured by an evaluation metric. In our work, we use macro-F1 score to judge the overall learning performance of $\mathcal{F}$ on a given task. IKAI ranks all learned tasks based on validation macro-F1 scores and chooses $\rho\%$ (see Table 1) relations with lowest macro-F1 to acquire more clues for them. In this process, while executing a_0 (see Sec. 3.1), besides the search operation, if the query relation r is evaluated as a poorly learned task, IKAI randomly sets $QRF = 0$ to fire a_1 for clue acquisition.

Our Predictor is based on the compositional vector space model [36, 13]. The model composes the implication of a relation path (feature) using RNN that takes as inputs the vector embeddings of relations in the relation path and outputs a vector in the semantic neighborhood of the test (i.e. query) relation. For a training instance $i \in \mathcal{D}$, IKAI uses a LSTM [20] to compose the vector representation of each path $p \in P_c$ as v_p and vector representation of r as v_r. Next, IKAI computes the prediction value, $\mathbb{P}(r|s,t)$ as sigmoid of the mean cosine similarity of all v_p with v_r: $\mathbb{P}(r|s,t) = sigmoid(\frac{1}{|P_c|}\sum_{p \in P_c} cos(v_r, v_p))$. $\mathcal{F}$ is trained by maximizing the log-likelihood. While predicting a query q involving r, the predictor infers q as +ve (true) if $\mathbb{P}(r|h,t) \geq 0.5$ and -ve (not true) otherwise.

4 Experiments

We evaluate IKAI in terms of its predictive performance and strategy formulation ability. IKAI is designed for a *multi-user* chatting environment. Thus, it is natural for IKAI to observe many query triples (and hence, accumulate more clues) for a relation from different users over time. Presently IKAI only adds the supporting facts into the KB. The predicted query triples are not added as they are unverified knowledge. In practice, IKAI can store these predicted triples in the KB as well after *cross-verification* by asking other users later. Similar cross-verification strategy can also be used to deal with (intentional or unintentional) incorrect knowledge injection by user to the system's KB.

4.1 Evaluation Setup

Evaluating IKAI with real users in a crowd-source environment would be prohibitively time-consuming because both training and testing need a large number of real-time interactions between users and the system. Thus, we design a simulated interactive environment for the IKAI evaluation.

In the simulated environment, a **simulated user** (a program) is created to interact with **IKAI**. The (simulated) user has (1) a ***knowledge base*** ($\mathcal{K}_u$) for answering questions from IKAI, and (2) an ***OKBC query dataset*** (D_q) from which the user issues queries to IKAI. Here, D_q consists of a set of structured OKBC query triples q of the form (s, $r?$, t) readable by IKAI. In practice, the user only issues OKBC queries to IKAI, but cannot evaluate the performance of the system unless the user knows the answer. To evaluate the performance of IKAI on D_q, we also label D_q while preparing the dataset (see Supplementary).

Table 4: Data statistics [*kwn*: known, *unk*: unknown, rel: relation]

KB Statistics	Freebase (FB)	WordNet (WN)
# Relations ($\mathcal{K}_{org}$ / $\mathcal{K}_b$)	1,345 / 1,273	18 / 12
# Entities ($\mathcal{K}_{org}$ / $\mathcal{K}_b$)	13, 871 / 13, 223	13, 595 / 13, 150
# Triples ($\mathcal{K}_{org}$ / $\mathcal{K}_b$)	854, 362 / 652, 790	107, 146 / 66, 338
# Test relations (*kwn* / *unk*)	25 (17 / 8)	18 (12 / 6)
# Train / valid / test instances	11,260 / 1223 / 7083	6628 / 711 / 3500
Entity statistics in Test query triples [s = source entity; t = target entity]		
Avg. % triples per rel with **only** s ***unk***	16.28	13.29
Avg. % triples per rel with **only** t ***unk***	16.29	10.94
Avg. % triples per rel **both** s **and** t ***unk***	4.69	4.17

Table 5: Comparison of predictive performance of *various versions* of IKAI .

Dataset	Models	Rel - K / Ent -K		Rel - K / Ent -UNK		Rel - UNK / Ent - K		Rel - UNK / Ent -UNK		Overall	
		F1(+)	Macro-F1	F1(+)	Macro-F1	F1(+)	Macro-F1	F1(+)	Macro-F1	F1(+)	Macro-F1
Freebase	BG	0.584	0.629	0.494	0.569	0.432	0.532	0.388	0.501	0.508	0.579
	w/o PTL	0.555	0.652	**0.533**	**0.620**	**0.528**	0.419	**0.525**	0.418	**0.538**	0.584
	IKAI	**0.587**	**0.671**	0.493	0.591	0.525	**0.616**	0.440	**0.577**	0.532	**0.627**
WordNet	BG	0.548	0.466	0.532	0.525	0.486	0.476	0.498	0.484	0.526	0.482
	w/o PTL	**0.666**	**0.741**	**0.561**	**0.624**	0.461	0.281	0.485	0.323	0.556	0.588
	IKAI	0.655	0.694	0.552	0.604	**0.612**	**0.659**	**0.509**	**0.506**	**0.612**	**0.653**

As IKAI performs continuous online knowledge acquisition and learning, we evaluate its performance over time at different temporal checkpoints. We assume that, IKAI has already interacted with some users up to the evaluation start time t_{eval} and gathered a knowledge base ($\mathcal{K}_b$) (called the **base KB**), which includes its initial knowledge base and the knowledge obtained in its past interactions with users. Apart from $\mathcal{K}_b$, we also assume that at t_{eval}, IKAI's predictor model has already been trained to perform inference over a set of query relations seen so far. Recall our single predictor covers all past learned relations. We call these relations the *known* query relations. The query relations in dataset D_q not belonging to the set of *known* query relations are called *unknown* query relations. Note, D_q consists of query triples involving both *known* and *unknown* query relations.

Evaluation KB Datasets. We create the simulated user's KB ($\mathcal{K}_u$), IKAI's base KB ($\mathcal{K}_b$) and the OKBC query dataset (D_q) along with the training and validation datasets (used in IKAI's *initial training phase*) from two standard KB datasets (see Table 4): (1) Freebase FB15k and (2) WordNet WN18 [9]. From each KB dataset, we build a large knowledge graph and use it as the original KB ($\mathcal{K}_{org}$). We also augment $\mathcal{K}_{org}$ with inverse triples (t, r^{-1}, s) for each (s, r, t) in $\mathcal{K}_{org}$ to increase graph connectivity following the existing KBC methods. Here, r^{-1} refers to the inverse relation of r that connects t to s (see "Actions" in Sec. 3.1). We provide details about the creation of simulated user's KB ($\mathcal{K}_u$), base KB ($\mathcal{K}_b$) and the OKBC query dataset (D_q) from $\mathcal{K}_{org}$ and Hyper-parameter Settings in Supplementary Material.

Compared Models. Since no existing KBC method can solve the proposed OKBC problem, we compare various versions of IKAI: **(1) IKAI**: The proposed full version of IKAI.
(2) BG (Blind Guessing). To evaluate the guessing mechanism (see action a_2 in Sec. 3.1), BG fills all the missing or connecting links blindly with "*-RelatedTo-*" (the user has no answer). Note that, if no guessing is used and the user does not answer an OKBC query, IKAI will reject the query.
(3) w/o PTL (without Past Task Learning): IKAI does not ask for additional clues for poorly learned tasks and retrain $\mathcal{F}$ on past learned tasks. Only training of $\mathcal{F}$ on new tasks is enabled.

Note that although [45, 44] deal with OKBC, they infer facts using a text corpus, while we do through user interactions. The two approaches are complementary and not comparable.

Evaluation Metrics. We use positive (+) F1 score and Macro F1 score for evaluation. To evaluate the correctness of strategy formulation, we introduce *Coverage* (C), the fraction of total query triples answered by IKAI. Note, C depends on the parameter δ_{IL} and δ_p.

4.2 Results and Analysis

For evaluation on a given KB (Freebase or WordNet), we first train IKAI using training and validation instances (see Table 4) from base KB $\mathcal{K}_b$ so that we get initial relation prediction model for all known tasks (relations). Once this *initial training phase* is over, we randomly generate a chronological ordering of all test instances (OKBC queries) in D_q, which are fed to IKAI one by one in a streaming fashion, and then, evaluate IKAI online. The **avg. test query processing time** of IKAI for Freebase is 0.24 sec and for WordNet is 0.15 sec (on a Nvidia Titan RTX GPU).

Predictive Performance. Table 5 shows the +ve F1 and Macro F1 scores of various versions of IKAI. We present the overall performance on the whole test query dataset as well as performance on subsets of test query dataset, denoted as (*Rel-X, Ent-Y*), where X and Y can be either *known* ('K') or *unknown* ('UNK') and '*Rel*' denotes query relation and '*Ent*' denotes query entities. So, here, (*Rel-K, Ent-UNK*) denotes the subset of the test dataset having query triples involving known relations, but unknown entities (either s or t is unknown) with respect to the base KB $\mathcal{K}_b$. From Table 5, IKAI performs the best among all variants on both KB datasets. Due to the use of contextual similarity of entity-pairs, IKAI's guessing mechanism also works better than blind guessing (BG). The past task selection and learning mechanism of IKAI also improves its performance over that for w/o PTL, as it acquires and learns more clues during testing for poorly performed tasks (evaluated on validation set).

Table 6 shows the result of IKAI performance improvement over time at various temporal checkpoints for **known (kwn)** [corresponds to (*Rel-K, Ent-K*)], **unknown (unk)** [corresponds to queries for which either Rel-UNK or Ent-UNK] and **overall** queries. We noted in our experiments that, the simulated user's query satisfaction rate (proportions of queries that can be answered by the simulated user) is very low (1% MLQs and 11% CLQs). Thus, to fairly evaluate the performance improvement of IKAI over time, we report the results (in Table 6) in *filtered setting*, where we only consider test queries for which the user is able to answer (provide supporting facts) at least one CLQ or MLQ (when asked by IKAI). From Table 6, we see that the IKAI performance improves significantly as it see more test queries, and gain more knowledge (whenever user is able to respond) over time.

Table 6: Performance (+ve-F1 scores) of IKAI on (filtered) test queries observed over time [% TTO = % of Test Triples Observed]. Number in [] shows the test triples count for TTO = 100%. "all" denotes overall performance.

% TTO	Freebase			WordNet		
	kwn [9]	*unk* [293]	*all* [302]	*kwn* [21]	*unk* [105]	*all* [126]
50%	0.0	0.492	0.507	0.947	0.799	0.819
100%	**0.545**	**0.580**	**0.578**	**0.950**	**0.870**	**0.884**

Table 7: Performance (+ve F1) of IKAI on user's responses to CLQs and MLQs evaluated on Full (100%) test data.

Response	Freebase			WordNet		
	kwn	*unk*	*Overall*	*kwn*	*unk*	*overall*
No	**0.594**	0.483	0.529	**0.664**	0.564	**0.612**
Yes	0.587	**0.496**	**0.532**	0.655	**0.571**	**0.612**

Table 7 shows the results of IKAI on user responses to MLQ's and CLQ's on the full test data. Answering MLQ's and CLQ's is hard for the simulated user as $\mathcal{K}_u$ often lacks the required triples. Thus, we analyze how the performance is affected if the user does not respond at all. The results show a trend in overall performance gain when the user responds (also, supported by Table 6). However, the improvement is not large as the simulated user's query satisfaction rate (1% MLQs and 11% CLQs) is very small. But, the analysis shows the effectiveness of IKAI's *guessing mechanism* despite such minimal knowledge acquisition by IKAI (see performance gain of IKAI over BG in Table 5).

Inference Strategy Formulation. We use WordNet test query dataset to illustrate various inference strategies formulated by IKAI (see Table 8) with varying δ_{IL} and δ_p values (column 2 follows collomn 1 in Table 8). When $\delta_{IL} = 0$, IKAI cannot interact with user and thus, can only solve KBC queries from user and rejects all queries involving unknown s, r or t. Thus, coverage C drops significantly (0.5). The only formulated strategy here is $\langle a_0, a_4, a_5 \rangle$, which involves only processing actions. None of the interactive actions a_1, a_2, a_3 was invoked by the FSM here.

When $\delta_{IL} = 1$, IKAI can only interact once with the user per query for knowledge acquisition. In such a case, IKAI acquires knowledge well for instances where either of an entity or relation is unknown. However, as one unknown entity or relation may appear in multiple future test triples, once it becomes known, IKAI doesn't need to ask for it again (knowledge accumulation). Thus, C increases significantly (0.97). Each of the formulated strategies (4 strategies in Table 8) has only one interactive action [one of $\{a_1, a_2, a_3\}$].

Table 8: IKAI's Inference strategies (ordered by frequency).

$\delta_{IL} = 0, \delta_p = 3$ **[C: 0.50]**	$\langle a_0, a_4, a_2, a_2, a_5 \rangle$	$\delta_{IL} = 5, \delta_p = 1$ **[C: 1.0]**
$\langle a_0, a_4, a_5 \rangle$	$\langle a_0, a_4, a_2, a_5 \rangle$	$\langle a_0, a_4, a_2, a_5 \rangle$
$\delta_{IL} = 1, \delta_p = 3$ **[C: 0.97]**	$\langle a_0, a_4, a_5 \rangle$	$\langle a_0, a_4, a_5 \rangle$
$\langle a_0, a_4, a_2, a_5 \rangle$	$\langle a_0, a_3, a_4, a_2, a_2, a_5 \rangle$	$\langle a_0, a_3, a_4, a_2, a_5 \rangle$
$\langle a_0, a_1, a_4, a_5 \rangle$	$\langle a_0, a_3, a_4, a_2, a_5 \rangle$	$\langle a_0, a_3, a_4, a_5 \rangle$
$\langle a_0, a_3, a_4, a_5 \rangle$	$\langle a_0, a_3, a_4, a_5 \rangle$	$\langle a_0, a_1, a_3, a_4, a_2, a_5 \rangle$
$\langle a_0, a_4, a_5 \rangle$	$\langle a_0, a_1, a_3, a_4, a_2, a_5 \rangle$	$\langle a_0, a_1, a_4, a_5 \rangle$
$\delta_{IL} = 3, \delta_p = 3$ **[C: 1.0]**	$\langle a_0, a_1, a_4, a_2, a_2, a_5 \rangle$	$\langle a_0, a_1, a_3, a_4, a_5 \rangle$
$\langle a_0, a_4, a_2, a_2, a_2, a_5 \rangle$	$\langle a_0, a_1, a_4, a_2, a_5 \rangle$	$\langle a_0, a_1, a_4, a_2, a_5 \rangle$

When $\delta_{IL} = 3$, IKAI is able to perform inference on all test queries and coverage C becomes 1. For $\delta_p = 1$, IKAI uses a_2 only once (as only one MLQ satisfies δ_p) compared to at most 3 MLQs in formulated strategies for $\delta_p = 3$. In summary, IKAI's FSM module can correctly formulate query-specific inference strategies based on specified parameter values.

5 Conclusions

This paper proposed to solve the *open-world knowledge base completion* (OKBC) problem through user interactions and inference. The work is motivated by the need to learn new knowledge in human-machine dialogues. IKAI (and together with our earlier work CILK [28]) can potentially serve as a chatbot knowledge learning engine. Our experimental results demonstrated both IKAI's predictive quality and strategy formulation ability. In our future work, we plan to learn other forms of knowledge to make learning during conversation more complete. It is also critical to implement this learning capability in a practice chatbot to test its validity and to improve it in the process.

Acknowledgments

This work was supported in part by a research gift from Northrop Grumman, a research contract from DARPA (HR001120C0023), and two research grants from National Science Foundation (IIS-1910424 and IIS-1838770).

References

[1] Vito Walter Anelli, Pierpaolo Basile, Derek Bridge, Tommaso Di Noia, Pasquale Lops, Cataldo Musto, Fedelucio Narducci, and Markus Zanker. Knowledge-aware and conversational recommender systems. In *ACM RecSys*, 2018.

[2] Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D Manning. Leveraging linguistic structure for open domain information extraction. In *ACL-IJCNLP*, 2015.

[3] Michele Banko, Michael J Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. Open information extraction from the web. In *IJCAI*, 2007.

[4] Junwei Bao, Nan Duan, Zhao Yan, Ming Zhou, and Tiejun Zhao. Constraint-based question answering with knowledge graph. In *COLING*, 2016.

[5] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on freebase from question-answer pairs. In *EMNLP*. 2013.

[6] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*, 2008.

[7] Antoine Bordes, Sumit Chopra, and Jason Weston. Question answering with subgraph embeddings. In *EMNLP*, 2014.

[8] Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*, 2015.

[9] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *NIPS*, 2013.

[10] Antoine Bordes and Jason Weston. Learning end-to-end goal-oriented dialog. *arXiv preprint arXiv:1605.07683*, 2016.

[11] Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. Learning structured embeddings of knowledge bases. In *AAAI*, 2011.

[12] Zhiyuan Chen and Bing Liu. *Lifelong machine learning*. Morgan & Claypool Publishers, 2018.

[13] Rajarshi Das, Arvind Neelakantan, David Belanger, and Andrew McCallum. Chains of reasoning over entities, relations, and text using recurrent neural networks. *arXiv preprint arXiv:1607.01426*, 2016.

[14] Anthony Fader, Stephen Soderland, and Oren Etzioni. Identifying relations for open information extraction. In *EMNLP*, 2011.

[15] Matt Gardner and Jayant Krishnamurthy. Open-vocabulary semantic parsing with both distributional statistics and formal knowledge. In *AAAI*, 2017.

[16] Matt Gardner and Tom M Mitchell. Efficient and expressive knowledge base completion using subgraph feature extraction. In *EMNLP*, 2015.

[17] Matt Gardner, Partha Pratim Talukdar, Jayant Krishnamurthy, and Tom Mitchell. Incorporating vector space similarity in random walk inference over knowledge bases. In *ACL*, 2014.

[18] Marjan Ghazvininejad, Chris Brockett, Ming-Wei Chang, Bill Dolan, Jianfeng Gao, Wen-Tau Yih, and Michel Galley. A knowledge-grounded neural conversation model. In *AAAI*, 2018.

[19] Ben Hixon, Peter Clark, and Hannaneh Hajishirzi. Learning knowledge graphs for question answering through conversational dialog. In *NAACL-HLT*, 2015.

[20] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 1997.

[21] Johannes Hoffart, Fabian M Suchanek, Klaus Berberich, and Gerhard Weikum. Yago2: A spatially and temporally enhanced knowledge base from wikipedia. *Artificial Intelligence*, 2013.

[22] Yoji Kiyota, Sadao Kurohashi, and Fuyuko Kido. Dialog navigator: A question answering system based on large text knowledge base. In *COLING*, 2002.

[23] Ni Lao, Einat Minkov, and William W Cohen. Learning relational features with backward random walks. In *ACL*, 2015.

[24] Ni Lao, Tom Mitchell, and William W Cohen. Random walk inference and learning in a large scale knowledge base. In *EMNLP*, 2011.

[25] Jiwei Li, Alexander H Miller, Sumit Chopra, Marc'Aurelio Ranzato, and Jason Weston. Dialogue learning with human-in-the-loop. In *ICLR*, 2017.

[26] Jiwei Li, Alexander H Miller, Sumit Chopra, Marc'Aurelio Ranzato, and Jason Weston. Learning through dialogue interactions by asking questions. In *ICLR*, 2017.

[27] Bing Liu and Sahisnu Mazumder. Lifelong and continual learning dialogue systems:learning during conversation. In *Proceedings of AAAI-2021*, 2021.

[28] Bing Liu and Chuhe Mei. Lifelong knowledge learning in rule-based dialogue systems. *arXiv preprint arXiv:2011.09811*, 2020.

[29] Yinong Long, Jianan Wang, Zhen Xu, Zongsheng Wang, Baoxun Wang, and Zhuoran Wang. A knowledge enhanced generative conversational service agent. In *DSTC6 Workshop*, 2017.

[30] Sahisnu Mazumder and Bing Liu. Context-aware path ranking for knowledge base completion. In *IJCAI*, 2017.

[31] Sahisnu Mazumder, Bing Liu, Shuai Wang, and Nianzu Ma. Lifelong and interactive learning of factual knowledge in dialogues. In *SIGDIAL*, 2019.

[32] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *In ICLR Workshop*, 2013.

[33] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013.

[34] George A Miller. Wordnet: a lexical database for english. *In Communications of the ACM*, 1995.

[35] Tom M Mitchell, William W Cohen, Partha Pratim Talukdar, Justin Betteridge, Andrew Carlson, Matthew Gardner, Bryan Kisiel, Jayant Krishnamurthy, et al. Never ending learning. In *AAAI*, 2015.

[36] Arvind Neelakantan, Benjamin Roth, and Andrew McCallum. Compositional vector space models for knowledge base completion. In *ACL*, 2015.

[37] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs. *arXiv preprint arXiv:1503.00759*, 2015.

[38] Kohei Ono, Ryu Takeda, Eric Nichols, Mikio Nakano, and Kazunori Komatani. Lexical acquisition through implicit confirmations over multiple dialogues. In *SIGDIAL*, 2017.

[39] Tsugumi Otsuka, Kazunori Komatani, Satoshi Sato, and Mikio Nakano. Generating more specific questions for acquiring attributes of unknown concepts from users. In *SIGDIAL*, 2013.

[40] Chen Qu, Liu Yang, W Bruce Croft, Johanne R Trippas, Yongfeng Zhang, and Minghui Qiu. Analyzing and characterizing user intent in information-seeking conversations. In *SIGIR*, 2018.

[41] Amrita Saha, Vardaan Pahuja, Mitesh M Khapra, Karthik Sankaranarayanan, and Sarath Chandar. Complex sequential question answering: Towards learning to converse over linked question answer pairs with a knowledge graph. In *AAAI*, 2018.

[42] Denis Savenkov and Eugene Agichtein. When a knowledge base is not enough: Question answering over knowledge bases with external text data. In *SIGIR*, 2016.

[43] Michael Schmitz, Robert Bart, Stephen Soderland, Oren Etzioni, et al. Open language learning for information extraction. In *EMNLP and CoNLL*, 2012.

[44] Haseeb Shah, Johannes Villmow, Adrian Ulges, Ulrich Schwanecke, and Faisal Shafait. An open-world extension to knowledge graph completion models. In *AAAI*, 2019.

[45] Baoxu Shi and Tim Weninger. Open-world knowledge graph completion. In *AAAI*, 2018.

[46] Y. Sun and Yi Zhang. Conversational recommender system. In *SIGIR*, 2018.

[47] Patrick Verga, Arvind Neelakantan, and Andrew McCallum. Generalizing to unseen entities and entity pairs with row-less universal schema. In *EACL*, 2017.

[48] Oriol Vinyals and Quoc Le. A neural conversational model. *arXiv preprint arXiv:1506.05869*, 2015.

[49] Quan Wang, Jing Liu, Yuanfei Luo, Bin Wang, and C Lin. Knowledge base completion via coupled path ranking. In *ACL*, 2016.

[50] Sida I Wang, Samuel Ginn, Percy Liang, and Christoper D Manning. Naturalizing a programming language via interactive learning. *ACL*, 2017.

[51] Sida I Wang, Percy Liang, and Christopher D Manning. Learning language games through interaction. *ACL*, 2016.

[52] Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. Knowledge base completion via search-based question answering. In *WWW*, 2014.

[53] Jason E Weston. Dialog-based language learning. In *NIPS*, 2016.

[54] Fei Wu and Daniel S Weld. Open information extraction using wikipedia. In *ACL*, 2010.

[55] Liu Yang, Minghui Qiu, Chen Qu, Jiafeng Guo, Yongfeng Zhang, W Bruce Croft, Jun Huang, and Haiqing Chen. Response ranking with deep matching networks and external knowledge in information-seeking conversation systems. In *SIGIR*, 2018.

[56] Tom Young, Erik Cambria, Iti Chaturvedi, Hao Zhou, Subham Biswas, and Minlie Huang. Augmenting end-to-end dialog systems with commonsense knowledge. In *AAAI*, 2018.

[57] Haichao Zhang, Haonan Yu, and Wei Xu. Listen, interact and talk: Learning to speak via interaction. *arXiv preprint arXiv:1705.09906*, 2017.

[58] Yongfeng Zhang, Xu Chen, Qingyao Ai, Liu Yang, and W Bruce Croft. Towards conversational search and recommendation: System ask, user respond. In *CIKM*, 2018.

[59] Hao Zhou, Tom Young, Minlie Huang, Haizhou Zhao, Jingfang Xu, and Xiaoyan Zhu. Commonsense knowledge aware conversation generation with graph attention. In *IJCAI*, 2018.

Supplementary Material

A Simulated User Creation and Test Dataset Generation

To simulate the inital interactions and knowledge learning process of IKAI up to time step t_{eval}, we create a training and validation dataset (in addition to the test query dataset D_q) involving triples of *known* query relations and entities. We assume that IKAI has already been trained on these training and validation dataset triples (in the past) before the evaluation on D_q starts. We call this training process the ***initial training phase*** of IKAI. Once the initial training phase is over, IKAI is ready to perform the online training (with clues to be acquired) and evaluation process (with D_q) at t_{eval}.

Given the original KB graph ($\mathcal{K}_{org}$) (Freebase or WordNet), we create the simulated user's knowledge base ($\mathcal{K}_u$), the base KB ($\mathcal{K}_b$), the OKBC test query dataset (D_q) and the training and validation datasets for initial training phase from $\mathcal{K}_{org}$ (Table 4 in main paper shows the statistics), as discussed below.

For Freebase, we found 86 relations with ≥ 1000 triples and randomly selected 25 from various domains. We randomly shuffle the list of 25 relations, select 34% of them as *unknown* relations and consider the rest (66%) as *known* relations.

For each *known* relation r, we randomly shuffle the list of distinct triples for r, choose (maximum) 500 triples and split them into 60% training, 10% validation and 20% test. Rest 10% along with the leftover (not included in the list of 500) triples are added to the user knowledge base $\mathcal{K}_u$. The training and validation triples are used in IKAI's initial training phase and the test triples constitute D_q.

For each *unknown* relation r, we remove all triples of r from $\mathcal{K}_{org}$, and then, randomly choose 20% triples among these triples as test queries for unknown r. Rest of the triples (80%) for *unknown* relation r are added to user's KB $\mathcal{K}_u$. In this process, we also make sure that the test query triples involving *unknown* relation r are excluded from $\mathcal{K}_u$. Thus, the user cannot provide the query triple itself as a clue to IKAI (during inference) and also, to simulate the case that the user does not know the answer of its issued query.

Note that, at this point, $\mathcal{K}_u$ has at least 10% of triples for each r (*known* and *unknown* to IKAI) and thus, the user is always able to provide clues for OKBC queries involving both *known* and *unknown* relations. Note, if the user cannot provide a clue for an unknown relation (not likely), the OKBC query cannot be converted to a KBC query and thus, IKAI will not be able to answer the query.

To create queries involving unknown entities, we first randomly choose 10% of the entities present in the test query triples, and then, remove all triples involving those entities from $\mathcal{K}_{org}$ and add them to user's KB $\mathcal{K}_u$. At this point, $\mathcal{K}_{org}$ gets reduced to $\mathcal{K}_b$ (the base KB) for IKAI.

Note that, due to the removal of a significant amount of triples, the base KB $\mathcal{K}_b$ becomes *sparser* (see "# triples" row in Table 4 in main paper), making the inference task much harder. The WordNet dataset being small, we use all its 18 relations for evaluation and then, create the OKBC test query dataset, training and validation dataset, $\mathcal{K}_u$ and $\mathcal{K}_b$ following the above procedure used for Freebase.

Although the simulated user may provide clues 100% of the time (otherwise, the query is not answerable), it often cannot respond to MLQs and CLQs due to lack of the required triples/facts. Thus, we further enrich $\mathcal{K}_u$ with external KB triples. Due to a fair amount of entity overlapping, we choose NELL[2] for enriching $\mathcal{K}_u$ in case of Freebase, and ConceptNet [3] for enriching $\mathcal{K}_u$ in case of WordNet.

B Dataset Labeling

We now discuss labeling of the test query dataset (D_q) and training and validation datasets. Given a relation r and a triple (s, r, t) in the training, validation or test query dataset, each pair (s, t) in them is regarded as a +ve triple (instance) for r. Thus, (s, $r?$, t) represents a +ve OKBC query (i.e., the true answer of the query is "YES"). For each +ve instance (s, t), we generate a negative instance, either by randomly corrupting the source s, or by corrupting the target t (same as in action a_1 in

[2] http://rtw.ml.cmu.edu/rtw/kbbrowser/

[3] http://conceptnet.io/

Sec. 3.1). The corrupted triple corresponding to each +ve triple are treated as -ve OKBC query (i.e., the true answer of the query is "NO") and added to the corresponding dataset (training / validation / test). These datasets are thus ***expanded*** with negative instances.

Note, the test triples are not in $\mathcal{K}_b$ or $\mathcal{K}_u$. Also, each triple/instance in training or testing is represented with its relation path features extracted while executing the inference strategy.

C Hyper-parameter Settings

Unless specified, the *empirically* set parameters (Table 1 in main paper) of IKAI are: $\delta_{IL} = 5$ set for each test query and $\delta_{IL} = 2$ set for each training queries / clues, $\delta_p = 3$, $\eta_\pi = 7$, $\eta_w = 20$, $l = 0.07$, $h = 0.2$, $\rho = 25\%$. For initial training phrase, number of training epochs is set as 10. For online training, training epoch as 2 (3) for learned (unlearned) task for each query processing, batch-size is set as 128, dropout is set as 0.1, hidden units and embedding size are set as 300, learning rate is 1e-4 and regularization parameter 0.001. Adam optimizer is used for optimization.