US 20200242444A1

(19) United States
(12) Patent Application Publication
ZHANG et al.

(10) Pub. No.: US 2020/0242444 A1
(43) Pub. Date: Jul. 30, 2020

(54) KNOWLEDGE-GRAPH-EMBEDDING-BASED QUESTION ANSWERING

(71) Applicant: **Baidu USA, LLC**, Sunnyvale, CA (US)

(72) Inventors: **Jingyuan ZHANG**, San Jose, CA (US); **Dingcheng LI**, Sammamish, WA (US); **Ping LI**, Bellevue, WA (US); **Xiao HUANG**, College Station, TX (US)

(73) Assignee: **Baidu USA LLC**, Sunnyvale, CA (US)

(21) Appl. No.: **16/262,618**

(22) Filed: **Jan. 30, 2019**

Publication Classification

(51) **Int. Cl.**

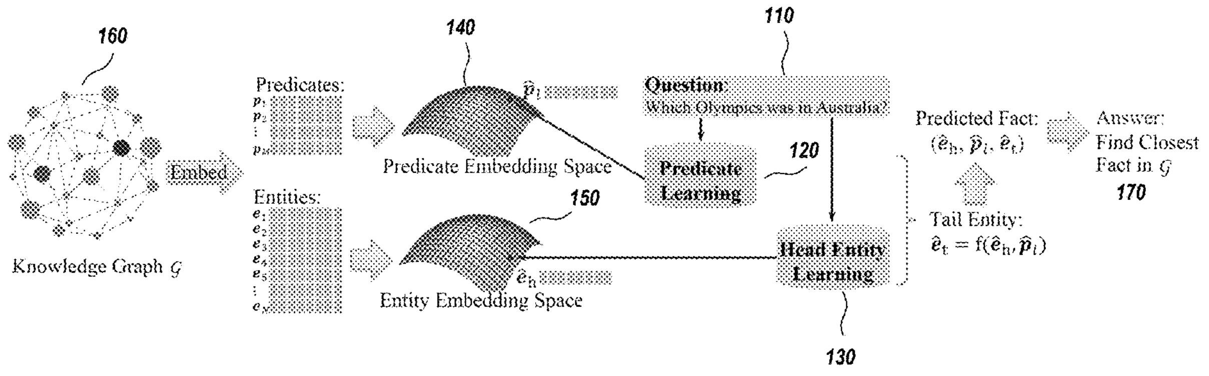
G06N 3/00	(2006.01)
G06F 16/901	(2006.01)
G06F 16/2452	(2006.01)
G06N 3/04	(2006.01)
G06N 3/08	(2006.01)

(52) **U.S. Cl.**
CPC ***G06N 3/006*** (2013.01); ***G06F 16/9024*** (2019.01); ***G06N 3/08*** (2013.01); ***G06N 3/0445*** (2013.01); ***G06N 3/0427*** (2013.01); ***G06F 16/24522*** (2019.01)

(57) **ABSTRACT**

Described herein are embodiments for question answering over knowledge graph using a Knowledge Embedding based Question Answering (KEQA) framework. Instead of inferring an input questions' head entity and predicate directly, KEQA embodiments target jointly recovering the question's head entity, predicate, and tail entity representations in the KG embedding spaces. In embodiments, a joint distance metric incorporating various loss terms is used to measure distances of a predicated fact to all candidate facts. In embodiments, the fact with the minimum distance is returned as the answer. Embodiments of a joint training strategy are also disclosed for better performance. Performance evaluation on various datasets demonstrates the effectiveness of the disclosed systems and methods using the KEQA framework.

100


160
Knowledge Graph $\mathcal{G}$
Embed
Predicates:
p_1 p_2 ⋮ p_M
Entities:
e_1 e_2 e_3 e_4 e_5 ⋮ e_N
140
$\hat{p}_l$
Predicate Embedding Space
150
$\hat{e}_h$
Entity Embedding Space
110
Question:
Which Olympics was in Australia?
Predicate Learning
120
Head Entity Learning
130
Tail Entity:
$\hat{e}_t = f(\hat{e}_h, \hat{p}_l)$
Predicted Fact:
$(\hat{e}_h, \hat{p}_l, \hat{e}_t)$
Answer:
Find Closest Fact in $\mathcal{G}$
170

100

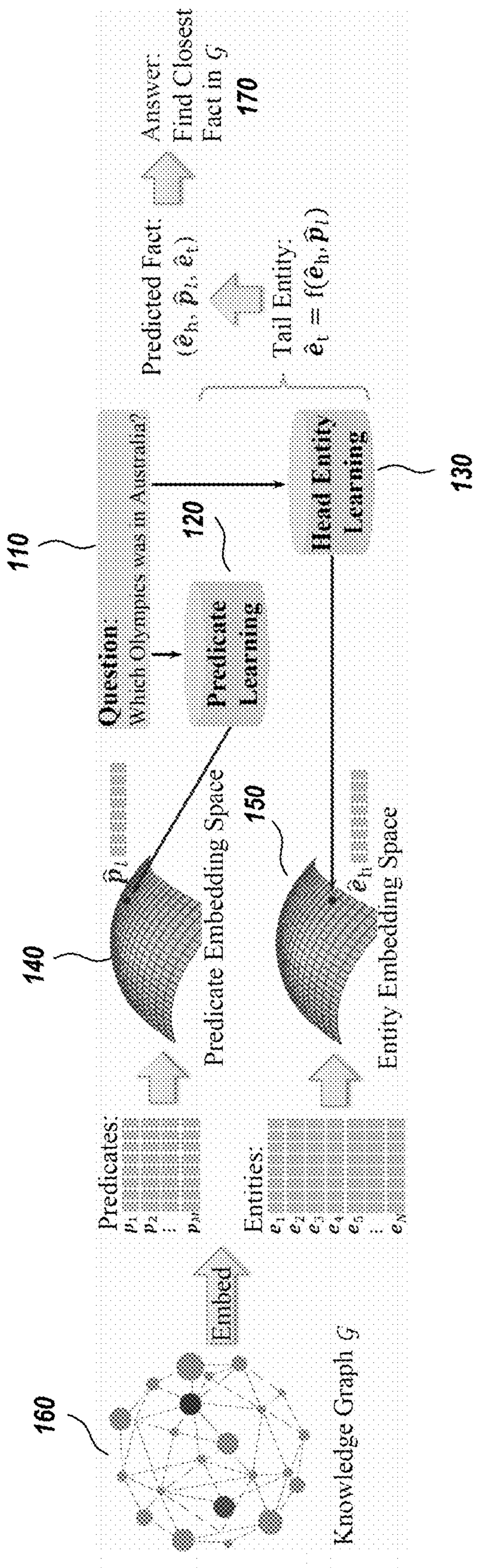

Knowledge Graph $\mathcal{G}$
160
Embed
Predicates:
p_1
p_2
$\vdots$
p_M
Entities:
e_1
e_2
e_3
e_4
e_5
$\vdots$
e_N
140
$\hat{p}_l$
Predicate Embedding Space
150
$\hat{e}_h$
Entity Embedding Space
110
Question
Which Olympics was in Australia?
120
Predicate Learning
Head Entity Learning
130
Predicted Fact:
$(\hat{e}_h, \hat{p}_l, \hat{e}_t)$
Tail Entity:
$\hat{e}_t = f(\hat{e}_h, \hat{p}_l)$
Answer:
Find Closest Fact in $\mathcal{G}$
170

FIG. 1

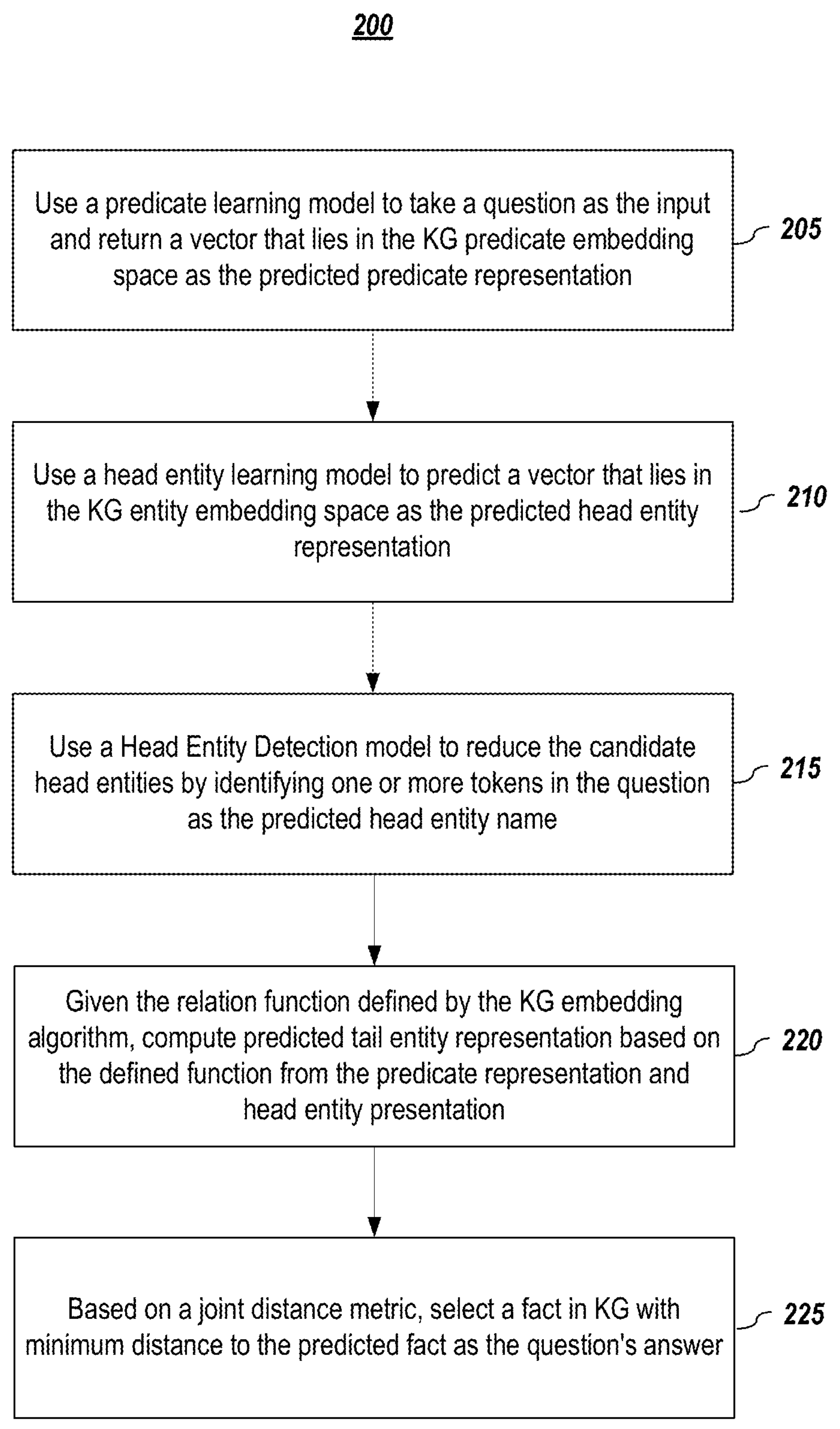

200
Use a predicate learning model to take a question as the input and return a vector that lies in the KG predicate embedding space as the predicted predicate representation
205
Use a head entity learning model to predict a vector that lies in the KG entity embedding space as the predicted head entity representation
210
Use a Head Entity Detection model to reduce the candidate head entities by identifying one or more tokens in the question as the predicted head entity name
215
Given the relation function defined by the KG embedding algorithm, compute predicted tail entity representation based on the defined function from the predicate representation and head entity presentation
220
Based on a joint distance metric, select a fact in KG with minimum distance to the predicted fact as the question's answer
225

FIG. 2

300

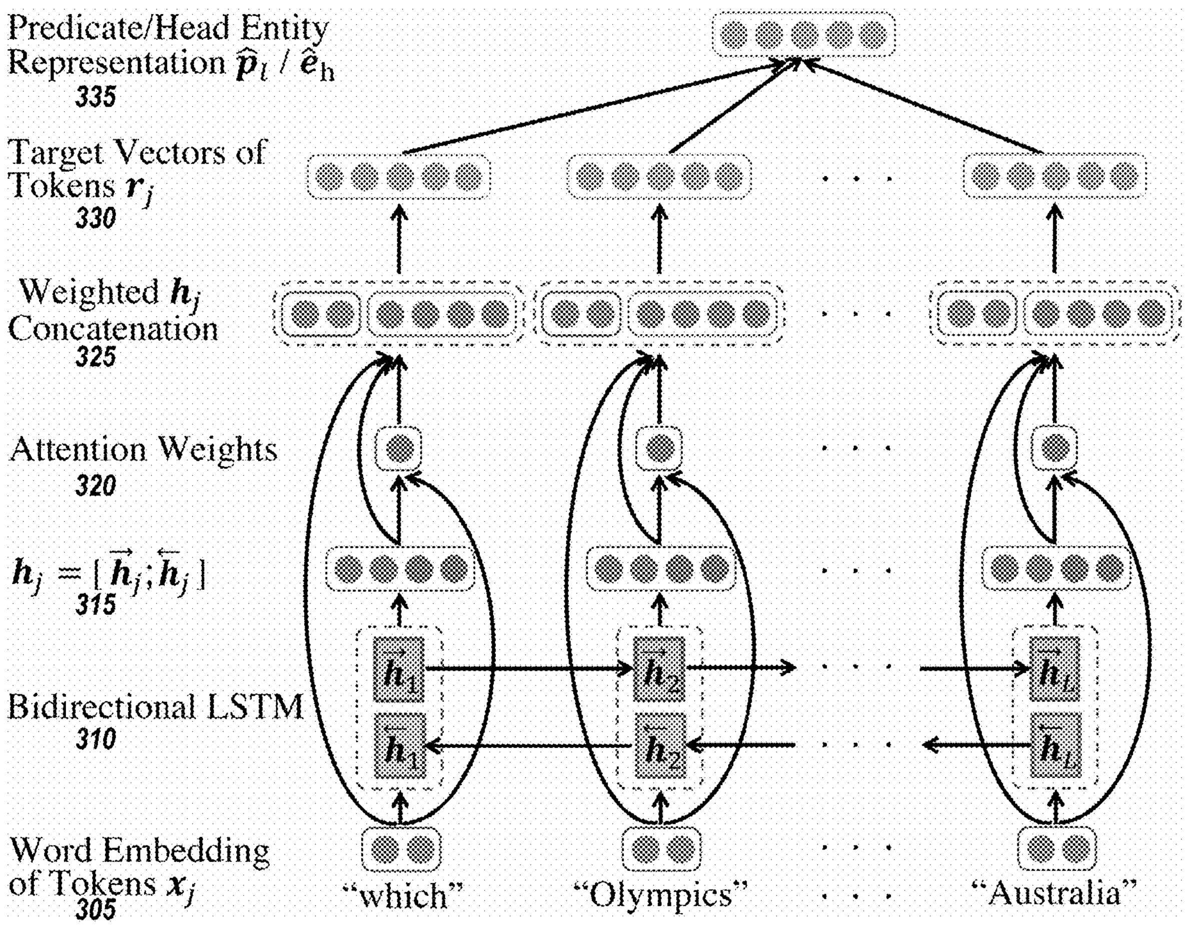

Predicate/Head Entity Representation $\hat{p}_l$ / $\hat{e}_h$
335
Target Vectors of Tokens r_j
330
Weighted h_j Concatenation
325
Attention Weights
320
$h_j = [\overrightarrow{h}_j; \overleftarrow{h}_j]$
315
Bidirectional LSTM
310
Word Embedding of Tokens x_j
305
$\overrightarrow{h}_1$
$\overleftarrow{h}_1$
$\overrightarrow{h}_2$
$\overleftarrow{h}_2$
$\overrightarrow{h}_L$
$\overleftarrow{h}_L$
"which"
"Olympics"
"Australia"

FIG. 3

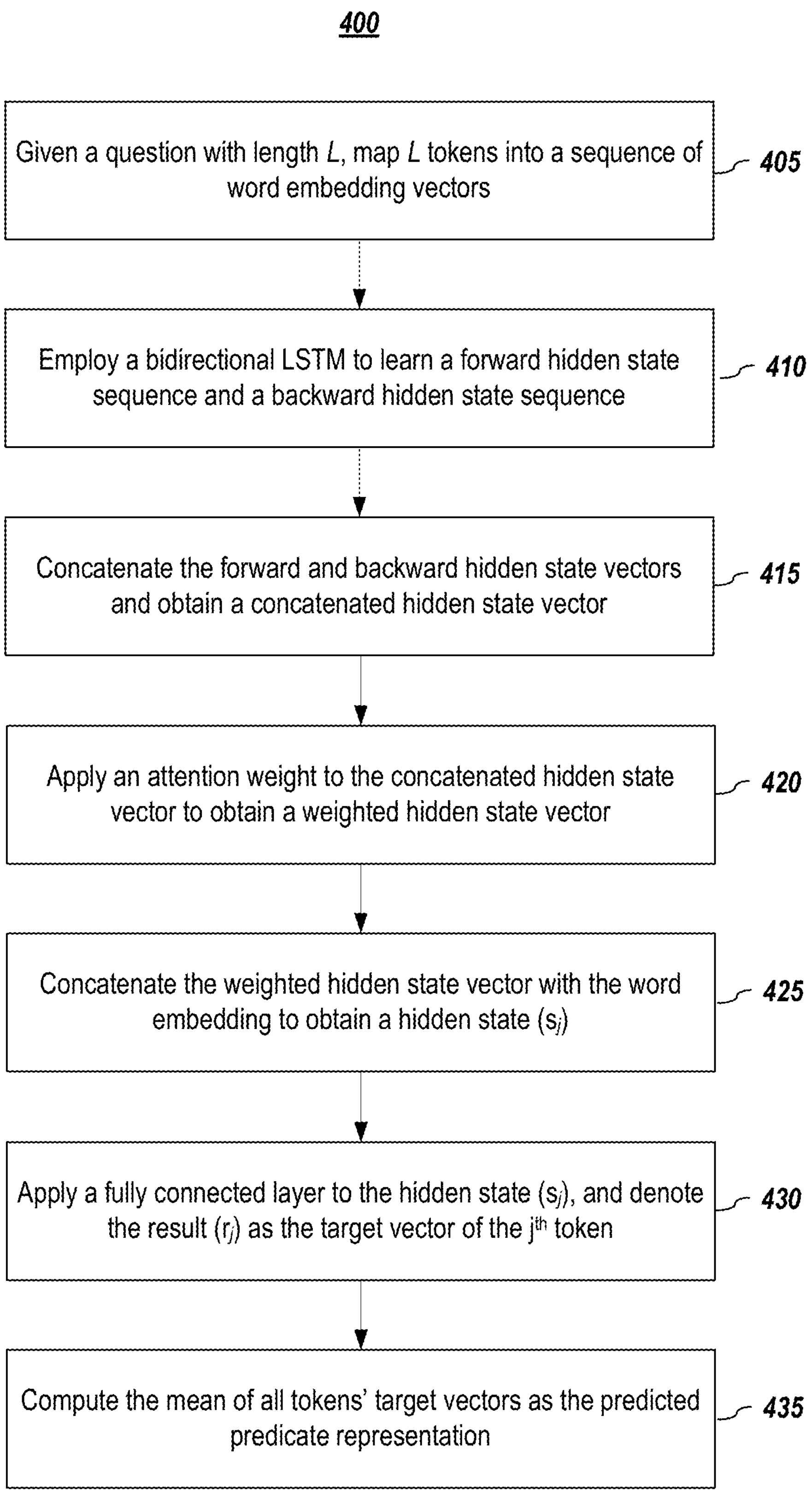

400
Given a question with length L, map L tokens into a sequence of word embedding vectors
405
Employ a bidirectional LSTM to learn a forward hidden state sequence and a backward hidden state sequence
410
Concatenate the forward and backward hidden state vectors and obtain a concatenated hidden state vector
415
Apply an attention weight to the concatenated hidden state vector to obtain a weighted hidden state vector
420
Concatenate the weighted hidden state vector with the word embedding to obtain a hidden state (s_j)
425
Apply a fully connected layer to the hidden state (s_j), and denote the result (r_j) as the target vector of the jth token
430
Compute the mean of all tokens' target vectors as the predicted predicate representation
435

FIG. 4

500

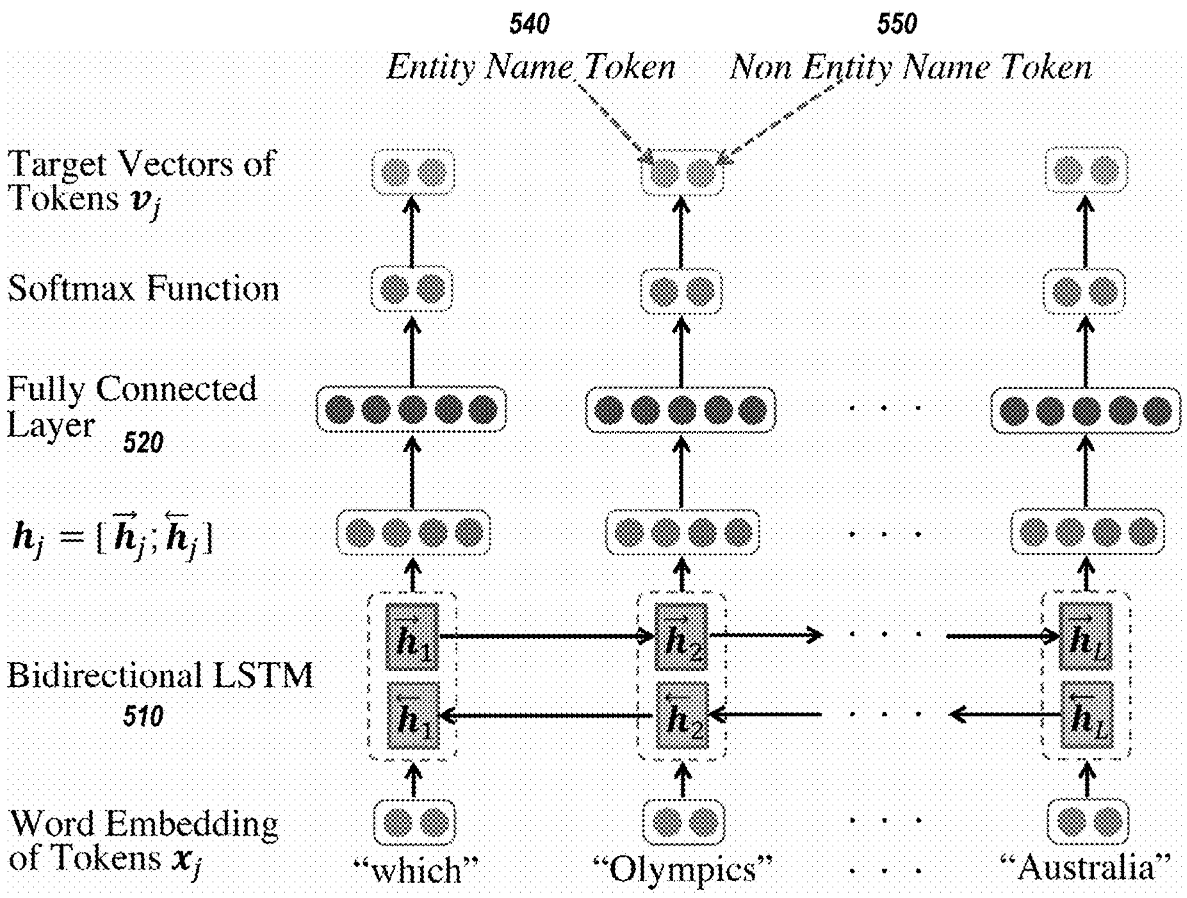

540
Entity Name Token
550
Non Entity Name Token
Target Vectors of Tokens v_j
Softmax Function
Fully Connected Layer 520
$h_j = [\overrightarrow{h}_j; \overleftarrow{h}_j]$
Bidirectional LSTM 510
$\overrightarrow{h}_1$
$\overrightarrow{h}_2$
$\overrightarrow{h}_L$
$\overleftarrow{h}_1$
$\overleftarrow{h}_2$
$\overleftarrow{h}_L$
Word Embedding of Tokens x_j
"which"
"Olympics"
"Australia"

FIG. 5

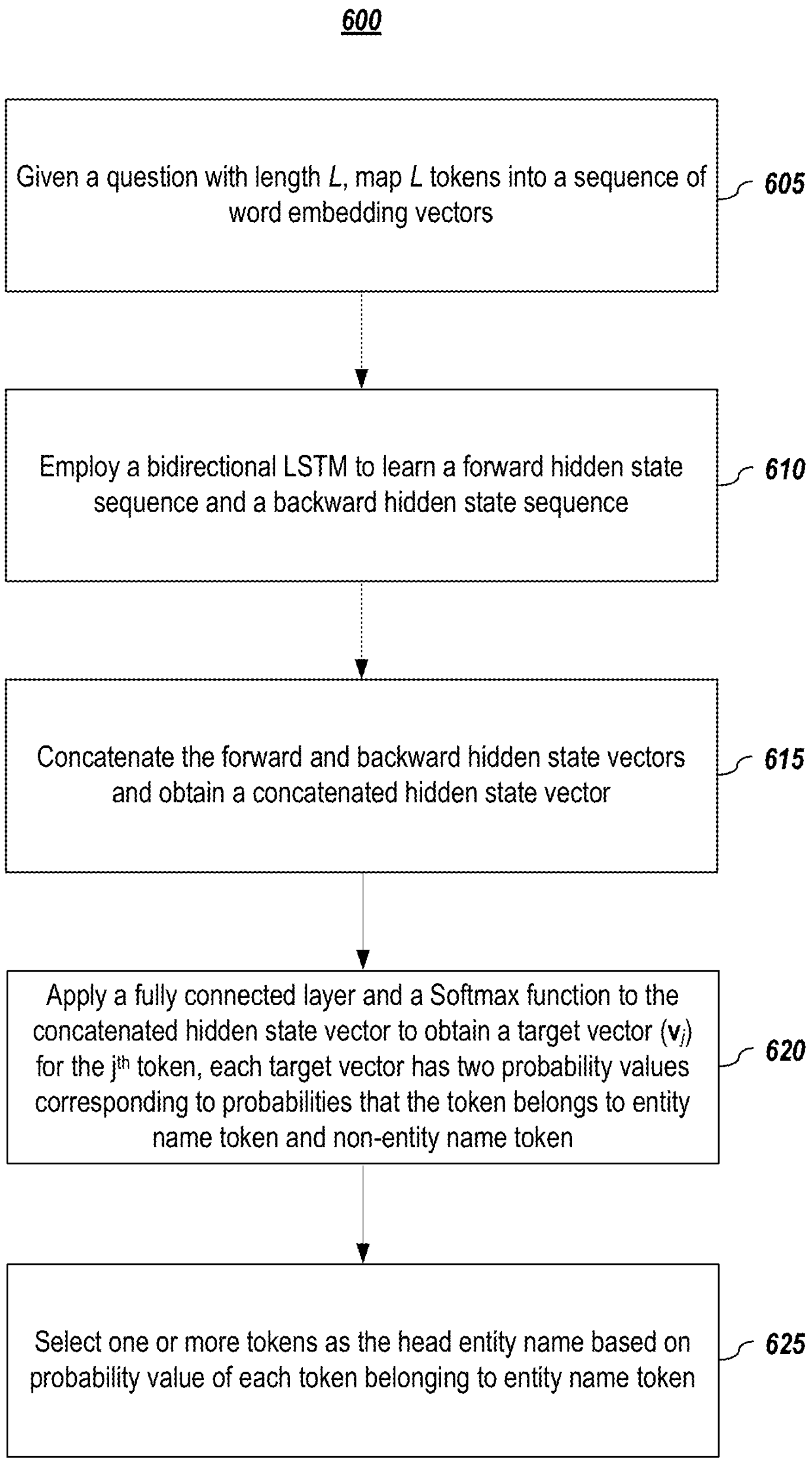

600
Given a question with length L, map L tokens into a sequence of word embedding vectors
605
Employ a bidirectional LSTM to learn a forward hidden state sequence and a backward hidden state sequence
610
Concatenate the forward and backward hidden state vectors and obtain a concatenated hidden state vector
615
Apply a fully connected layer and a Softmax function to the concatenated hidden state vector to obtain a target vector ($\mathbf{v}_j$) for the j^{th} token, each target vector has two probability values corresponding to probabilities that the token belongs to entity name token and non-entity name token
620
Select one or more tokens as the head entity name based on probability value of each token belonging to entity name token
625

FIG. 6

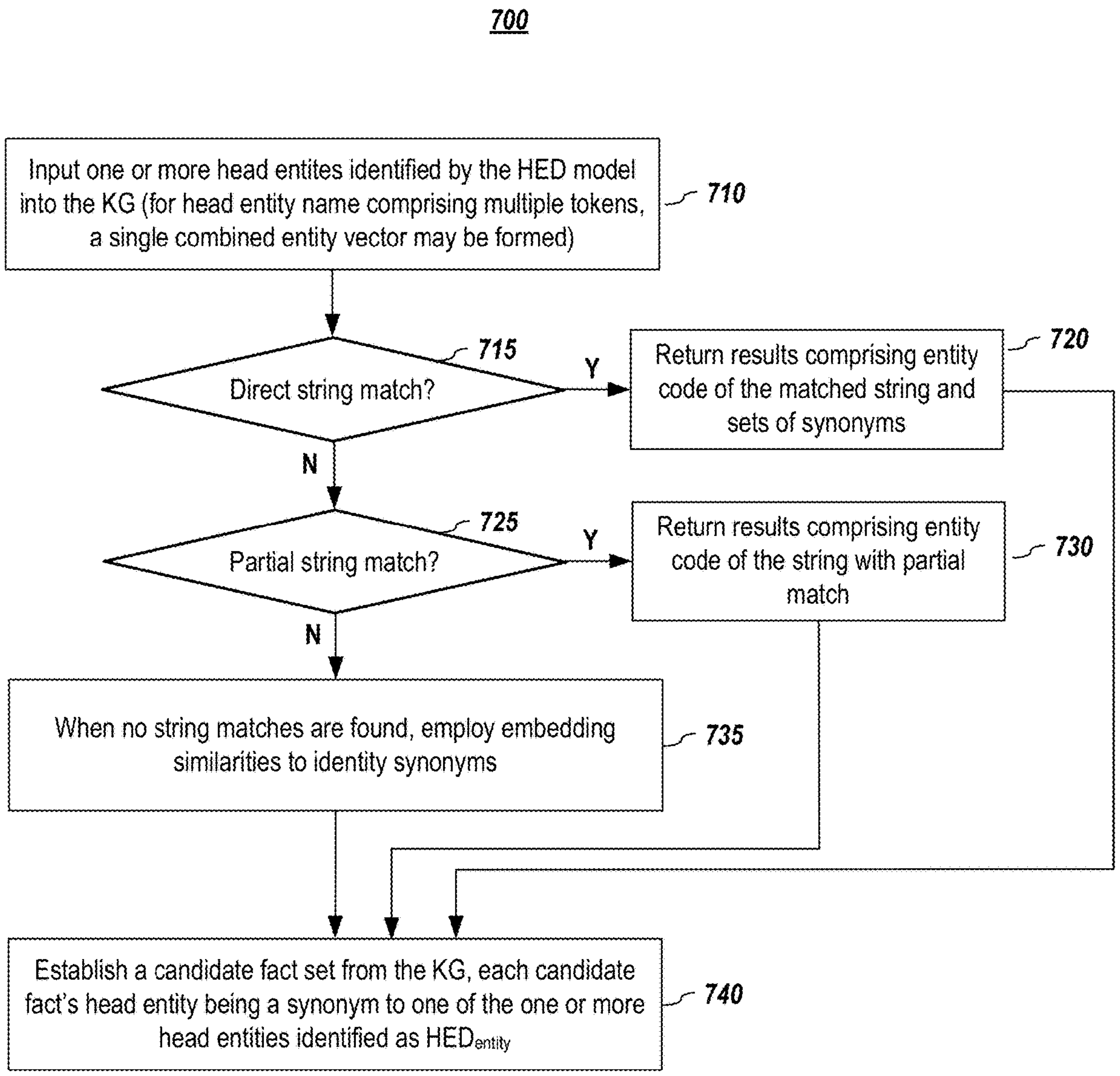

700
Input one or more head entites identified by the HED model into the KG (for head entity name comprising multiple tokens, a single combined entity vector may be formed)
710
Direct string match?
715
Y
N
Return results comprising entity code of the matched string and sets of synonyms
720
Partial string match?
725
Y
N
Return results comprising entity code of the string with partial match
730
When no string matches are found, employ embedding similarities to identity synonyms
735
Establish a candidate fact set from the KG, each candidate fact's head entity being a synonym to one of the one or more head entities identified as HEDentity
740

FIG. 7

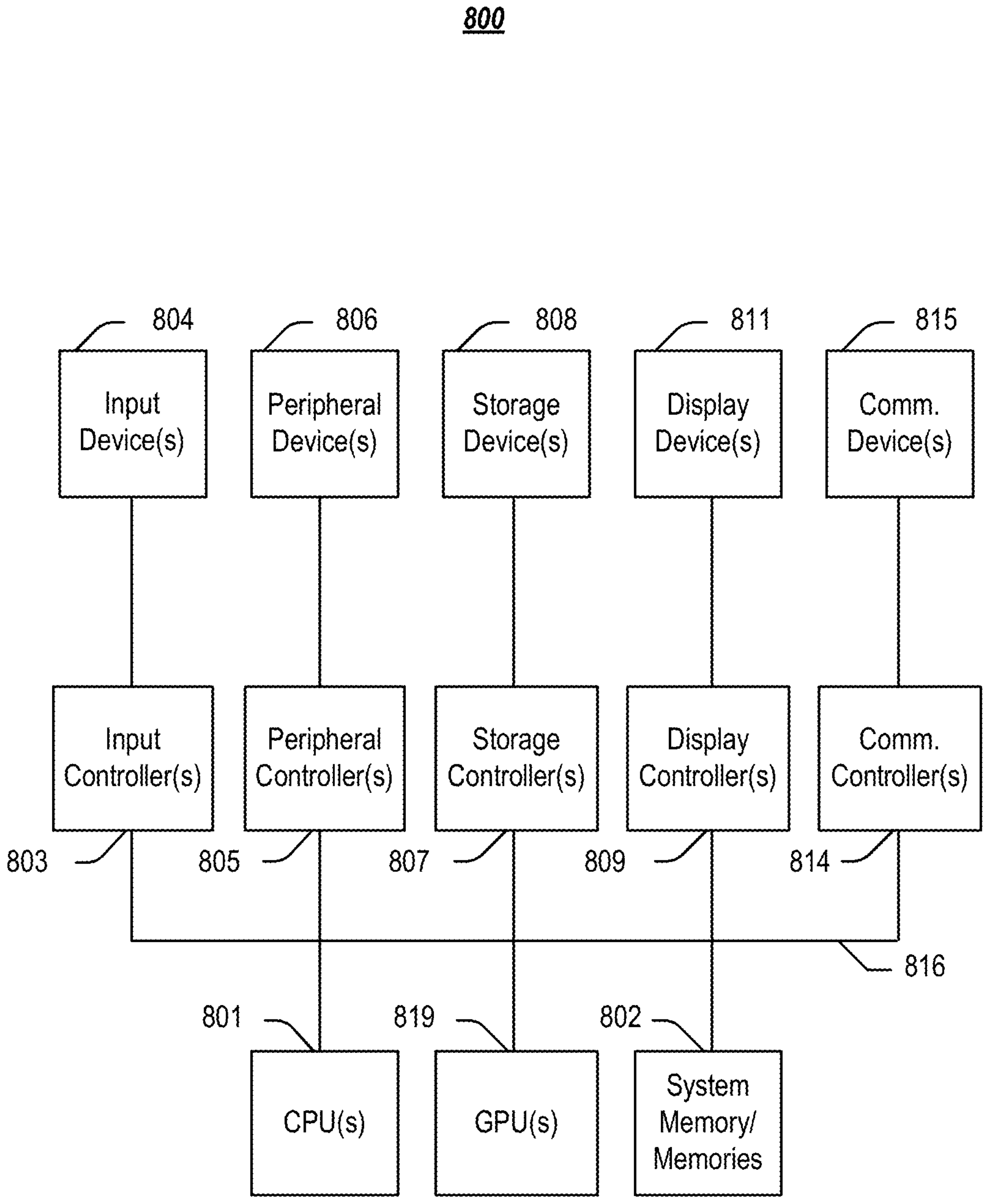

800
804
806
808
811
815
Input Device(s)
Peripheral Device(s)
Storage Device(s)
Display Device(s)
Comm. Device(s)
Input Controller(s)
Peripheral Controller(s)
Storage Controller(s)
Display Controller(s)
Comm. Controller(s)
803
805
807
809
814
816
801
819
802
CPU(s)
GPU(s)
System Memory/ Memories

FIG. 8

KNOWLEDGE-GRAPH-EMBEDDING-BASED QUESTION ANSWERING

BACKGROUND

A. Technical Field

[0001] The present disclosure relates generally to systems and methods for question answering. More particularly, the present disclosure relates to systems and methods for question answering over knowledge graph.

B. Background

[0002] Question answering over knowledge graph (QA-KG) aims to use facts in a knowledge graph (KG) to answer natural language questions. It helps end users more efficiently and more easily access the substantial and valuable knowledge in the KG, without knowing its data structures. QA-KG is a nontrivial problem since capturing the semantic meaning of natural language is difficult for a machine. Many knowledge graph embedding methods have been proposed. One key idea is to represent each predicate/entity as a low-dimensional vector, such that the relation information in the KG could be preserved. However, this remains a challenging task since a predicate could be expressed in different ways in natural language questions. Furthermore, the ambiguity of entity names and partial names makes the number of possible answers large.

[0003] Accordingly, what is needed are systems and methods that can be used to make question answering over knowledge graph more effective and more robust.

BRIEF DESCRIPTION OF THE DRAWINGS

[0004] References will be made to embodiments of the invention, examples of which may be illustrated in the accompanying figures. These figures are intended to be illustrative, not limiting. Although the invention is generally described in the context of these embodiments, it should be understood that it is not intended to limit the scope of the invention to these particular embodiments. Items in the figures are not to scale.

[0005] FIG. (“FIG.”) **1** graphically depicts a knowledge embedding based question answering (KEQA) framework, according to embodiments of the present disclosure.

[0006] FIG. **2** depicts a method for question answering with a KEQA framework, according to embodiments of the present disclosure.

[0007] FIG. **3** graphically depicts architecture of predicate and head entity learning models, according to embodiments of the present disclosure.

[0008] FIG. **4** depicts a method for predicting predicate of an input question using a predicate and head entity learning model, according to embodiments of the present disclosure.

[0009] FIG. **5** depicts a structure of a Head Entity Detection (HED) model, according to embodiments of the present disclosure.

[0010] FIG. **6** depicts a method for identifying one or more head entities of an input question using a HED model, according to embodiments of the present disclosure.

[0011] FIG. **7** depicts a method for searching head entity synonyms in a KG using head entity names identified by a HED model, according to embodiments of the present disclosure.

[0012] FIG. **8** depicts a simplified block diagram of a computing device/information handling system, in accordance with embodiments of the present document.

DETAILED DESCRIPTION OF EMBODIMENTS

[0013] In the following description, for purposes of explanation, specific details are set forth in order to provide an understanding of the present disclosure. It will be apparent, however, to one skilled in the art that embodiments may be practiced without these details. Furthermore, one skilled in the art will recognize that embodiments of the present disclosure, described below, may be implemented in a variety of ways, such as a process, an apparatus, a system, a device, or a method on a tangible computer-readable medium.

[0014] Components, or modules, shown in diagrams are illustrative of exemplary embodiments of the invention and are meant to avoid obscuring the present disclosure. It shall also be understood that throughout this discussion that components may be described as separate functional units, which may comprise sub-units, but those skilled in the art will recognize that various components, or portions thereof, may be divided into separate components or may be integrated together, including integrated within a single system or component. It should be noted that functions or operations discussed herein may be implemented as components. Components may be implemented in software, hardware, or a combination thereof.

[0015] Furthermore, connections between components or systems within the figures are not intended to be limited to direct connections. Rather, data between these components may be modified, re-formatted, or otherwise changed by intermediary components. Also, additional or fewer connections may be used. It shall also be noted that the terms “coupled,” “connected,” or “communicatively coupled” shall be understood to include direct connections, indirect connections through one or more intermediary devices, and wireless connections.

[0016] Reference in the specification to “one embodiment,” “preferred embodiment,” “an embodiment,” or “embodiments” means that a particular feature, structure, characteristic, or function described in connection with the embodiment is included in at least one embodiment of the invention and may be in more than one embodiment. Also, the appearances of the above-noted phrases in various places in the specification are not necessarily all referring to the same embodiment or embodiments.

[0017] The use of certain terms in various places in the specification is for illustration and should not be construed as limiting. A service, function, or resource is not limited to a single service, function, or resource; usage of these terms may refer to a grouping of related services, functions, or resources, which may be distributed or aggregated. An image may be a still image or from a video.

[0018] The terms “include,” “including,” “comprise,” and “comprising” shall be understood to be open terms and any lists the follow are examples and not meant to be limited to the listed items. Any headings used herein are for organizational purposes only and shall not be used to limit the scope of the description or the claims. Each reference mentioned in this patent document is incorporated by reference herein in its entirety.

[0019] Furthermore, one skilled in the art shall recognize that: (1) certain steps may optionally be performed; (2) steps may not be limited to the specific order set forth herein; (3) certain steps may be performed in different orders; and (4) certain steps may be done concurrently.

A. Introduction

[0020] With the rise of large-scale knowledge graphs such as Wikidata, Freebase, Dbpedia, and YAGO, question answering (QA) over knowledge graph has become a crucial topic and attracts massive attention. A knowledge graph (KG) typically is a directed graph with real-world entities as nodes and their relations as edges. In this graph, each directed edge, along with its head entity and tail entity, constitute a triple, i.e., (head entity, predicate, tail entity), which is also named as a fact. Real-world knowledge graphs may contain millions or billions of facts. Their large volume and complex data structures make it difficult for regular users to access the substantial and valuable knowledge in them. To bridge the gap, Question Answering over Knowledge Graph (QA-KG) is proposed. It targets trying to automatically translate the end users' natural language questions into structured queries such as SPARQL, and returning entities and/or predicates in the KG as answers. For example, given the question "Which Olympics was in Australia?", QA-KG aims to identify its corresponding two facts, i.e., (Australia, olympics_participated_in, 1952/2004 Summer Olympics).

[0021] Question answering over knowledge graph provides a way for artificial intelligence systems to incorporate knowledge graphs as a key ingredient to answer human questions, with applications ranging from search engine design to conversational agent building. However, the QA-KG problem is far from solved since it involves multiple challenging subproblems such as semantic analysis and entity linking.

[0022] The effectiveness of knowledge graph embedding in different real-world applications motivates exploring its potential usage in solving the QA-KG problem in this patent document. Knowledge graph embedding targets learning a low-dimensional vector representation for each predicate/entity in a KG, such that the original relations are well preserved in the vectors. These learned vector representations may be employed to complete a variety of downstream applications efficiently. Examples include KG completion, recommender systems, and relation extraction. In this patent document, embodiments of the knowledge graph embedding are presented to perform QA-KG. The KG embedding representations may advance the QA-KG in several ways. They not only are within a low-dimensional space, but also could promote the downstream applications to take the entire KG into consideration, because even a single predicate/entity representation is a result of interactions with the whole KG. In addition, similar predicates/entities tend to have similar vectors. This property may be used to help the downstream algorithms handle predicates or entities that are not in the training data.

[0023] However, it remains a nontrivial task to conduct QA-KG based on the knowledge graph embedding. There are three major challenges. First, a predicate often has various expressions in natural language questions. These expressions could be quite different from the predicate names. For instance, the predicate person.nationality can be expressed as "what is . . . 's nationality", "which country is . . . from", "where is . . . from", etc. Second, even assuming that the entity names could be accurately identified, the ambiguity of entity names and partial names would still make it difficult to find the correct entity, since the number of candidates is often large. As the size of KG keeps increasing, many entities would share the same names. Also, end users could use partial names in their utterances. For example, in the question "How old is Obama?", only part of the entity name Barack Obama is indicated. Third, the domains of end users' questions are often unbounded, and any KG is far from complete. New questions might involve predicates that are different from the ones in the training. This makes demands on the robustness of the QA-KG algorithms.

[0024] To bridge the gap, this patent document discloses how to take advantage of the knowledge graph embedding to perform question answering. In the present disclosure, a focus is on the most common type of questions in QA-KG, i.e., simple questions. A simple question is a natural language question that only involves a single head entity and a single predicate. Through analyzing the problem, three research questions are answered: (i) How to apply the predicate embedding representations to bridge the gap between the natural language expressions and the KG's predicates?; (ii) How to leverage the entity embedding representations to tackle the ambiguity challenge?; and (iii) How to take advantage of the global relations preserved in the KG embedding representations to advance the QA-KG framework? Following these questions, the present document discloses embodiments of a framework named Knowledge Embedding based Question Answering (KEQA). In summary, some key contributions of the present document are as follows:

[0025] Formally define the knowledge graph embedding based question answering problem.

[0026] Disclosure of embodiments of an effective framework KEQA that answer a natural language question by jointly recovering its head entity, predicate, and tail entity representations in the knowledge graph embedding spaces.

[0027] Design a joint distance metric that takes the structures and relations preserved in the knowledge graph embedding representations into consideration.

[0028] Empirically demonstrate the effectiveness and robustness of KEQA embodiments on a large benchmark, i.e., SimpleQuestions.

B. Some Related Work

[0029] Some related works in various aspects are summarized in this Section.

[0030] Embedding-based question answering over KG attracts lots of attention recently. It is related to, but different from, the presented KG embedding based question answering problem. The former relies on low-dimensional representations that are learned during the training of the QA-KG methods. The latter performs KG embedding to learn the low-dimensional representations first, and then conducts the QA-KG task. Yih et al. (Semantic Parsing via Staged Query Graph Generation: Question Answering with Knowledge Base. In ACL-IJCNLP) and Bao et al. (Constraint-Based Question Answering with Knowledge Graph. In COLING. 2503-2514) reformulated the question answering problem as the generation of particular subgraphs. A series of work proposed to project questions and candidate answers (or entire facts) into a unified low-dimensional space based on the training questions, and measure their matching scores by the similarities between their low-dimensional representations. Some achieved this projection by learning low-dimensional representations for all words, predicates, and entities, based on the training questions and paraphrases of questions. Some achieved this projection by using the logical properties of questions and potential facts, such as semantic embedding and entity types. Several deep learning based models achieved this projection by feeding words in questions into convolutional neural networks, LSTM networks, or gated recurrent units neural networks. Das et al. (Question Answering on Knowledge Bases and Text using Universal Schema and Memory Networks. In ACL, 2017) achieved this projection by using matrix factorization to incorporate the corpus into the KG, and LSTM to embed a question. Most of these models rely on the margin-based ranking objective functions to learn the model weights. Several works explored leveraging the character-level neural networks to advance the performance. Most recently, Mohammed et al. (Strong Baselines for Simple Question Answering over Knowledge Graphs with and without Neural Networks, NAACL-HLT. 291-296) and Ture et al. (No Need to Pay Attention: Simple Recurrent Neural Networks Work, EMNLP. 2866-2872) considered each predicate as a label category, and performed predicate linking via deep classification models.

[0031] Knowledge graph embedding targets at representing the high-dimensional KG as latent predicate and entity representations P and E. Bordes et al. (Learning Structured Embeddings of Knowledge Bases. 2011 AAAI) achieved this goal by constructing two transform matrices M_{head} and M_{tail} for each type of predicate ℓ, and minimizing the distance between projections $M_{head}e_h$ and $M_{tail}e_t$ for all facts (h,ℓ,t) with ℓ as predicate. Bordes et al. (Translating Embeddings for Modeling Multi-relational Data. 2013 NIPS. 2787-2795) designed a translation-based model TransE. It trains two matrices P and E, aiming to minimize the overall distance $\Sigma\|e_h+\mathbf{p}_\ell-e_t\|_2^2$ for all facts (h,ℓ,t). Motivated by TransE, a series of translation-based models have been explored. Wang et al. (Knowledge Graph Embedding by Translating on Hyperplanes. 2014 AAAI) proposed TransH to handle one-to-many or many-to-one relations. Instead of measuring the distance between e_h and e_t directly, TransH projects them into a predicate-specific hyperplane. Lin et al. (Learning Entity and Relation Embeddings for Knowledge Graph Completion. 2015 AAAI 2181-2187) proposed TransR, which defines a transform matrix $\mathbf{M}_\ell$ for each predicate ℓ and targets at minimizing $\Sigma\|e_h\mathbf{M}_\ell+\mathbf{p}_\ell-e_t\mathbf{M}_\ell\|_2^2$. Lin et al. (Modeling Relation Paths for Representation Learning of Knowledge Bases, 2015 EMNLP. 705-814) proposed PTransE, which advances TransE via taking multi-hop relations into consideration.

[0032] Efforts have also been devoted to incorporating the semantic information in a corpus into KG embedding. Some demonstrated that using pre-trained word embedding to initialize KG embedding methods would enhance the performance. Several work explored trying to advance TransE, either via taking relation mentions in corpus into consideration, or via projecting predicate/entity representations into a semantic hyperplane learned from the topic model. Attempts have also been made to apply TransE and word2vec to model a KG and a corpus respectively, and then fuse them based on anchors in Wikipedia, entity descriptions, or contextual words of predicates/entities learned from the corpus. Zhang et al. (Joint Semantic Relevance Learning with Text Data and Graph Knowledge. In Workshop on Continuous Vector Space Models and their Compositionality. 32-40) jointly embedded the KG and corpus via negative sampling (Distributed Representations of Words and Phrases and Their Compositionality, 2013 NIPS. 3111-3119). Xie et al. (Representation Learning of Knowledge Graphs with Entity Descriptions. 2016 AAAI 2659-2665) and Fan et al. (Distributed Representation Learning for Knowledge Graphs with Entity Descriptions, Pattern Recognition Letters 93 (2017), 31-37) explored the semantic information in entity descriptions to advance KG embedding.

C. Problem Statement

[0033] Notations:

[0034] In this patent document, an uppercase bold letter is used to denote a matrix (e.g., W) and a lower case bold letter to represent a vector (e.g., p). The i^{th} row of a matrix P is denoted as p_i. The transpose of a vector is denoted as p^T. The ℓ^2 norm of a vector is denoted as $\|p\|_2$. $\{p_i\}$ is used to represent a sequence of vectors p_i. The operation $s=[x; h]$ denotes concatenating column vectors x and h into a new vector s.

[0035] Definition 1 (Simple Question) If a natural language question only involves a single head entity and a single predicate in the knowledge graph, and takes their tail entity/entities as the answer, then this question is referred as a simple question.

[0036] Some symbols in this patent document are summarize in Table 1. (h,ℓ,t) is used to represent a fact, which means that there exists a relation ℓ from a head entity h to a tail entity t. Let $\mathcal{G}$ be a knowledge graph that consists of a large number of facts. The total numbers of predicates and entities are represented as M and N. The names of these predicates and entities are given. In one or more embodiments, a scalable KG embedding algorithm, such as TransE and TransR, is applied to $\mathcal{G}$, and the embedding representations of its predicates and entities denoted as P and E, respectively, are obtained. Thus, the vector representations of the i^{th} predicate and j^{th} entity are denoted as p_i and e_j respectively. The relation function defined by the KG embedding algorithm is $f(\cdot)$, i.e., given a fact (h,ℓ,t), one may have $e_t \approx f(e_h, \mathbf{p}_\ell)$. Letting Q be a set of simple questions. For each question in Q, the corresponding head entity and predicate are given.

TABLE 1

Some symbols and their definitions

Notations	Definitions
$\mathcal{G}$	a knowledge graph
(h, ℓ, t)	a fact, i.e., (head entity, predicate, tail entity)
Q	a set of simple questions with ground truth facts
M	total number of predicates in $\mathcal{G}$
N	total number of entities in $\mathcal{G}$
d	dimension of the embedding representations
$P \in \ell^{M \times d}$	embedding representations of all predicates in $\mathcal{G}$
$E \in \ell^{M \times d}$	embedding representations of all entities in $\mathcal{G}$
$f(\bullet)$	relation function, given (h, ℓ, t), $\Rightarrow e_t \approx f(e_h, \mathbf{p}_\ell)$
$\hat{\mathbf{p}}_\ell \in \ell^{1 \times d}$	predicted predicate representation
$\hat{e}_h \in \ell^{1 \times d}$	predicted head entity representation
HED	Head Entity Detection model
HED_{entity}	head entity name tokens returned by the HED
HED_{non}	non entity name tokens returned by the HED

[0037] The terminology simple question is defined in Definition 1. A simple question may be answered by the machine straightforwardly if its single head entity and single predicate are identified. Given the conditions described above, the knowledge graph embedding based question answering problem is now formally defined as follows:

[0038] Given a knowledge graph $\mathcal{G}$ associated with all its predicates' and entities' names and embedding representations P & E, the relation function $f(\cdot)$, as well as a set of simple questions Q associated with corresponding head entities and predicates, embodiments of an end-to-end framework are disclosed to take a new simple question as input and automatically return the corresponding head entity and predicate. Performance of the framework is evaluated by the accuracy of predicting both head entity and predicate correctly.

D. Embodiments of Knowledge Embedding Based OA-KG

[0039] Simple questions constitute the majority of questions in the QA-KG problem. Each of them may be answered by the tail entity/entities if the correct head entity and predicate are identified. To accurately predict the head entity and predicate, this patent document discloses embodiments of a Knowledge Embedding based Question Answering (KEQA) framework, which is illustrated in FIG. **1**. The KG $\mathcal{G}$ **160** is already embedded into two low-dimensional spaces (Predicate Embedding Space **140** and Entity Embedding Space **150**), and each fact (h,ℓ,t) may be represented as three latent vectors, i.e., (e_h, $\mathbf{p}_\ell$, e_t). Thus, given a question **110**, as long as its corresponding fact's e_h and $\mathbf{p}_\ell$ may be predicted, this question may be answered **170** correctly. Instead of inferring the head entity and predicate directly, KEQA embodiments target jointly recovering the question's head entity, predicate, and tail entity representations ($\hat{e}_h$, $\mathbf{p}_\ell$, $\hat{e}_t$) in the knowledge graph embedding spaces.

[0040] FIG. **2** depicts a method for question answering with a KEQA framework, according to embodiments of the present disclosure. In one or more embodiments, KEQA achieves an answer via the following steps. (i) Based on the questions in Q and their predicates' embedding representations, KEQA trains (**205**) a predicate learning model **120** that takes a question **110** as the input and returns a vector $\hat{\mathbf{p}}_\ell$ that lies in the KG predicate embedding space **140** as the predicted predicate representation. Similarly, a head entity learning model **130** is constructed to predict (**210**) the question's head entity representation $\hat{e}_h$ in the KG entity embedding space **150**. (ii) Since the number of entities in a KG is often large, KEQA employs a Head Entity Detection model to reduce (**215**) the candidate head entities. A main goal is to identify one or more tokens in the question as the predicted head entity name, then the search space in $\mathcal{G}$ is reduced from the entire entities to a number of entities with the same or similar names. Then, $\hat{e}_h$ is mainly used to tackle the ambiguity challenge. (iii) Given the relation function $f(\cdot)$ defined by the KG embedding algorithm, the KEQA embodiment computes (**220**) the predicted tail entity representation $\hat{e}_t = f(\hat{e}_h, \hat{\mathbf{p}}_\ell)$. The predicted predicate representation $\hat{\mathbf{p}}_\ell$, the predicted head entity representation $\hat{e}_h$, and the predicted tail entity representation $\hat{e}_t$ form predicted fact ($\hat{e}_h$, $\hat{\mathbf{p}}_\ell$, $\hat{e}_t$). Based on a carefully-designed joint distance metric, the predicted fact ($\hat{e}_h$, $\hat{\mathbf{p}}_\ell$, $\hat{e}_t$)'s closest fact in $\mathcal{G}$ is selected (**225**) and returned as the question's answer **170**.

1. Embodiments of Knowledge Graph Embedding

[0041] In one or more embodiments, the disclosed framework KEQA employs the embedding representations of all predicates P and entities E as the infrastructure. In one or more embodiments, an existing KG embedding algorithm may be utilized to learn P and E. Examples of existing KG embedding methods that may be used include, but are not limited to, TransE, TransR, TransH, etc.

[0042] Knowledge graph embedding aims to represent each predicate/entity in a KG as a low-dimensional vector, such that the original structures and relations in the KG are preserved in these learned vectors. A core idea of most of the existing KG embedding methods could be summarized as follows. For each fact (h, ℓ, t) in $\mathcal{G}$, its embedding representations is denoted as (e_h, $\mathbf{p}_\ell$, e_t). The embedding algorithm initializes the values of e_h, $\mathbf{p}_\ell$, and e_t randomly or based on the trained word embedding models. Then, a function $f(\cdot)$ that measures the relation of a fact (h, ℓ, t) in the embedding spaces is defined, i.e., $e_t \approx f(e_h, \mathbf{p}_\ell)$. For example, TransE defines the relation as $e_t \approx e_h + \mathbf{p}_\ell$ and TransR defines it as $e_t \mathbf{M}_\ell \approx e_h \mathbf{M}_\ell + \mathbf{p}_\ell$, where $\mathbf{M}_\ell$ is a transform matrix of predicate ℓ. Finally, the embedding algorithm minimizes the overall distance between e_t and $f(e_h, \mathbf{p}_\ell)$, for all the facts in $\mathcal{G}$. A typical way is to define a margin-based ranking criterion and train on both positive and negative samples, i.e., facts and synthetic facts that do not exist in $\mathcal{G}$.

[0043] As shown in FIG. **1**, the surface is defined where the learned predicate representations $\{p_i\}$ for i=1, . . . , M lie in, as the predicate embedding space. The surface where $\{e_1\}$ for i=1, . . . , N lie in is denoted as the entity embedding space.

2. Embodiments of Predicate and Head Entity Learning Models

[0044] Given a simple question, the objective is to find a point in the predicate embedding space as its predicate representation $\hat{\mathbf{p}}_\ell$, and a point in the entity embedding space as its head entity representations $\hat{e}_h$.

[0045] In one or more embodiments, for all the questions that can be answered by $\mathcal{G}$, their predicates' vector representations should lie in the predicate embedding space. Thus, an aim is to design a model that takes a question as the input and returns a vector $\hat{\mathbf{p}}_\ell$ that is as close as possible to this question's predicate embedding representation $\mathbf{p}_\ell$. To achieve this goal, a neural network architecture embodiment, as shown in FIG. **3**, is employed. In one or more embodiments, the architecture mainly comprises a bidirectional recurrent neural network layer **310** and an attention layer **325**. In one or more embodiments, the bidirectional recurrent neural network layer **310** is a bidirectional long short-term memory (LSTM). A core idea is to take the order and the importance of words into consideration. Words with different orders could have different meanings, and the importance of words could be different. For example, the entity name related words in a question often have less contribution to the predicate learning model.

[0046] Neural Network Based Predicate Representation Learning.

[0047] To Predict the Predicate of a question, a traditional solution is to learn the mapping based on the semantic parsing and manually-created lexicons, or simply consider each type of predicate as a label category to transform it into a classification problem. However, since the domains of end users' questions are often unbounded, a new question's predicate might be different from all the ones in the training data. The traditional solutions could not handle this scenario. In addition, it is observed that the global relation information preserved in P and E is available and could be potentially used to improve the overall question answering accuracy. To bridge the gap, embodiments of a predicate learning model based on neural networks are set forth herein.

[0048] With the long short-term memory (LSTM) as a typical example of the recurrent neural network, FIG. **3** illustrates the architecture of predicate and head entity learning models, according to one or more embodiments of the present disclosure. FIG. **4** depicts a method for predicting a predicate of an input question using a predicate and head entity learning model, according to embodiments of the present disclosure. Given a question with length L, its L tokens are first mapped (**405**) into a sequence of word embedding vectors $\{x_1\}$ **305**, for j=1, . . . , L, based on a pre-trained model, such as GloVe (Pennington, et al., GloVe: Global Vectors for Word Representation, In EMNLP. 1532-1543), although other embedding techniques may be employed. Then, a bidirectional LSTM **310** is employed (**410**) to learn a forward hidden state sequence $(\vec{h}_1, \vec{h}_2, \ldots, \vec{h}_L)$ and a backward hidden state sequence $(\overleftarrow{\mathbf{h}}_1, \overleftarrow{\mathbf{h}}_2, \ldots, \overleftarrow{\mathbf{h}}_L)$. Taking the backward one as an example, $\{\overleftarrow{\mathbf{h}}_j\}$ are computed via the following equations.

$$f_j=\sigma(W_{xf}x_j+W_{hf}\overleftarrow{\mathbf{h}}_{j+1}+b_f) \quad (1)$$

$$i_j=\sigma(W_{xi}x_j+W_{hi}\overleftarrow{\mathbf{h}}_{j+1}+b_i) \quad (2)$$

$$o_j=\sigma(W_{xo}x_j+W_{ho}\overleftarrow{\mathbf{h}}_{j+1}+b_o) \quad (3)$$

$$c_j=f_j\circ c_{j+1}+i_j \tanh(W_{xc}x_j+W_{hc}\overleftarrow{\mathbf{h}}_{j+1}+b_c) \quad (4)$$

$$\overleftarrow{\mathbf{h}}_j=o_j\circ \tanh(c_j) \quad (5)$$

[0049] where f_j, i_j, and o_j are the forget, input, and output gates' activation vectors respectively. c_j is the cell state vector. σ and tanh are the sigmoid and Hyperbolic tangent functions. ∘ denotes the Hadamard product. Concatenating (**415**) the forward and backward hidden state vectors, one may obtain concatenated hidden state vector $h_j=[\vec{h}_j; \overleftarrow{\mathbf{h}}_j]$ 315.

[0050] In one or more embodiments, the attention weight **320** of the j^{th} token, i.e., α_j, is calculated based on the following formulas:

$$\alpha_j = \frac{\exp(q_j)}{\sum_{i=1}^{L}\exp(q_j)} \quad (6)$$

$$q_j = \tanh(w^{\top}[x_j; h_j] + b_q) \quad (7)$$

[0051] where b_q is a bias term. The attention weight α_j may be applied (**420**) to h_j to obtain a weighted hidden state vector, which is then concatenated (**425**) with the word embedding x_j, resulting a hidden state $s_j=[x_j; \alpha_j h_j]$ **325**. A fully connected layer is then applied (**430**) to s_j, and its result, $r_j\in\mathbb{R}^{d\times 1}$, is denoted as the target vector **330** of the j^{th} token. The predicted predicate representation $\hat{\mathbf{p}}_\ell$ **335** may be computed (**435**) as the mean of all tokens' target vectors, that is:

$$\hat{p}_\ell = \frac{1}{L}\sum_{j=1}^{L} r_j^{\top} \quad (8)$$

[0052] In one or more embodiments, all the weight matrices, weight vector w, and bias terms are calculated based on the training data, i.e., questions Q in and their predicates' embedding representations.

[0053] Neural Network based Head Entity Learning Model.

[0054] In one or more embodiments, given a question, instead of inferring the head entity directly, a target is recovering its representation in the KG embedding space. Thus, a goal of the head entity learning model is to compute a vector $\hat{e}_h$ that is as close as possible to this question's head entity embedding representation. Similar to the computation of $\hat{\mathbf{p}}_\ell$, the same neural network architecture in FIG. **3** may be used to obtain the predicted head entity representation $\hat{e}_h$.

[0055] However, the number of entities in a KG is often large, and it could be expensive and noisy when comparing $\hat{e}_h$ with all entity embedding representations in E. To make the learning more efficient and effective, KEQA embodiments may employ a head entity detection model to reduce the number of candidate head entities.

3. Embodiments of Head Entity Detection Model

[0056] In this step, the goal is to select one or several successive tokens in a question, as the name of the head entity, such that the search space could be reduced from the entire entities to a number of entities with the same or similar names. Then the main role of $\hat{e}_h$ would become handling the ambiguity challenge.

[0057] In one or more embodiments, to make the framework simple, a bidirectional recurrent neural network (e.g., LSTM) based model is employed to perform the head entity token detection task. FIG. **5** shows an architecture of a Head Entity Detection (HED) model, according to one or more embodiments of the present disclosure. As shown in FIG. **5**, the HED model comprise a bidirectional LSTM **510** and a fully connected layer **520**. The HED model has a similar structure to the one in predicate/head entity learning models, but without the attention layer.

[0058] FIG. **6** depicts a method for identifying one or more head entities of an input question using a HED model, according to one or more embodiments of the present disclosure. In one or more embodiments, the question is first mapped (**605**) into a sequence of word embedding vectors $\{x_j\}$, for j=1, . . . , L, and then a bidirectional recurrent neural network is applied (**610**) to x_j to learn a forward hidden state sequence $\vec{h}_1$ and a backward hidden state sequence $\overleftarrow{\mathbf{h}}_j$. The forward and backward hidden states are concatenated (**615**) into a concatenated hidden state $h_j=[\vec{h}_j;]$. A fully connected layer and a softmax function are then applied (**620**) to h_j, resulting the target vector $v_j\in\mathbb{R}^{2\times 1}$. The two values in v_j are corresponding to the probabilities that the j^{th} token belongs to the two label categories, i.e., entity name token and non-entity name token. In such a way, each token is classified and one or several tokens are recognized as the head entity name. These tokens are denoted as HED_{entity}, and the remaining tokens in the question are denoted as HED_{non}. One or more tokens are selected (**625**) as the head entity name based on probability value of each token belonging to entity name token.

[0059] In one or more embodiments, the questions in Q and their head entity names are used as the training data to train the HED model. Since entity name tokens in these questions are successive, the trained model would also return successive tokens as HED_{entity} with a high probability. If discrete HED_{entity} is returned, then each successive part would be considered as an independent head entity name. It should be noted that HED_{entity} might be only part of the correct head entity name. Thus, all entities that are the same as or contain HED_{entity} would be included as the candidate head entities, which might still be large since many entities would share the same names in a large KG.

4. Embodiments of Joint Search on Embedding Spaces

[0060] For each new simple question, with its predicate and head entity representations $\hat{\mathbf{p}}_\ell$ and $\hat{e}_h$, as well as its candidate head entities being predicted, the goal is to find a fact in $\mathcal{G}$ that matches these learned representations and candidates the most.

[0061] Joint Distance Metric.

[0062] If a fact's head entity belongs to the candidate head entities, it is named as a candidate fact. Let C be a set that collects all the candidate facts. To measure the distance between a candidate fact (h, ℓ, t) and the predicted representations $(\hat{e}_h, \hat{\mathbf{p}}_\ell)$, an intuitive solution is to represent (h, ℓ, t) as $(e_h, \mathbf{p}_\ell)$ and define the distance metric as the sum of the distance between e_h and $\hat{e}_h$ and distance between $\mathbf{p}_\ell$ and $\hat{\mathbf{p}}_\ell$. This solution, however, does not take the meaningful relation information preserved in the KG embedding representations into consideration.

[0063] In one or more embodiments, a joint distance metric used that takes advantage of the relation information $e_t \approx f(e_h, \mathbf{p}_\ell)$. Mathematically, the proposed joint distance metric may be defined as:

$$\underset{(h,\ell,t)\in C}{\text{minimize}} \|p_\ell - \hat{p}_\ell\|_2 + \beta_1\|e_h - \hat{e}_h\|_2 + \beta_2\|f(e_h, p_\ell) - \hat{e}_t\|_2 - \beta_3 sim[n(h), HED_{entity}] - \beta_4 sim[n(\ell), HED_{non}] \tag{9}$$

[0064] where $\hat{e}_t = f(\hat{e}_h, \hat{\mathbf{p}}_\ell)$. Function $n(\cdot)$ returns the name of the entity or predicate. HED_{entity} and HER_{non} denote the tokens that are classified as entity name and non-entity name by the HED model. Function sim[.,.] measures the similarity of two strings. β_1, β_2, β_3, and β_4 are predefined weights to balance the contribution of each term. In one or more embodiments, ℓ^2 norm is used to measure the distance, and it is straightforward to extend to other vector distance measures.

[0065] The first three terms (which may be referred to as vector distance terms in Eq. (9) measure the distance between a fact (h, ℓ, t) and the prediction in the KG embedding spaces. In one or more embodiments, $f(e_h, \mathbf{p}_\ell)$ is used to represent the tail entity's embedding vector, instead of e_t. In other words, the tail entity embedding vector of the candidate fact used in the joint distance metric is calculated using the defined function $f(\cdot)$ defined by the KG, from a head entity embedding vector and a predicate embedding vector of the candidate fact. This is because in a KG, there might be several facts that have the same head entity and predicate, but different tail entities. Thus, a single tail entity e_t might not be able to answer the question. Meanwhile, $f(e_h, \mathbf{p}_\ell)$ matches the predicted tail entity $\hat{e}_t$ since it is also inferred based on $f(\cdot)$. It is tended to select a fact with head entity name exactly the same as HED_{entity}, and with predicate name mentioned by the question. In one or more embodiments, these two goals are achieved via the fourth and fifth terms (referred as string similarity terms in Eq. (9) respectively. In one or more embodiments, the string similarity terms are incorporated in the joint distance metric to help select a fact with the head entity name exactly the same as HED_{entity}, and with predicate name mentioned by the question. The fact (h^*, ℓ^*, t^*) that minimizes the objective function is returned.

[0066] Knowledge Embedding based Question Answering.

[0067] The entire processes of a KEQA embodiments is summarized in Methodology 1. Given a KG $\mathcal{G}$ and a question set Q with corresponding answers, a predicate learning model, a head entity learning model, and a HED model are trained, as shown from line 1 to line 9. Then, for any new simple question Q, it is input into the trained predicate learning model, head entity learning model, and HED model to learn its predicted predicate representation $\hat{\mathbf{p}}_\ell$, head entity representation $\hat{e}_h$, entity name tokens HED_{entity}, and non-entity name tokens HED_{non}. Based on the learned entity name/names in HED_{entity}, the entire $\mathcal{G}$ is searched to find the candidate fact set C. For all facts in C, their joint distances to the predicted representations $(\hat{e}_h, \hat{\mathbf{p}}_\ell, \hat{e}_t)$ are computed based on the objective function in Eq. (9). The fact (h^*, ℓ^*, t^*) with the minimum distance is selected. Finally, the head entity h* and predicate ℓ^* are returned as the answer of Q.

Methodology 1: A KEQA framework embodiment

```
   Input: $\mathcal{G}$, predicates' and entities' names, P, E, Q, a new simple question Q.
   Output: head entity h* and predicate $\ell^*$.
   /*Training the predicate learning model:                          */
1  for $Q_i$ in Q do
2  |    Take the L tokens of $Q_i$ as the input and its predicate $\ell$ as the label to train, as
   |      shown in Figure 3;
3  |    Update weight matrices {W}, w, {b}, and $b_q$ to minimize the predicate objective
   |
          function $\left\| p_\ell - \frac{1}{L}\sum_{j=1}^{L} r_j^T \right\|_2$
   /*Training the head entity learning model:                        */
4  for $Q_i$ in Q do
```

-continued

Methodology 1: A KEQA framework embodiment

```
5   |  Take the L tokens of $Q_i$ as the input and its head entity h as the label to train, as
    |     shown in Figure 3;
6   |  Update weight matrices and bias terms to minimize the head entity objective
    |     function $\left\| e_h - \frac{1}{L}\sum_{j=1}^{L} r_j^T \right\|_2$
    /*Training the HED model:                                                  */
7   for $Q_i$ in Q do
8   |  Take the L tokens of $Q_i$ as the input and its head entity name positions as the label
    |     to train;
9   |  Update weight matrices and bias as shown in Figure 5
    /*Question answering processes:                                            */
10  Input Q into the predicate learning model to learn $\hat{p}_\ell$;
11  Input Q into the head entity learning model to learn $\hat{e}_h$;
12  Input Q into the HED model to learn $HED_{entity}$ and $HED_{non}$;
13  Find the candidate fact set C from $\mathcal{G}$, based on $HED_{entity}$;
14  For all facts in C, calculate the fact (h*, $\ell$*, t*) that minimizes the objective function in
      Eq. (9).
```

[0068] By way of example related to step 12 (above), from a HED model, the result in FIG. **5** would be that "Australia" would have a high probability of being an entity name token. By way of another example, in one or more embodiments, a phrase that contained "President Abraham Lincoln" would return results with each of the words "Abraham" and "Lincoln" having high probabilities of being combined, at least because the tokens consecutive and/or the tokens are name related, together as one entity.

[0069] FIG. **7** depicts an embodiment implementation of step 13 (above), according to one or more embodiments of the present disclosure. FIG. **7** illustrates an approach for searching head entity synonyms in a KG using head entity names identified by a HED model, according to embodiments of the present disclosure. HED_{entity} may be a single entity, or it may contain several entities. In one or more embodiments, one or more entities identified as head entity by the HED model are input (**710**) into a KG, which comprises entities, predicates, their unique code, and set of synonyms and their embeddings. An entity may comprise one or more tokens, such as "President Abraham Lincoln." Thus, in one or more embodiments, for candidate entity comprising multiple tokens, an entity vector may be formed, such as by a dot product of entity vectors of each token of the entity. In one or more embodiments, the search strategy comprises searching the KG with embedding comparison, string matching, or both, for each identified head entity.

[0070] In one or more embodiments, upon determining (**715**) whether a direct string match exists for each identified head entity, the process either goes to returning (**720**) results, which results may comprise entity code of the matched string and a set or sets of synonyms. In one or more embodiments, if a direct string match is not found, the search may be extended to attempt to identify (**725**) whether one or more partial string matches exists. For example, the two strings "President Abraham Lincoln" and "the President of the United States during the Civil War" are partial matched and also regarded to be the same entity. If one or more partial string matches are identified, the search process returns (**730**) results, which may comprise, for each partial match, its entity code of one or more sets of synonyms. In one or more embodiments, in response to no direct or partial string matches being found, embedding similarities are employed to identity (**735**) head entity synonyms for each identified head entity. All synonyms for the identified head entity via direct string match, partial string match, and embedding similarity are collected together to establish (**740**) a candidate fact set for the one or more identified head entities.

[0071] In one or more embodiments, for each search strategy (string match and embedding comparison), a threshold or thresholds may be used to decide whether enough similarity or matching exists. The threshold in string match may or may not the same as the threshold for embedding comparison.

[0072] By way of further illustration related to steps 13 and 14 (above), once a set of candidate head entities are found (e.g., from a search process such as that shown in FIG. **7**), a candidate fact set C can be constructed based on a set of found head entities, the predicate from the Q found in training, and the tail entity, which is known from the training data. Given the constructed candidate fact set with the known tail entity (or ground truth) from the training data, the candidate fact set may be put into Eq. (9) for joint training of models in the KEQA framework. Once the training is done, the KEQA framework may be used to predict tail entity for new question Q in a testing data.

[0073] It shall be noted that these training embodiments and results are provided by way of illustration and were performed under specific conditions using a specific embodiment or embodiments; accordingly, neither these training embodiments nor their results shall be used to limit the scope of the disclosure of the current patent document.

[0074] By way of general summary, the disclosed framework KEQA embodiments enjoy several nice properties. First, by performing question answering based on the KG embedding, KEQA embodiments are able to handle questions with predicates and entities that are different from all the ones in the training data. Second, by taking advantage of the structure and relation information preserved in the KG embedding representations, KEQA embodiments can perform the head entity, predicate, and tail entity predictions jointly. The three subtasks would mutually complement each other. Third, KEQA framework is generalizable to different KG embedding algorithms. Thus, the performance of a KEQA embodiment may be further improved by more sophisticated KG embedding algorithms.

E. Some Experiments

[0075] It shall be noted that these experiments and results are provided by way of illustration and were performed under specific conditions using a specific embodiment or embodiments; accordingly, neither these experiments nor their results shall be used to limit the scope of the disclosure of the current patent document.

[0076] In this section, the effectiveness and generalizability of tested embodiments of the disclosed framework KEQA on a large QA-KG benchmark are evaluated. In one or more experiments, the following three research questions are studied:

[0077] Q1: How effective is the KEQA embodiment compared with the state-of-the-art QA-KG methods w.r.t. different freebase subsets?

[0078] Q2: How does the performance of the KEQA embodiment vary when different KG embedding algorithms are employed?

[0079] Q3: The objective function of the KEQA embodiment comprises five terms as shown in Eq. (9). How much does each term contribute?

[0080] 1. Embodiments of Datasets

[0081] In this section, the knowledge graph subsets and question answering dataset used in the experiments are first introduced. All the data are publicly available. Their statistics are shown in Table 2.

TABLE 2

The statistics of the question answering datasets

	FB2M	FB5M	SimpleQuestions
# Training	14,174,246	17,872,174	75,910
# Validation	N.A.	N.A.	10,845
# Test	N.A.	N.A.	21,687
# Predicates (M)	6,701	7,523	1,837
# Entities (N)	1,963,130	3,988,105	131,681
Vocabulary Size	733,278	1,213,205	61,336

[0082] FB2M and FB5M: Freebase is often regarded as a reliable KG since it is collected and trimmed mainly by the community members. Two large subsets of freebase are employed in this paper, i.e., FB2M and FB5M. Their predicate number M and entity number N are list in Table 2. The repeated facts have been deleted. The application programming interface (API) of freebase is no long available. Thus, an entity name collection may be used to build the mapping between entities and their names.

[0083] SimpleQuestions (Borders, et al., Scale Simple Question Answering with Memory Networks. 2015 arXiv preprint: 1506.02075): It contains more than ten thousand simple questions associated with corresponding facts. All these facts belong to FB2M. All questions are phrased by English speakers based on the facts and their context. It has been used as the benchmark for various recent QA-KG methods.

[0084] 2. Experiment Settings

[0085] In one or more embodiments, to evaluate the performance of the QA-KG methods, traditional settings and use the same training, validation and test splits that are originally provided in SimpleQuestions are used. Either FB2M or FB5M is employed as the KG. Then a KG embedding algorithm, such as TransE and TransR, is applied to learn the P and E. It should be noted that P and E are not extra information sources. Then, a QA-KG method is applied to predict the head entity and predicate of each question in the test split. Its performance is measured by the accuracy of predicting both head entity and predicate correctly.

[0086] As claimed in the formal problem definition, the evaluation criterion is defined as the accuracy of predicting a new question' both head entity and predicate correctly. The dimension of the KG embedding representations d is set to be 250. A pre-trained word embedding based on GloVe is used. In one or more embodiments, to measure the similarity of two string, i.e., to build the function sim[.,.], implementation Fuzzy is used. If it is not specific, the KG embedding algorithm TransE would be employed to learn the embedding representations of all predicates P and entities E.

[0087] 3. Effectiveness of the Tested KEQA Embodiments

[0088] The first research question asked at the beginning of this section, i.e., how effective is KEQA, is now answered. In one or more embodiments, 7 state-of-the-art QA-KG algorithms and one variation of KEQA are included as the baselines.

[0089] Bordes et al. (Large Scale Simple Question Answering with Memory Networks. arXiv preprint 1506.02075): It learns latent representations for words, predicates, and entities, based on the training questions, such that a new question and candidate facts could be projected into the same space and compared.

[0090] Dai et al. (CFO: Conditional Focused Neural Question Answering with Large-Scale Knowledge Bases. arXiv preprint arXiv:1606.01994): It employs a bidirectional gated recurrent units based neural network to rank the candidate predicates. Suggestions from the freebase API are used.

[0091] Yin et al. (Simple Question Answering by Attentive Convolutional Neural Network, 2016 COLING. 1746-1756): It employs a character-level convolutional neural network to match the questions and predicates.

[0092] Golub and He (Character-Level Question Answering with Attention. In EMNLP. 1598-1607): It designs a character-level and attention-based LSTM to encode and decode questions.

[0093] Bao et al. (Constraint-Based Question Answering with Knowledge Graph. In COLING. 2503-2514): It manually defines several types of constraints and performs constraint learning to handle complex questions, in which each question is related to several facts. Extra training questions and freebase API are used.

[0094] Lukovnikov et al. (Neural Network-Based Question Answering over Knowledge Graphs on Word and Character Level. In WWW. 1211-1220): It utilizes a character-level gated recurrent units neural network to project questions and predicates/entities into the same space.

[0095] Mohammed et al. (Strong Baselines for Simple Question Answering over Knowledge Graphs with and without Neural Networks, In NAACL-HLT. 291-296): It treats the predicate prediction as a classification problem and uses different neural networks to solve it. It performs entity linking based on Fuzzy.

[0096] KEQA_noEmbed: No KG embedding algorithm is used. Instead, it generates the predicate and entity embedding representations P and E randomly.

[0097] As shown in the introduction above, all the baselines have taken advantage of deep learning models to advance their methods. Their results reported in the corresponding papers or the authors' implementations are used. The performance of different methods on SimpleQuestions with respect to FB2M and FB5M is listed in Table 3.

TABLE 3

Performance of all methods on SimpleQuestions

	FB2M (Accuracy)	FB5M
Bordes et al.	0.627	0.639
Dai et al.	N.A.	0.626
Yin et al.	0.683 (+8.9%)	0.672
Golub and He	0.709 (+13.1%)	0.703
Bao et al.	0.728 (+16.1%)	Entire Freebase
Lukovnikov et al.	0.712 (+13.6%)	N.A.
Mohammed et al.	0.732 (+16.7%)	N.A.
KEQA_noEmbed	0.731 (+16.6%)	0.726
KEQA	0.754 (+20.3%)	0.749

[0098] As mentioned by several other work by Lukovnikov et al. and Mohammed et al., a few algorithms achieve high accuracy, but they either used extra information sources or have no available implementations. The extra training data freebase API suggestions, freebase entity linking results, and trained segmentation models. These rely on the freebase API, which is no longer available. Instead, the presented framework KEQA embodiment uses an entity name collection. Thus, for Dai et al. and Yin et al., their results are reported when no extra training data is used. There are two work claimed much good accuracy, but without publicly available implementations. Thus, it was not possible to replicate them, which has also been pointed out by other work.

[0099] From the results in Table 3, three observations are taken. First, the proposed framework KEQA outperforms all the baselines. KEQA achieves 20.3% improvement comparing to the accuracy when SimpleQuestions was released. Second, KEQA achieves 3.1% higher accuracy compared to KEQA_noEmbed. It demonstrates that the separate task KG embedding indeed could help the question answering task. Third, the performance of KEQA decreases 0.7% when applied to FB5M. It is because all the ground truth facts belong to FB2M, and FB5M has 26.1% more facts than FB2M.

[0100] By jointly predicting the question's predicate and head entity, KEQA achieves an accuracy of 0.754. In the predicate prediction subtask, KEQA achieves an accuracy of 0.815 on the validation split, which is worse than the most recent one 0.828 achieved by Mohammed et al. This gap suggests that the presented KEQA framework in this patent document might be further improved by a more sophisticated model. Nevertheless, KEQA still outperforms Mohammed et al. in the simple question answering task. This confirms the effectiveness of the presented jointly learning framework. Through the jointly learning, KEQA achieves an accuracy of 0.816 in predicting the head entity, 0.754 in predicting both head entity and predicate, and 0.680 in predicting the entire fact, on the test split and FB2M. It implies that some of the ground truth facts do not exist in FB2M.

[0101] 4. Embodiments of Generalizability and Robustness Evaluation

[0102] E.4.1 Generalizability of KEQA.

[0103] In one or more embodiments, to study how general is KEQA when different KG embedding algorithms are used, three scalable KG embedding methods are included in the comparison. Detailed introductions are listed as follows:

[0104] KEQA_TransE: TransE is used to perform the KG embedding. It is a typical translation-based method. It defines the relation function as $e_t \approx f(e_h, \mathbf{p}_\ell) = e_h + \mathbf{p}_\ell$, and then performs the margin-based ranking to make all the facts approach to satisfy the relation function.

[0105] KEQA_TransH: TransH is used to perform the KG embedding. TransH is similar to TransE, and defines the relation function as $e_t^{\perp} \approx e_h^{\perp} + \mathbf{p}_\ell$, where $e_t^{\perp} = e_t - \mathbf{m}_\ell^{\top} \mathbf{e}_t \mathbf{m}_\ell$ and $\mathbf{m}_\ell$ is the hyperplane of predicate ℓ.

[0106] KEQA_TransR: TransR is similar to TransE, and defines the relation function as $e_t \mathbf{M}_\ell \approx e_h \mathbf{M}_\ell + \mathbf{p}_\ell$, where $\mathbf{M}_\ell$ is a transform matrix of ℓ.

[0107] The performance of KEQA when not using the KG embedding and when using different KG embedding algorithms is shown in Table 4. From the results, three major observations are obtained. First, the KG embedding algorithms have improved the performance of KEQA. For example, KEQA achieves 3.1% improvement when it is based on TransE, comparing to KEQA_noEmbed. Second, KEQA has similar performance when using different KG embedding algorithms. It demonstrates the generalizability of KEQA. Third, even when not using the KG embedding, KEQA could still achieve comparable performance to the state-of-the-art QA-KG methods as shown in Table 3. It validates the robustness of KEQA. The reason that randomly-generated P and E could achieve comparable performance is that it tends to make all $\mathbf{p}_\ell$ uniformly distributed and far away from each other. This would convert the representation prediction problem to a one that is similar to the classification task.

TABLE 4

The performance of KEQA with different knowledge graph embedding algorithm on FB2M

	SimpleQuestions	SimpleQ_Missing
KEQA_noEmbed	0.731	0.386
KEQA_TransE	0.754 (+3.1%)	0.418 (+8.3%)
KEQA_TransH	0.749 (+2.5%)	0.411 (+6.5%)
KEQA_TransR	0.753 (+3.0%)	0.417 (+8.0%)

[0108] 4.2 Robustness of KEQA.

[0109] To further validate the robustness of KEQA, all the 108,442 questions in SimpleQuestions are reshuffled and a new dataset named SimpleQ_Missing is obtained. In one or more embodiments, to perform the reshuffle, all the types of predicates are randomly split into three groups, and assign questions to these groups based on the predicates. Thus, in SimpleQ_Missing, all the corresponding predicates of the questions in the test split have never been mentioned in the training and validation splits. In the end, 75,474 questions in the training split, 11,017 questions in the validation split, and 21,951 questions in the test split are obtained, which are roughly the same ratios as the ones in SimpleQuestions. The performance of KEQA with different KG embedding algorithms on SimpleQ_Missing is shown in Table 4.

[0110] From the results in Table 4, it is observed that KEQA could still achieve an accuracy of 0.418 with the help of TransE. The global relation and structure information preserved in the KG embedding representations P and E enables KEQA to perform 8.3% better than Random. These observations demonstrate the robustness of KEQA.

[0111] 5. Embodiments of Parameter Analysis

[0112] In this section, investigation is carried out on how much could each term in the objective function of KEQA contribute. There are five terms in the objective function as shown in Eq. (9). In one or more embodiments, the performance of KEQA with respect to three groups of different combinations of terms is investigated. To study the contribution of every single term in Eq. (9), in the first group, i.e., Only_Keep, only one of the five terms is kept as the new objective function. To study the impact of missing one of the five terms, in the second group, i.e., Remove, one of the five terms is removed. To study the accumulated contributions, in the third group, i.e., Accumulate, terms area added as the new objective function one by one. The performance of KEQA with respect to different groups of objective functions on FB2M is summarized in Table 5.

TABLE 5

The performance of the KEQA embodiment with different objective functions on FB2M

	Only_Keep	Remove	Accumulate
$\|\mathbf{p}_\ell - \hat{\mathbf{p}}_\ell\|_2$	0.728	0.701	0.728
$\|P_h - \hat{e}_h\|_2$	0.195	0.751	0.745
$\|f(e_h,\mathbf{p}_\ell) - \hat{e}_t\|_2$	0.730	0.753	0.745
sim[n(h), HED_{entity}]	0.173	0.754	0.746
sim[n(ℓ), HED_{non}]	0.435	0.746	0.754

[0113] From the results in Table 5, three major observations are noted. First, the predicted predicate representation $\mathbf{p}_\ell$ has the most significant contribution in the presented framework. The first term achieves an accuracy of 0.728 independently. It is because the number of predicates 1,837 is much smaller than the number of training questions 75,910. Second, the predicted head entity representation $\hat{e}_h$ could complement $\mathbf{p}_\ell$ in the joint learning. The accuracy increases from 0.728 to 0.745 when $\hat{e}_h$ is used. The second term achieves a low accuracy independently since the total number of entities N is too large, e.g., N=1,963,115 in FB2M. Third, the predicate name n(ℓ) improves the performance of the KEQA by 1.1%. It could be explained by the fact that some utterances share a few words with the corresponding predicate names.

F. Some Conclusions

[0114] Question answering over knowledge graph is a crucial problem since it enables regular users to easily access the valuable but complex information in the large knowledge graphs via natural language. It is also a challenging problem since a predicate could have different natural language expressions. It is hard for a machine to capture their semantic information. In addition, even assuming that the entity name of a question is correctly identified, the ambiguity of entity names and partial names would still make the number of candidate entities large.

[0115] To bridge the gap, embodiments of a novel knowledge graph embedding based question answering problem are disclosed herein and embodiments of a simple and effective KEQA framework are presented. The KEQA framework targets solving simple questions, i.e., the most common type of question in QA-KG. Instead of inferring the head entity and predicate directly, KEQA jointly recovers the question's head entity, predicate, and tail entity representations in the KG embedding spaces. In one or more embodiments, attention-based bidirectional LSTM models are employed to perform the predicate and head entity representation learning. Since it is expensive and noisy to comparing with all entities in a KG, a head entity detection model is used to select successive tokens in a question as the name of the head entity, such that candidate head entity set would be reduced to a number of entities with the same or similar names. Given the predicted fact $\hat{e}_h$, $\hat{\mathbf{p}}_\ell$, $\hat{e}_t$, embodiments of a carefully-designed joint distance metric are used to measure its distances to all candidate facts. The fact with the minimum distance is returned as the answer. Comprehensive experiments were conducted to evaluate the performance of the presented KEQA framework embodiments. Experiments on a large benchmark demonstrate that KEQA embodiments achieve better performance than state-of-the-art methods.

[0116] In one or more embodiments, the KEQA framework embodiments may be extended in various scenarios. The extension includes but not limits to (i) KEQA embodiments performing the question answering based on the pre-trained KG embedding. KEQA may be advanced by jointly conducting the KG embedding and question answering. (ii) Real-world knowledge graphs and training questions are often updated dynamically. KEQA framework embodiments may be extended to handle such a scenario.

G. System Embodiments

[0117] In embodiments, aspects of the present patent document may be directed to, may include, or may be implemented on one or more information handling systems/computing systems. A computing system may include any instrumentality or aggregate of instrumentalities operable to compute, calculate, determine, classify, process, transmit, receive, retrieve, originate, route, switch, store, display, communicate, manifest, detect, record, reproduce, handle, or utilize any form of information, intelligence, or data. For example, a computing system may be or may include a personal computer (e.g., laptop), tablet computer, phablet, personal digital assistant (PDA), smart phone, smart watch, smart package, server (e.g., blade server or rack server), a network storage device, camera, or any other suitable device and may vary in size, shape, performance, functionality, and price. The computing system may include random access memory (RAM), one or more processing resources such as a central processing unit (CPU) or hardware or software control logic, ROM, and/or other types of memory. Additional components of the computing system may include one or more disk drives, one or more network ports for communicating with external devices as well as various input and output (I/O) devices, such as a keyboard, a mouse, touchscreen and/or a video display. The computing system may also include one or more buses operable to transmit communications between the various hardware components.

[0118] FIG. **8** depicts a simplified block diagram of a computing device/information handling system (or computing system) according to embodiments of the present disclosure. It will be understood that the functionalities shown for system **800** may operate to support various embodiments of a computing system—although it shall be understood that a computing system may be differently configured and include different components, including having fewer or more components as depicted in FIG. **8**.

[0119] As illustrated in FIG. **8**, the computing system **800** includes one or more central processing units (CPU) **801** that provides computing resources and controls the computer. CPU **801** may be implemented with a microprocessor or the like, and may also include one or more graphics processing units (GPU) **819** and/or a floating-point coprocessor for mathematical computations. System **800** may also include a system memory **802**, which may be in the form of random-access memory (RAM), read-only memory (ROM), or both.

[0120] A number of controllers and peripheral devices may also be provided, as shown in FIG. **8**. An input controller **803** represents an interface to various input device (s) **804**, such as a keyboard, mouse, touchscreen, and/or stylus. The computing system **800** may also include a storage controller **807** for interfacing with one or more storage devices **808** each of which includes a storage medium such as magnetic tape or disk, or an optical medium that might be used to record programs of instructions for operating systems, utilities, and applications, which may include embodiments of programs that implement various aspects of the present invention. Storage device(s) **808** may also be used to store processed data or data to be processed in accordance with the invention. The system **800** may also include a display controller **809** for providing an interface to a display device **811**, which may be a cathode ray tube (CRT), a thin film transistor (TFT) display, organic light-emitting diode, electroluminescent panel, plasma panel, or other type of display. The computing system **800** may also include one or more peripheral controllers or interfaces **805** for one or more peripherals **806**. Examples of peripherals may include one or more printers, scanners, input devices, output devices, sensors, and the like. A communications controller **814** may interface with one or more communication devices **815**, which enables the system **800** to connect to remote devices through any of a variety of networks including the Internet, a cloud resource (e.g., an Ethernet cloud, an Fiber Channel over Ethernet (FCoE)/Data Center Bridging (DCB) cloud, etc.), a local area network (LAN), a wide area network (WAN), a storage area network (SAN) or through any suitable electromagnetic carrier signals including infrared signals.

[0121] In the illustrated system, all major system components may connect to a bus **816**, which may represent more than one physical bus. However, various system components may or may not be in physical proximity to one another. For example, input data and/or output data may be remotely transmitted from one physical location to another. In addition, programs that implement various aspects of the invention may be accessed from a remote location (e.g., a server) over a network. Such data and/or programs may be conveyed through any of a variety of machine-readable medium including, but are not limited to: magnetic media such as hard disks, floppy disks, and magnetic tape; optical media such as CD-ROMs and holographic devices; magneto-optical media; and hardware devices that are specially configured to store or to store and execute program code, such as application specific integrated circuits (ASICs), programmable logic devices (PLDs), flash memory devices, and ROM and RAM devices.

[0122] Aspects of the present invention may be encoded upon one or more non-transitory computer-readable media with instructions for one or more processors or processing units to cause steps to be performed. It shall be noted that the one or more non-transitory computer-readable media shall include volatile and non-volatile memory. It shall be noted that alternative implementations are possible, including a hardware implementation or a software/hardware implementation. Hardware-implemented functions may be realized using ASIC(s), programmable arrays, digital signal processing circuitry, or the like. Accordingly, the "means" terms in any claims are intended to cover both software and hardware implementations. Similarly, the term "computer-readable medium or media" as used herein includes software and/or hardware having a program of instructions embodied thereon, or a combination thereof. With these implementation alternatives in mind, it is to be understood that the figures and accompanying description provide the functional information one skilled in the art would require to write program code (i.e., software) and/or to fabricate circuits (i.e., hardware) to perform the processing required.

[0123] It shall be noted that embodiments of the present invention may further relate to computer products with a non-transitory, tangible computer-readable medium that have computer code thereon for performing various computer-implemented operations. The media and computer code may be those specially designed and constructed for the purposes of the present invention, or they may be of the kind known or available to those having skill in the relevant arts. Examples of tangible computer-readable media include, but are not limited to: magnetic media such as hard disks, floppy disks, and magnetic tape; optical media such as CD-ROMs and holographic devices; magneto-optical media; and hardware devices that are specially configured to store or to store and execute program code, such as application specific integrated circuits (ASICs), programmable logic devices (PLDs), flash memory devices, and ROM and RAM devices. Examples of computer code include machine code, such as produced by a compiler, and files containing higher level code that are executed by a computer using an interpreter. Embodiments of the present invention may be implemented in whole or in part as machine-executable instructions that may be in program modules that are executed by a processing device. Examples of program modules include libraries, programs, routines, objects, components, and data structures. In distributed computing environments, program modules may be physically located in settings that are local, remote, or both.

[0124] One skilled in the art will recognize no computing system or programming language is critical to the practice of the present invention. One skilled in the art will also recognize that a number of the elements described above may be physically and/or functionally separated into sub-modules or combined together.

[0125] It will be appreciated to those skilled in the art that the preceding examples and embodiments are exemplary and not limiting to the scope of the present disclosure. It is intended that all permutations, enhancements, equivalents, combinations, and improvements thereto that are apparent to those skilled in the art upon a reading of the specification and a study of the drawings are included within the true spirit and scope of the present disclosure. It shall also be noted that elements of any claims may be arranged differently including having multiple dependencies, configurations, and combinations.

What is claimed is:

1. A computer-implemented method for question answering using one or more processors to cause steps to be performed comprising:

generating, using a predicate learning model, a predicted predicate representation in a knowledge graph (KG) predicate embedding space for a question comprising one or more tokens;

generating, using a head entity learning model, a predicted head entity representation in an KG entity embedding space for the question;

obtaining a predicted tail entity representation, based on a relation function that relates, for a fact in KG embedding space, a head entity representation and a predicate representation to a tail entity representation, from the predicted predicate representation and the predicted head entity representation, the predicted predicate representation, the predicted head entity representation, and the predicted tail entity representation forming a predicted fact;

identifying, using a head entity detection (HED) model, one or more predicted head entity names for the question, each predicted head entity name comprises one or more tokens from the question;

searching, in the KG, head entity synonyms related to the one or more predicted head entity names;

constructing a candidate fact set comprising one or more candidate facts, each candidate fact comprises a head entity from among the head entity synonyms; and

choosing, based on a joint distance metric, one candidate fact in the candidate fact set with a minimum joint distance to the predicted fact as an answer to the question.

2. The computer-implemented method of claim **1** wherein the predicate learning model has a neural network structure comprising a bidirectional recurrent neural network layer and an attention layer, the generation of the predicted predicate representation comprising:

mapping the one or more tokens in the question into a sequence of word embedding vectors;

generating, using the bidirectional recurrent neural network layer, a forward hidden state sequence and a backward hidden state sequence;

concatenating the forward and backward hidden state vectors into a concatenated hidden state vector;

applying, by the attention layer, an attention weight to the concatenated hidden state vector to obtain a weighted hidden state vector;

concatenating the weighted hidden state vector with the word embedding to obtain a hidden state for each token;

applying a fully connected layer to the hidden state to obtain a target vector for each token; and

using a mean of all target vectors as the predicted predicate representation.

3. The computer-implemented method of claim **2** wherein the head entity learning model have a neural network structure the same as the predicate learning model.

4. The computer-implemented method of claim **3** wherein the predicate learning model and the head entity learning model are pre-trained using a training data set with ground truth facts via a predicate objective function and a head entity objective function respectively.

5. The computer-implemented method of claim **1** wherein the HED model has a neural network structure comprising a bidirectional recurrent neural network layer and a fully connecter layer, the identification of the one or more predicted head entity names for the question comprising:

mapping the one or more tokens in the question into a sequence of word embedding vectors;

generating, at the bidirectional recurrent neural network layer, a forward hidden state sequence and a backward hidden state sequence;

concatenating the forward and backward hidden state vectors to obtain a concatenated hidden state vector;

applying the fully connected layer and a Softmax function to the concatenated hidden state vector to obtain a target vector for each token, each target vector has two probability values corresponding to probabilities that the token belongs to entity token name and non-entity token name; and

selecting one or more tokens as the head entity name based on probability value of each token belonging to entity token name.

6. The computer-implemented method of claim **1** wherein the joint distance metric comprises distance terms representing distance between a vector in the candidate fact and a corresponding vector in the predicted fact, each term is a ℓ^2 norm to measure vector distance.

7. The computer-implemented method of claim **6** wherein the joint distance metric further comprises string similarity terms representing string similarity between name of entity in the candidate fact and the tokens classified as entity name by the HED model, and string similarity between name of the predicate in the candidate fact and the tokens classified as non entity name by the HED model.

8. The computer-implemented method of claim **7** wherein the joint distance metric is a weighted combination of the distance terms and the string similarity terms.

9. The computer-implemented method of claim **6** wherein in the joint distance metric, the candidate fact has a tail entity embedding vector calculated, using the relation function, from a head entity embedding vector and a predicate embedding vector of the candidate fact.

10. The computer-implemented method of claim **1** wherein searching head entity synonyms in the KG related to the one or more predicted head entity names comprising:

inputting entity vector for each head entity name into the KG; and

searching, in the KG, head entity synonyms with corresponding token embedding, by both embedding comparison and string match, each head entity synonym has direct or partial string match to the head entity name, or has embedding similarity to the entity vector.

11. The computer-implemented method of claim **10** wherein for head entity name comprising multiple tokens, the entity vector is combined from a dot product of entity vectors of each token.

12. A computer-implemented method for question answering using one or more processors that cause steps to be performed comprising:

generating, using a predicate learning model stored in one or more memories of one or more computing devices, a predicted predicate representation for a question comprising one or more tokens in a predicate embedding space, the predicate learning model being pre-trained using training data with ground truth facts and a predicate objective function;

generating, using a head entity learning model stored in one or more memories of one or more computing devices, a predicted head entity representation for the question in an entity embedding space, head entity learning model being pre-trained using training data with ground truth facts and a head entity objective function;

identifying, using a relation function based upon knowledge graph (KG) embedding, a predicted tail entity presentation from the predicted predicate representation and the predicted head entity presentation, the predicted head entity representation, the predicted predicate representation, and the predicted tail entity representation forming a predicted fact; and

selecting a fact from among at least a subset of facts in the KG, based on a joint distance metric, as answer to the question, the selected fact having a minimum joint distance between it and the predicted fact according to the joint distance metric.

13. The computer-implemented method of claim **12** wherein the at least a subset is a candidate fact set comprising one or more candidate facts chosen from the one or more facts in the KG, each candidate fact comprises a head entity as a synonym for one or more predicted head entity names identified by a head entity detection (HED) model comprising at least a bidirectional recurrent neural network layer and a fully connected layer.

14. The computer-implemented method of claim **13** wherein the one or more predicted head entity names are identified by the HED model by steps comprising:

generating, using the bidirectional recurrent neural network layer, a forward hidden state sequence and a backward hidden state sequence from a sequence of word embedding vectors of the one or more tokens in the question;

concatenating the forward and backward hidden state vectors into a concatenated hidden state vector;

applying at least the fully connected layer to the concatenated hidden state vector to obtain a target vector for each token, each target vector has two probability values corresponding to probabilities that the token belongs to entity token name and non-entity token name; and

selecting one or more tokens as the head entity name based on probability value of each token belonging to entity token name.

15. The computer-implemented method of claim **13** wherein the joint distance metric comprises vector distance terms representing ℓ^2 norm of vector distance between a vector in the candidate fact and a corresponding vector in the predicted fact, and string similarity terms representing string similarity between name of entity in the candidate fact and the tokens classified as entity name by the HED model, and string similarity between name of the predicate in the candidate fact and the tokens classified as non entity name by the HED model.

16. The computer-implemented method of claim **15** wherein the joint distance metric is a weighted combination of the vector distance terms and the string similarity terms with a weight for each term in the joint distance metric.

17. A non-transitory computer-readable medium or media comprising one or more sequences of instructions which, when executed by one or more processors, causes the steps for question answering to be performed comprising:

generating a vector in a knowledge graph (KG) predicate embedding space as a predicted predicate representation for a question comprising one or more tokens;

generating a vector in a KG entity embedding space as a predicted head entity representation for the question;

obtaining a predicted tail entity representation, based on a relation function based upon knowledge graph (KG) embedding, from the predicted predicate representation and the predicted head entity presentation, the predicted predicate representation, and the predicted tail entity representation forming a predicted fact;

identifying one or more predicted head entity names for the question, each predicted head entity name comprises one or more tokens from the question;

searching, in the KG, head entity synonyms to the one or more predicted head entity names by both embedding comparison and string match;

constructing a candidate fact set comprising one or more candidate facts, each candidate fact comprises a head entity among the head entity synonyms; and

choosing one candidate fact in the candidate fact set with a minimum joint distance to the predicted fact based on a joint distance metric as an answer to the question.

18. The non-transitory computer-readable medium or media of claim **17** wherein the joint distance metric comprises vector distance terms representing ℓ^2 norm of vector distance between a vector in the candidate fact and a corresponding vector in the predicted fact, and string similarity terms representing string similarity between entity name of candidate fact and entity tokens in the question, and string similarity between predicate name of candidate fact and non-entity tokens in the question.

19. The non-transitory computer-readable medium or media of claim **18** wherein the joint distance metric is a weighted combination of the vector distance terms and the string similarity terms.

20. The non-transitory computer-readable medium or media of claim **19** wherein in the joint distance metric, the string similarity terms counterweight the vector distance terms.

* * * * *

(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 111506714 A
(43)申请公布日 2020.08.07

(21)申请号 201911148232.0

(22)申请日 2019.11.21

(30)优先权数据
16/262,618 2019.01.30 US

(71)申请人 百度(美国)有限责任公司
地址 美国加利福尼亚州

(72)发明人 张婧媛　李定成　李平　黄啸

(74)专利代理机构 北京英赛嘉华知识产权代理有限责任公司 11204
代理人 马晓亚　王艳春

(51)Int.Cl.
G06F 16/332(2019.01)
G06F 16/35(2019.01)
G06F 16/36(2019.01)

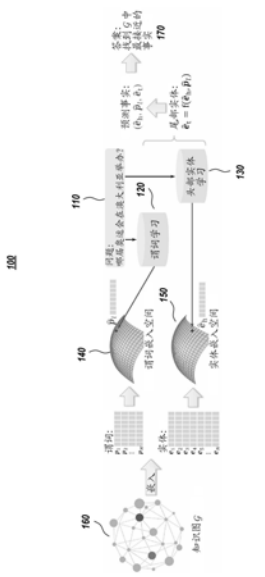
权利要求书4页　说明书20页　附图8页

(54)发明名称

基于知识图嵌入的问题回答

(57)摘要

本文描述了使用基于知识嵌入的问题回答(KEQA)框架在知识图上回答问题的实施方式。KEQA实施方式的目标不是直接推断输入问题的头部实体和谓词,而是在KG嵌入空间中联合恢复问题的头部实体、谓词和尾部实体表示。在实施方式中,使用结合各种损失项的联合距离度量来测量预测事实到所有候选事实的距离。在实施方式中,返回具有最小距离的事实作为答案。还公开了联合训练策略的实施方式以获得更好的性能。对各种数据集的性能评估证明了所公开的使用KEQA框架的系统和方法的有效性。



1.一种使用一个或多个处理器进行问题回答的由计算机实施的方法,所述处理器使得执行步骤,所述步骤包括:

使用谓词学习模型在知识图谓词嵌入空间中生成用于包括一个或多个令牌的问题的预测谓词表示;

使用头部实体学习模型,在知识图实体嵌入空间中生成用于所述问题的预测头部实体表示;

基于关系函数,从所述预测谓词表示和所述预测头部实体表示获取预测尾部实体表示,所述预测谓词表示、所述预测头部实体表示和所述预测尾部实体表示形成预测事实,所述关系函数针对知识图嵌入空间中的事实将头部实体表示和谓词表示与尾部实体表示关联;

使用头部实体检测模型识别用于所述问题的一个或多个预测头部实体名称,每个预测头部实体名称包括来自所述问题的一个或多个令牌;

在所述知识图中搜索与所述一个或多个预测头部实体名称相关的头部实体同义词;

构建包括一个或多个候选事实的候选事实集,每个候选事实包括所述头部实体同义词中的头部实体;以及

基于联合距离度量,选择所述候选事实集中与所述预测事实具有最小联合距离的一个候选事实作为所述问题的答案。

2.根据权利要求1所述的由计算机实施的方法,其中所述谓词学习模型具有包括双向递归神经网络层和注意层的神经网络结构,所述预测谓词表示的生成包括:

将所述问题中的所述一个或多个令牌映射到单词嵌入向量的序列中;

使用所述双向递归神经网络层生成前向隐藏状态序列和后向隐藏状态序列;

将所述前向隐藏状态序列和所述后向隐藏状态序列串联成串联隐藏状态向量;

由所述注意层对所述串联隐藏状态向量应用注意权重,以获得加权隐藏状态向量;

将所述加权隐藏状态向量与所述单词嵌入向量串联,以获得每个令牌的隐藏状态;

对所述隐藏状态应用完全连接层以获得每个令牌的目标向量;以及

使用所有目标向量的平均值作为所述预测谓词表示。

3.根据权利要求2所述的由计算机实施的方法,其中所述头部实体学习模型具有与所述谓词学习模型相同的神经网络结构。

4.根据权利要求3所述的由计算机实施的方法,其中分别通过谓词目标函数和头部实体目标函数,使用具有真实事实的训练数据集对所述谓词学习模型和所述头部实体学习模型进行预训练。

5.根据权利要求1所述的由计算机实施的方法,其中所述头部实体检测模型具有包括双向递归神经网络层和完全连接层的神经网络结构,所述问题的所述一个或多个预测头部实体名称的识别包括:

将所述问题中的所述一个或多个令牌映射到单词嵌入向量的序列中;

在所述双向递归神经网络层处生成前向隐藏状态序列和后向隐藏状态序列;

串联所述前向隐藏状态向量和后向隐藏状态向量以获得串联隐藏状态向量;

对所述串联隐藏状态向量应用所述完全连接层和Softmax函数,以获得每个令牌的目标向量,每个目标向量具有两个概率值,所述两个概率值对应于所述令牌属于实体令牌名

称和非实体令牌名称的概率;以及

基于每个令牌属于实体令牌名称的概率值,选择一个或多个令牌作为所述头部实体名称。

6.根据权利要求1所述的由计算机实施的方法,其中所述联合距离度量包括表示所述候选事实中的向量和所述预测事实中的相应向量之间的距离的距离项,每个距离项是测量向量距离的l^2范数。

7.根据权利要求6所述的由计算机实施的方法,其中所述联合距离度量还包括字符串相似性项,所述字符串相似性项表示所述候选事实中的实体名称和由所述头部实体检测模型分类为实体名称的令牌之间的字符串相似性,以及所述候选事实中的所述谓词的名称和由所述头部实体检测模型分类为非实体名称的令牌之间的字符串相似性。

8.根据权利要求7所述的由计算机实施的方法,其中所述联合距离度量是所述距离项和所述字符串相似性项的加权组合。

9.根据权利要求6所述的由计算机实施的方法,其中在所述联合距离度量中,所述候选事实具有尾部实体嵌入向量,所述尾部实体嵌入向量使用所述关系函数从所述候选事实的头部实体嵌入向量和谓词嵌入向量计算得到。

10.根据权利要求1所述的由计算机实施的方法,其中在所述知识图中搜索与所述一个或多个预测头部实体名称相关的头部实体同义词包括:

将每个头部实体名称的实体向量输入到所述知识图中;以及

在所述知识图中,通过嵌入比较和字符串匹配来搜索具有相应令牌嵌入的头部实体同义词,每个头部实体同义词具有与所述头部实体名称的直接字符串匹配或部分字符串匹配,或者具有与所述实体向量的嵌入相似性。

11.根据权利要求10所述的由计算机实施的方法,其中对于包括多个令牌的头部实体名称,从每个令牌的实体向量的点积组合得到所述实体向量。

12.一种使用一个或多个处理器进行问题回答的由计算机实施的方法,所述一个或多个处理器使得执行步骤,所述步骤包括:

使用存储在一个或多个计算设备的一个或多个存储器中的谓词学习模型,在谓词嵌入空间中生成用于包括一个或多个令牌的问题的预测谓词表示,所述谓词学习模型使用具有真实事实的训练数据和谓词目标函数进行预训练;

使用存储在一个或多个计算设备的一个或多个存储器中的头部实体学习模型,在实体嵌入空间中生成用于所述问题的预测头部实体表示,头部实体学习模型使用具有真实事实的训练数据和头部实体目标函数进行预训练;

使用基于知识图嵌入的关系函数,从所述预测谓词表示和所述预测头部实体表示识别预测尾部实体表示,所述预测头部实体表示、所述预测谓词表示和所述预测尾部实体表示形成预测事实;以及

基于联合距离度量,从所述知识图中的事实的至少一个子集中选择事实作为所述问题的答案,所选择的事实根据所述联合距离度量在其与所述预测事实之间具有最小联合距离。

13.根据权利要求12所述的由计算机实施的方法,其中所述至少一个子集是候选事实集,所述候选事实集包括从所述知识图中的一个或多个事实中选择的一个或多个候选事

实,每个候选事实包括头部实体,所述头部实体作为由头部实体检测模型识别的一个或多个预测头部实体名称的同义词,所述头部实体检测模型至少包括双向递归神经网络层和完全连接层。

14.根据权利要求13所述的由计算机实施的方法,其中由所述头部实体检测模型通过包括以下的步骤来识别所述一个或多个预测头部实体名称:

使用所述双向递归神经网络层,从所述问题中的所述一个或多个令牌的单词嵌入向量的序列生成前向隐藏状态序列和后向隐藏状态序列;

将所述前向隐藏状态序列和后向隐藏状态序列串联成串联隐藏状态向量;

对所述串联隐藏状态向量至少应用所述完全连接层以获得用于每个令牌的目标向量,每个目标向量具有与所述令牌属于实体令牌名称和非实体令牌名称的概率相对应的两个概率值;以及

基于每个令牌属于实体令牌名称的概率值,选择一个或多个令牌作为所述头部实体名称。

15.根据权利要求13所述的由计算机实施的方法,其中所述联合距离度量包括表示所述候选事实中的向量和所述预测事实中的相应向量之间的向量距离的1^2范数的向量距离项,以及表示所述候选事实中的实体名称和由所述头部实体检测模型分类为实体名称的令牌之间的字符串相似性和所述候选事实中的所述谓词的名称和由所述头部实体检测模型分类为非实体名称的令牌之间的字符串相似性的字符串相似性项。

16.根据权利要求15所述的由计算机实施的方法,其中所述联合距离度量是所述向量距离项和所述字符串相似性项的加权组合,其中所述加权组合具有用于所述联合距离度量中每个项的权重。

17.一种包括一个或多个指令序列的非暂时性计算机可读介质,当由一个或多个处理器执行时,该指令序列使得执行用于问题回答的步骤,所述步骤包括:

在知识图谓词嵌入空间中生成向量,作为包括一个或多个令牌的问题的预测谓词表示;

在知识图实体嵌入空间中生成向量作为所述问题的预测头部实体表示;

基于根据知识图嵌入的关系函数,从所述预测谓词表示和所述预测头部实体表示获得预测尾部实体表示,所述预测谓词表示和所述预测尾部实体表示形成预测事实;

识别所述问题的一个或多个预测头部实体名称,每个预测头部实体名称包括来自所述问题的一个或多个令牌;

通过嵌入比较和字符串匹配,在所述知识图中搜索所述一个或多个预测头部实体名称的头部实体同义词;

构建包括一个或多个候选事实的候选事实集,每个候选事实包括所述头部实体同义词中的头部实体;以及

基于联合距离度量选择所述候选事实集中与所述预测事实具有最小联合距离的一个候选事实作为所述问题的答案。

18.根据权利要求17所述的非暂时性计算机可读介质,其中所述联合距离度量包括表示所述候选事实中的向量和所述预测事实中的相应向量之间的向量距离的1^2范数的向量距离项,以及表示候选事实的实体名称和所述问题中的实体令牌之间的字符串相似性以及

候选事实的谓词名称和所述问题中的非实体令牌之间的字符串相似性的字符串相似性项。

19.根据权利要求18所述的非暂时性计算机可读介质，其中所述联合距离度量是所述向量距离项和所述字符串相似性项的加权组合。

20.根据权利要求19所述的非暂时性计算机可读介质，其中在所述联合距离度量中，所述字符串相似性项平衡所述向量距离项。

基于知识图嵌入的问题回答

技术领域

[0001] 共

背景技术

[0002] (- G) (G)
G
- G
/ G
共

[0003]

发明内容

[0004]
问
问
问
共

[0005]
问
共
问
共

事实;以及基于联合距离度量,从所述 知识图中的事实的至少一个子集中选择事实作为所述问题的答案,所 选择的事实根据所述联合距离度量在其与所述预测事实之间具有最小联合距离。

[0006] 根据本申请的又一方面,公开了一种包括一个或多个指令序列的 非暂时性计算机可读介质,当由一个或多个处理器执行时,该指令序 列使得执行用于问题回答的步骤,所述步骤包括:在知识图谓词嵌入 空间中生成向量,作为包括一个或多个令牌的问题的预测谓词表示; 在知识图实体嵌入空间中生成向量作为所述问题的预测头部实体表 示;基于根据知识图嵌入的关系函数,从所述预测谓词表示和所述预 测头部实体表示获得预测尾部实体表示,所述预测谓词表示和所述预 测尾部实体表示形成预测事实;识别所述问题的一个或多个预测头部 实体名称,每个预测头部实体名称包括来自所述问题的一个或多个令 牌;通过嵌入比较和字符串匹配,在所述知识图中搜索所述一个或多 个预测头部实体名称的头部实体同义词;构建包括一个或多个候选事 实的候选事实集,每个候选事实包括所述头部实体同义词中的头部实 体;以及基于联合距离度量选择所述候选事实集中与所述预测事实具 有最小联合距离的一个候选事实作为所述问题的答案。

附图说明

[0007] 将参考本发明的实施方式,其示例可以在附图中示出。这些附图 是说明性的,而不是限制性的。尽管本发明通常在这些实施方式的上 下文中描述,但是应当理解,这并不意味着将本发明的范围限制于这 些特定实施方式。图中的项目未按比例绘制。

[0008] 图1图示了根据本公开的实施方式的基于知识嵌入的问题回答 (KEQA)框架。

[0009] 图2示出了根据本公开的实施方式的利用KEQA框架回答问题的 方法。

[0010] 图3图示了根据本公开的实施方式的谓词和头部实体学习模型的 架构。

[0011] 图4示出了根据本公开的实施方式的使用谓词和头部实体学习模 型预测输入问题的谓词的方法。

[0012] 图5示出了根据本公开的实施方式的头部实体检测(HED)模型 的结构。

[0013] 图6示出了根据本公开的实施方式的使用HED模型识别输入问题 的一个或多个头部实体的方法。

[0014] 图7示出了根据本公开的实施方式的使用由HED模型识别的头部 实体名称在KG中搜索头部实体同义词的方法。

[0015] 图8示出了根据本文献的实施方式的计算设备/信息处理系统的简 化框图。

具体实施方式

[0016] 在以下描述中,出于解释的目的,阐述了具体细节以提供对本公 开的理解。然而,对于本领域技术人员来说显而易见的是,没有这些 细节也可以实施实施方式。此外,本领域技术人员将认识到,下面描 述的本公开的实施方式可以以多种方式实现,例如以有形计算机可读 介质上的过程、装置、系统、设备或方法实现。

[0017] 图中所示的组件或模块是本发明的示例性实施方式的说明,并且 旨在避免模糊本公开。还应当理解,在整个讨论中,组件可以被描述 为单独的功能单元,其可以包括子单元,但是本领域技术人员将认识 到,各种组件或其部分可以被分成单独的组件或者可以集

成在一起，包括集成在单个系统或组件中。应当注意，这里讨论的功能或操作可 以实现为组件。组件可以用软件、硬件或其组合来实现。

[0018] 此外，图中组件或系统之间的连接并不旨在局限于直接连接。相 反，这些组件之间的数据可以由中间组件修改、重新格式化或以其他 方式改变。此外，可以使用更多或更少的连接。还应注意，术语“耦 合”、“连接”或“通信耦合”应理解为包括直接连接、通过一个或多 个中间设备的间接连接以及无线连接。

[0019] 说明书中对“一个实施方式”、“优选实施方式”、“实施方式”或 “一些实施方式”的引用意味着结合该实施方式描述的特定特征、结 构、特性或功能包括在本发明的至少一个实施方式中，并且可以在不 止一个实施方式中。此外，说明书中不同地方出现的上述短语不一定 都指相同的实施方式。

[0020] 在说明书的不同地方使用某些术语是为了说明，不应被解释为限 制。服务、功能或资源不限于单一的服务、功能或资源；这些术语的 使用可以指相关服务、功能或资源的分组，这些服务、功能或资源可 以是分布式的或聚合的。图像可以是静止图像或视频中的图像。

[0021] 术语“包括(include)”、“包括(including)”、“包含(comprise)” 和“包含(comprising)”应理解为开放式术语，以下任何列表都是示 例，并不意味着限于所列项目。本文中使用的任何标题仅用于组织目 的，不应用于限制说明书或权利要求的范围。该专利文献中提到的每 个参考文献都通过引用整体并入本文。

[0022] 此外，本领域技术人员应该认识到：(1)可以可选地执行某些步 骤；(2)步骤可以不限于本文所述的特定顺序；(3)某些步骤可以以 不同的顺序执行；以及(4)某些步骤可以同时进行。

[0023] A.引言

[0024] 随着诸如Wikidata、Freebase、Dbpedia和YAGO等大规模知识图 的兴起，知识图问题回答(QA)已经成为一个至关重要的话题，并引 起了广泛的关注。知识图(KG)通常是一个有向图，现实世界的实体 作为节点，它们的关系作为边。在这个图中，每条有向边连同其头部 实体和尾部实体构成一个三元组，即(头部实体，谓词，尾部实体)，其也被命名为事实。现实世界的知识图可能包含数百万或数十亿个事 实。它们庞大的数据量和复杂的数据结构使得普通用户很难获取其中 大量有价值的知识。为了弥补这一差距，提出了知识图问题回答 (QA-KG)。它的目标是尝试将最终用户的自然语言问题自动翻译成 结构化查询，如SPARQL，并返回KG中的实体和/或谓词作为答案。例如，在给定“哪届奥运会在澳大利亚举办?”这个问题的情况下，QA-KG旨在确定其相应的两个事实，即(澳大利亚，奥运会_参与，1952/2004年夏季奥运会)。

[0025] 知识图问题回答为人工智能系统提供了一种方法，将知识图作为 回答人类问题的关键要素，应用范围从搜索引擎设计到会话代理构建。然而，QA-KG问题远未解决，因为它涉及多个具有挑战性的子问题，如语义分析和实体链接。

[0026] 知识图嵌入在不同的现实世界应用中的有效性激发了探索其在本 专利文献中解决QA-KG问题中的潜在用途。知识图嵌入的目标是学 习KG中每个谓词/实体的低维向量表示，以便原始关系在向量中得到 很好的保留。这些学习到的向量表示可以用来高效地完成各种下游应 用。示例包括KG完成、推荐系统和关系提取。在本专利文献中，呈 现了知识图

嵌入的实施方式来执行QA-KG。KG嵌入表示可以以多种 方式推动QA-KG。它们不仅在低维空间内，而且可以促进下游应用以 考虑到整个KG，因为即使是单个谓词/实体表示也是与整个KG交互 的结果。此外，相似的谓词/实体往往有相似的向量。该属性可用于帮 助下游算法处理不在训练数据中的谓词或实体。

[0027] 然而，基于知识图嵌入进行QA-KG仍然是一项不平凡的任务。有三大挑战。首先，谓词在自然语言问题中通常有多种表达方式。这 些表达方式可能与谓词名称有很大不同。例如，谓词“人.国籍”可以 表达为“……是什么国籍”、“……来自哪个国家”、“……来自哪里” 等等。其次，即使假设实体名称可以被精确识别，实体名称和部分名 称的模糊性仍然会使找到正确的实体变得困难，因为候选项的数量通 常很大。随着KG规模的不断增加，许多实体将共享相同的名称。此 外，最终用户可以在他们的话语中使用部分名称。例如，在问题“杰 克多大了?”中，只显示了实体名称大卫·杰克的一部分。第三，最 终用户问题的领域通常是无界的，任何KG都远远不够完整。新问题 可能涉及与训练中不同的谓词。这就对QA-KG算法的鲁棒性提出了 要求。

[0028] 为了弥补这一差距，本专利文献公开了如何利用知识图嵌入来执 行问题回答。在本公开中，焦点集中在QA-KG中最常见的问题类型， 即简单问题。简单问题是只涉及一个头部实体和一个谓词的自然语言 问题。通过对问题的分析，回答了三个研究问题：(i)如何运用谓词 嵌入表示来弥补自然语言表达与KG谓词之间的差距?;(ii)如何利 用嵌入表示的实体来应对模糊性挑战?;以及(iii)如何利用KG嵌 入表示中保留的全局关系来推进QA-KG框架?在这些问题之后，本 文献公开了名为基于知识嵌入的问题回答(KEQA)的框架的实施方 式。总之，本文献的一些主要贡献如下：

[0029] •正式定义基于知识图嵌入的问题回答的问题。

[0030] •公开了通过在知识图嵌入空间中联合恢复自然语言问题的头部 实体、谓词和尾部实体表示来回答自然语言问题的有效框架KEQA的 实施方式。

[0031] •设计联合距离度量，考虑知识图嵌入表示中保留的结构和关系。

[0032] •在大的基准上，即SimpleQuestions(简单问题)上，以经验 方式证明KEQA实施方式的有效性和鲁棒性。

[0033] B.一些相关工作

[0034] 本节总结了各方面的相关工作。

[0035] 基于嵌入的KG问题回答最近引起了广泛关注。它与所提出的基 于KG嵌入的问题回答问题相关，但又不同。前者依赖于在QA-KG 方法的训练过程中学习到的低维表示。后者先执行KG嵌入以学习低 维表示，然后执行QA-KG任务。Yih等人在ACL-IJCNLP上发表的“通过分段查询图生成的语义解析：知识库的问题回答”(Semantic Parsing via Staged Query Graph Generation:Question Answering with Knowledge Base.In ACL-IJCNLP)和Bao等人在COLING.上发表的 “知识图的基于约束的问题回答”(Constraint-Based Question Answering with Knowledge Graph.In COLING.2503-2514)将问题回答 问题重新表述为特定子图的生成。一系列工作提议基于训练问题将问 题和候选答案(或全部事实)投射到统一的低维空间中，并通过它们 的低维表示之间的相似性来测量它们的匹配分数。有些方法通过学习 所有单词、谓词和实体的低维表示，基于训练问题和问题的释义，实现了这种投射。有些方法通过使用问题的逻辑属性和潜在事实(如语 义嵌入和实体类型)

实现了这种投射。一些基于深度学习的模型通过 将问题中的单词馈入卷积神经网络、LSTM网络或门控递归单元神经 网络实现了这种投射。Das等人在ACL上发表的“使用通用架构和存 储网络的文本和知识库问题回答”(Question Answering on Knowledge Bases and Text using Universal Schema and Memory Networks.In ACL, 2017),通过使用矩阵分解将语料库合并到KG中,并使用LSTM嵌入 问题,实现了这种投射。这些模型大多依赖于基于边际的排序目标函 数来学习模型权重。一些工作探索了利用字符级神经网络来提高性能。最近,Mohammed等人在NAACL-HLT.上发表的“具有和不具有神经 网络的知识图简单问题回答的强大基准”(Strong Baselines for Simple Question Answering over Knowledge Graphs with and without Neural Networks,NAACL-HLT.291-296)和Ture等人在EMNLP.上发表的“无 需关注:简单的递归神经网络”(No Need to Pay Attention: Simple Recurrent Neural Networks Work,EMNLP.2866-2872)将每个谓词视为 一个标签类别,并通过深度分类模型执行谓词链接。

[0036] 知识图嵌入旨在将高维KG表示为潜在谓词和实体表示P和E。Bordes等人在2011AAAI上发表的“学习知识库的结构化嵌入” (Learning Structured Embeddings of Knowledge Bases.2011AAAI)通 过为每种类型的谓词l构建两个变换矩阵M_{head}和M_{tail},并针对所有事实 (h,l,t)最小化投射$M_{head}e_h$与$M_{tail}e_t$之间的距离,实现了这一目标,其中用l 作为谓词。Bordes等人在2013NIPS.上发表的“转换嵌入以建模多关 系数据”(Translating Embeddings for Modeling Multi-relational Data. 2013NIPS.2787-2795)设计了基于翻译的模型TransE。其训练两个矩 阵P和E,旨在针对所有事实(h,l,t)最小化总距离$\sum\|\mathbf{e}_h+\mathbf{p}_\ell-\mathbf{e}_t\|_2^2$。在 TransE的推动下,探索了一系列基于翻译的模型。Wang等人在2014 AAAI上发表的“通过在超平面上平移嵌入知识图”(Knowledge Graph Embedding by Translating on Hyperplanes.2014AAAI)提出了TransH 以处理一对多或多对一关系。TransH没有直接测量e_h与e_t之间的距离,而是将它们投射到谓词特定的超平面中。Lin等人在2015AAAI上发 表的“学习实体和关系嵌入以完成知识图”(Learning Entity and Relation Embeddings for Knowledge Graph Completion.2015AAAI 2181-2187)提出了TransR,它为每个谓词l定义了变换矩阵M_l,目标 是将$\sum\|\mathbf{e}_h\mathbf{M}_\ell+\mathbf{p}_\ell-\mathbf{e}_t\mathbf{M}_\ell\|_2^2$最小化。Lin等人在2015EMNLP上发表的“知 识库表示学习的关系路径建模”(Modeling Relation Paths for Representation Learning of Knowledge Bases,2015EMNLP.705-814)提 出了PTransE,其通过考虑多跳关系来推进TransE。

[0037] 还致力于将语料库中的语义信息纳入KG嵌入。一些人证明使用 预先训练好的单词嵌入来初始化KG嵌入方法可以提高性能。通过考 虑语料库中的关系提及,或者通过将谓词/实体表示投射到从主题模型 中学习到的语义超平面中,探索了试图推进TransE的几项研究。还尝 试了分别应用TransE和word2vec分别为KG和语料库建模,然后基 于Wikipedia中的锚点、实体描述或从语料库中学习的谓词/实体的上 下文单词来融合它们。Zhang等人在关于连续向量空间模型及其组合 的研讨会上发表的“结合文本数据和图形知识的联合语义相关性学 习”。(Joint Semantic Relevance Learning with Text Data and Graph Knowledge.In Workshop on Continuous Vector Space Models and their Compositionality.32-40)通过负采样共同嵌入KG和语料库(单词和 短语的分布式表示及其组成(Distributed Representations of Words and Phrases and Their

Compositionality,2013NIPS.3111-3119))。Xie等人 在2016AAAI上发表的“具有实体描述的知识图的表示学习” (Representation Learning of Knowledge Graphs with Entity Descriptions.2016AAAI 2659-2665)和Fan等人在模式识别快报上发 表的“具有实体描述的知识图的分布式表示学习”(Distributed Representation Learning for Knowledge Graphs with Entity Descriptions, Pattern Recognition Letters 93(2017),31-37)探索了实体描述中的语义 信息,以推进KG嵌入。

[0038]　C.问题陈述

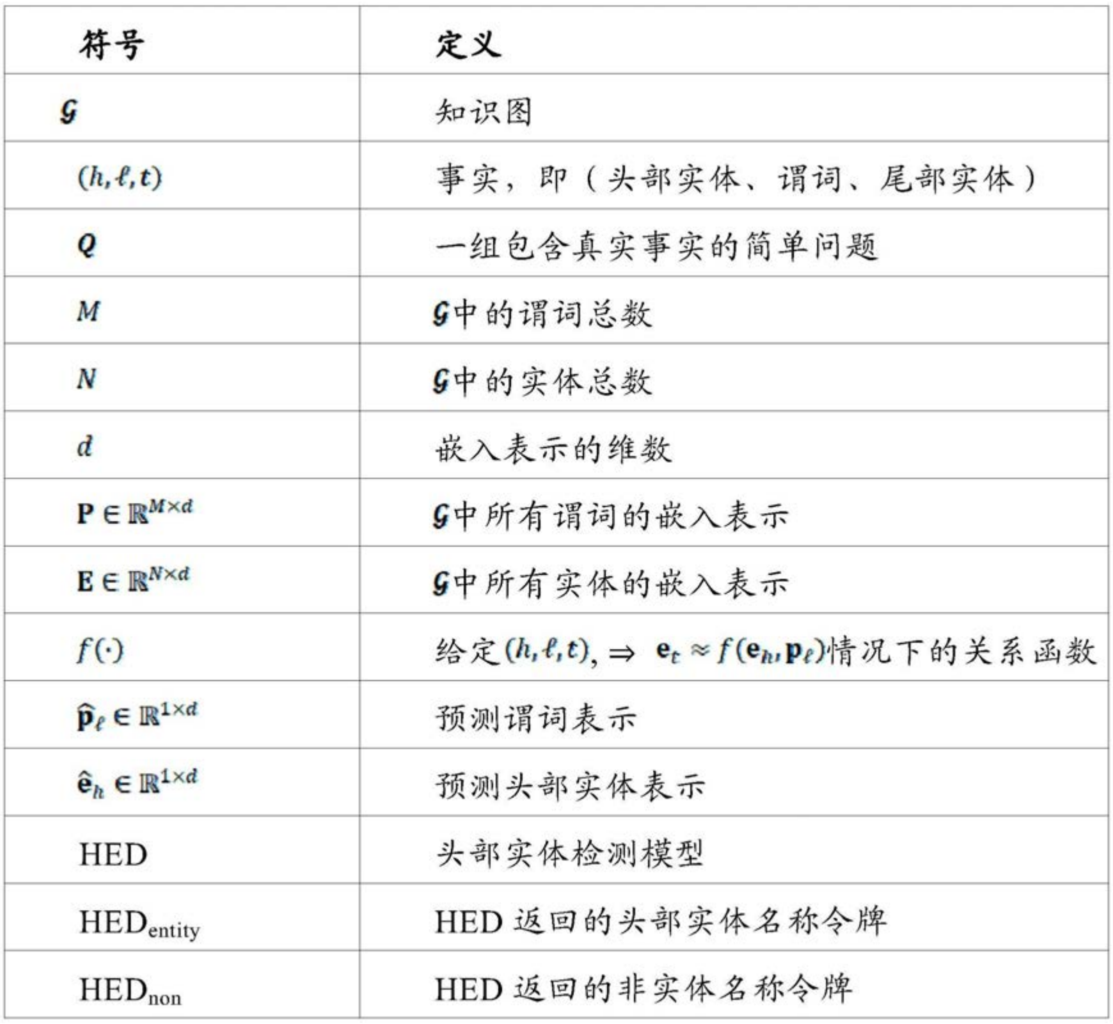
[0039]　符号:在本专利文献中,大写粗体字母用于表示矩阵(例如w), 小写粗体字母用于表示向量(例如,p)。矩阵P的第i行表示为p_i。 向量的转置表示为p^T。向量的l^2范数表示为$||p||_2$。$\{p_i\}$用于表示向量p_i的 序列。s=[x;h]运算表示将列向量x和h串联成新向量s。

[0040]　定义1(简单问题)如果自然语言问题在知识图中只涉及一个头 部实体和一个谓词,并以它们的(多个)尾部实体作为答案,那么这 个问题就称为简单问题。

[0041]　本专利文献中的一些符号总结在表1中。(h,l,t)用于表示事实,这 意味着存在从头部实体h到尾部实体t的关系l。假设$\mathcal{G}$是由大量事实组 成的知识图。谓词和实体的总数表示为M和N。给出了这些谓词和实 体的名称。在一个或多个实施方式中,可缩放的KG嵌入算法,例如 TransE和TransR,被应用于$\mathcal{G}$,并且获得其谓词和实体的嵌入表示, 分别被表示为P和E。因此,第i个谓词和第j个实体的向量表示分别 表示为p_i和e_j。由KG嵌入算法定义的关系函数是f(•),即,给定事实 (h,l,t),可以有$e_t \approx f(e_h, p_l)$。假设Q是一组简单问题。对于Q中的每个 问题,都给出了相应的头部实体和谓词。

[0042]　表1:一些符号及其定义

[0043]



符号	定义
$\mathcal{G}$	知识图
(h,ℓ,t)	事实，即（头部实体、谓词、尾部实体）
Q	一组包含真实事实的简单问题
M	$\mathcal{G}$中的谓词总数
N	$\mathcal{G}$中的实体总数
d	嵌入表示的维数
$\mathbf{P} \in \mathbb{R}^{M\times d}$	$\mathcal{G}$中所有谓词的嵌入表示
$\mathbf{E} \in \mathbb{R}^{N\times d}$	$\mathcal{G}$中所有实体的嵌入表示
$f(\cdot)$	给定(h,ℓ,t), $\Rightarrow$ $\mathbf{e}_t \approx f(\mathbf{e}_h,\mathbf{p}_\ell)$情况下的关系函数
$\hat{\mathbf{p}}_\ell \in \mathbb{R}^{1\times d}$	预测谓词表示
$\hat{\mathbf{e}}_h \in \mathbb{R}^{1\times d}$	预测头部实体表示
HED	头部实体检测模型
$\text{HED}_{\text{entity}}$	HED 返回的头部实体名称令牌
HED_{non}	HED 返回的非实体名称令牌

[0044] 术语简单问题在定义1中定义。如果简单问题的单个头部实体和 单个谓词被识别，则可以由机器直接回答简单问题。给定上述条件，基于知识图嵌入的问题回答问题现在正式定义如下：

[0045] 给定与其所有谓词和实体的名称相关联的知识图$\mathcal{G}$和嵌入表示P &E、关系函数f（•）以及与相应的头部实体和谓词相关联的一组简单问 题Q，公开了端到端框架的实施方式，以新的简单问题作为输入并自 动返回相应的头部实体和谓词。框架的性能通过正确预测头部实体和 谓词的来评估。

[0046] D.<u>基于知识嵌入的QA-KG实施方式</u>

[0047] 简单问题构成QA-KG问题中的大多数问题。如果识别出正确的 头部实体和谓词，(多个)尾部实体可以回答每一个简单问题。为了精 确预测头部实体和谓词，本专利文献公开了基于知识嵌入的问题回答 (KEQA)框架的实施方式，如图1所示。KG$\mathcal{G}$160已经嵌入到两个 低维空间(谓词嵌入空间140和实体嵌入空间150)中，并且每个事 实(h,l,t)可以表示为三个潜在向量，即(e_h,p_l,e_t)。因此，给定问题110，只要其对应的事实e_h和p_l可以被预测，这个问题就可以被正确地回答 170。KEQA实施方式的目标不是直接推断头部实体和谓词，而是在知 识图嵌入空间中联合恢复问题的头部实体、谓词和尾部实体表示 $(\hat{\mathbf{e}}_h, \hat{\mathbf{p}}_\ell, \hat{\mathbf{e}}_t)$。

[0048] 图2描绘了根据本公开的实施方式的利用KEQA框架回答问题的 方法。在一个或多个实施方式中，KEQA通过以下步骤获得答案。(i) 基于Q中的问题及其谓词的嵌入表示，

KEQA训练(205)谓词学习模 型120,其将问题110作为输入,并返回位于KG谓词嵌入空间140中的向量 $\hat{\mathbf{p}}_\ell$ 作为预测谓词表示。类似地,头部实体学习模型130被构 造成预测(210)KG实体嵌入空间150中问题的头部实体表示$\hat{\mathbf{e}}_h$。(ii) 由于KG中的实体数量通常很大,KEQA采用头部实体检测模型来减 少(215)候选头部实体。主要目标是将问题中的一个或多个令牌识别 为预测的头部实体名称,然后将$\mathcal{G}$中的搜索空间从整个实体减少到具有 相同或相似名称的多个实体。然后,$\hat{\mathbf{e}}_h$主要用于解决歧义挑战。(iii) 给定由KG嵌入算法定义的关系函数f(•),KEQA实施方式计算(220) 预测尾部实体表示$\hat{\mathbf{e}}_t = f(\hat{\mathbf{e}}_h, \hat{\mathbf{p}}_\ell)$。预测谓词表示 $\hat{\mathbf{p}}_\ell$、预测头部实体表示$\hat{\mathbf{e}}_h$ 和预测尾部实体表示$\hat{\mathbf{e}}_t$形成预测事实$(\hat{\mathbf{e}}_h, \hat{\mathbf{p}}_\ell, \hat{\mathbf{e}}_t)$。基于精心设计的联合距离度量,选择(225) $\mathcal{G}$中的预测事实$(\hat{\mathbf{e}}_h, \hat{\mathbf{p}}_\ell, \hat{\mathbf{e}}_t)$中的最近的事实,并将 其作为问题的答案170返回。

[0049]　1.知识图嵌入的实施方式

[0050]　在一个或多个实施方式中,所公开的框架KEQA采用所有谓词P 和实体E的嵌入表示作为基础结构。在一个或多个实施方式中,可以 利用现有的KG嵌入算法来学习P和E。可以使用的现有KG嵌入方 法的示例包括但不限于TransE、TransR、TransH等。

[0051]　知识图嵌入旨在将KG中的每个谓词/实体表示为低维向量,从而 KG中的原始结构和关系保留在这些学习向量中。大多数现有KG嵌 入方法的核心思想可以总结如下。对于$\mathcal{G}$中的每个事实(h,l,t),其嵌入 表示被表示为(e_h,p_l,e_t)。嵌入算法随机地或基于训练的单词嵌入模型 初始化e_h,p_l和e_t的值。然后,定义了测量嵌入空间中事实(h,,t)的关系 的函数f(•),即$e_t≈f(e_h,p_l)$。例如,TransE将该关系定义为$e_t≈e_h+p_l$, TransR将其定义为$e_tM_l≈e_hM_l+p_l$,其中M_l是谓词l的转换矩阵。最后, 对于$\mathcal{G}$中的所有事实,嵌入算法将e_t和f(e_h,p_l)之间的总距离最小化。一 种典型的方法是定义基于边际的排序标准,并对正样本和负样本(即 不存在于$\mathcal{G}$中的事实和合成事实)进行训练。

[0052]　如图1所示,将学习预测表示{p_i}(其中i=1,...,M)所在的表 面定义为谓词嵌入空间。将{e_i}(其中i=1,...,N)所在的表面表示 为实体嵌入空间。

[0053]　2.谓词和头部实体学习模型的实施方式

[0054]　给定简单问题,目标是在谓词嵌入空间中找到一个点作为它的谓 词表示$\hat{\mathbf{p}}_\ell$,在实体嵌入空间中找到一个点作为它的头部实体表示$\hat{\mathbf{e}}_h$。

[0055]　在一个或多个实施方式中,对于所有可以由$\mathcal{G}$回答的问题,它们的 谓词的向量表示应该位于谓词嵌入空间中。因此,目标是设计一个模 型,该模型将问题作为输入,并返回尽可能接近该问题的谓词嵌入表 示p_l的向量$\hat{\mathbf{p}}_\ell$。为了实现这个目标,采用了如图3所示的神经网络架 构实施方式。在一个或多个实施方式中,该架构主要包括双向递归神 经网络层310和注意层325。在一个或多个实施方式中,双向递归神 经网络层310是双向长短期存储器(LSTM)。核心思想是考虑单词的 顺序和重要性。不同顺序的单词可能有不同的含义,单词的重要性也 可能不同。例如,问题中与实体名称相关的单词对谓词学习模型的贡 献通常较小。

[0056]　基于神经网络的谓词表示学习。为了预测问题的谓词,传统的解 决方案是基于语

义解析和人工创建的词典来学习映射，或者简单地将 每种类型的谓词视为标签类别来将其转换成分类问题。然而，由于最 终用户问题的领域通常是无界的，新问题的谓词可能不同于训练数据 中的所有谓词。传统的解决方案无法处理这种情况。此外，我们还观 察到，在P和E中保留的全局关系信息是可用的，并且有可能用于提 高整体问题回答的精确度。为了弥补这一差距，本文阐述了基于神经 网络的谓词学习模型的实施方式。

[0057] 利用长短期记忆(LSTM)作为递归神经网络的典型例子，图3 示出了根据本公开的一个或多个实施方式的谓词和头部实体学习模型 的架构。图4描绘了根据本公开的实施方式的使用谓词和头部实体学 习模型来预测输入问题的谓词的方法。给定长度为L的问题，基于预 先训练的模型，如GloVe(Pennington，等人，GloVe：单词表示的全局向 量(Global Vectors for Word Representation)，In EMNLP.1532-1543)， 首先将其L个令牌映射(405)到单词嵌入向量序列{x_j}305中，其中 j＝1，...，L，但是也可以使用其他嵌入技术。然后，使用(410)双向 LSTM 310来学习前向隐藏状态序列$(\overrightarrow{\mathbf{h}_1},\overrightarrow{\mathbf{h}_2},\ldots,\overrightarrow{\mathbf{h}_L})$和后向隐藏状态序列 $(\overleftarrow{\mathbf{h}_1},\overleftarrow{\mathbf{h}_2},\ldots,\overleftarrow{\mathbf{h}_L})$。以后向隐藏状态序列为例，通过以下方程进行计算。

[0058] $$\mathbf{f}_j=\sigma(\mathbf{W}_{xf}\mathbf{x}_j+\mathbf{W}_{hf}\overleftarrow{\mathbf{h}}_{j+1}+\mathbf{b}_f)\qquad(1)$$

[0059] $$\mathbf{i}_j=\sigma(\mathbf{W}_{xi}\mathbf{x}_j+\mathbf{W}_{hi}\overleftarrow{\mathbf{h}}_{j+1}+\mathbf{b}_i)\qquad(2)$$

[0060] $$\mathbf{o}_j=\sigma(\mathbf{W}_{xo}\mathbf{x}_j+\mathbf{W}_{ho}\overleftarrow{\mathbf{h}}_{j+1}+\mathbf{b}_o)\qquad(3)$$

[0061] $$\mathbf{c}_j=\mathbf{f}_j\circ\mathbf{c}_{j+1}+\mathbf{i}_j\tanh(\mathbf{W}_{xc}\mathbf{x}_j+\mathbf{W}_{hc}\overleftarrow{\mathbf{h}}_{j+1}+\mathbf{b}_c)\qquad(4)$$

[0062] $$\overleftarrow{\mathbf{h}}_j=\mathbf{o}_j\circ\tanh(\mathbf{c}_j)\qquad(5)$$

[0063] 其中，f_j、i_j和o_j分别是遗忘门、输入门和输出门的激活向量。c_j是 单元状态向量。σ和tanh是sigmoid和双曲线正切函数。o表示Hadamard 积。串联(415)前向和后向隐藏状态向量，可以获得串联的隐藏状态 向量$\mathbf{h}_j=[\overrightarrow{\mathbf{h}}_j;\overleftarrow{\mathbf{h}}_j]$315。

[0064] 在一个或多个实施方式中，第j个令牌的注意权重320，即α_j，基 于以下公式计算：

[0065] $$\alpha_j=\frac{\exp(q_j)}{\sum_{i=1}^{L}\exp(q_i)}\qquad(6)$$

[0066] $q_j=\tanh(W^T[x_j;h_j]+b_q)$ (7)

[0067] 其中b_q是偏置项。注意权重α_j可被应用(420)于h_j以获得加权隐 藏状态向量，然后将该向量与单词嵌入x_j串联(425)，产生隐藏状态 $s_j=[x_j;\alpha_j h_j]$325。然后，将完全连接的层应用(430)到于s_j，并且其 结果$\mathbf{r}_j\in\mathbb{R}^{d\times 1}$被表示为第j个令牌的目标向量330。预测谓词表示$\widehat{\mathbf{p}}_\ell$ 335可以被计算(435)为所有令牌的目标向量的平均值，即：

[0068] $$\widehat{\mathbf{p}}_\ell=\frac{1}{L}\sum_{j=1}^{L}\mathbf{r}_j^{\mathrm{T}}\qquad(8)$$

[0069] 在一个或多个实施方式中，基于训练数据，即简单问题Q及其谓词 的嵌入表示，计算所有权重矩阵、权重向量w和偏置项。

[0070] 基于神经网络的头部实体学习模型。在一个或多个实施方式中， 给定问题，目标

在KG嵌入空间中恢复其表示，而不是直接推断头部 实体。因此，头部实体学习模型的目标是计算尽可能接近该问题的头 部实体嵌入表示的向量$\hat{\mathbf{e}}_h$。类似于$\hat{\mathbf{p}}_\ell$的计算，可以使用图3中相同的 神经网络架构来获得预测的头部实体表示$\hat{\mathbf{e}}_h$.。

[0071] 然而，KG中的实体数量通常很大，并且当将$\hat{\mathbf{e}}_h$与E中的所有实 体嵌入表示比较时，这可能是昂贵的并且有噪声。为了使学习更加高 效和有效，KEQA实施方式可以采用头部实体检测模型来减少候选头 部实体的数量。

[0072] 3.头部实体检测模型的实施方式

[0073] 在这个步骤中，目标是在问题中选择一个或几个连续的令牌作为 头部实体的名称，使得搜索空间可以从整个实体减少到具有相同或相 似名称的多个实体。那么$\hat{\mathbf{e}}_h$的主要角色将变成处理模糊挑战。

[0074] 在一个或多个实施方式中，为了使框架简单，采用基于双向递归 神经网络(例如，LSTM)的模型来执行头部实体令牌检测任务。图5 示出了根据本公开的一个或多个实施方式的头部实体检测(HED)模 型的架构。如图5所示，HED模型包括双向LSTM 510和完全连接层 520。HED模型与谓词/头部实体学习模型的结构相似，但没有注意层。

[0075] 图6描绘了根据本公开的一个或多个实施方式的使用HED模型识 别输入问题的一个或多个头部实体的方法。在一个或多个实施方式中， 首先将问题映射(605)到单词嵌入向量序列$\{x_j\}$中(其中$j=1,...,L$)， 然后应用(610)双向递归神经网络来学习前向隐藏状态序列$\vec{\mathbf{h}}_j$和后向 隐藏状态序列$\overleftarrow{\mathbf{h}}_j$。将前向和后向隐藏状态串联(615)成串联隐藏状态$\mathbf{h}_j=[\vec{\mathbf{h}}_j;\overleftarrow{\mathbf{h}}_j]$。然后，将完全连接的层和softmax函数应用(620)于h_j， 得到目标向量$\mathbf{v}_j\in\mathbb{R}^{2\times1}$。$v_j$中的两个值对应于第j个令牌属于两个标签 类别(即实体名称令牌和非实体名称令牌)的概率。这样，每个令牌 被分类，并且一个或多个令牌被识别为头部实体名称。这些令牌被表 示为HED_{entity}，问题中剩余的令牌被表示为HED_{non}。基于属于实体名 称令牌的每个令牌的概率值，选择(625)一个或多个令牌作为头部实 体名称。

[0076] 在一个或多个实施方式中，Q中的问题及其头部实体名称被用作 训练数据来训练HED模型。由于这些问题中的实体名称令牌是连续 的，所以经过训练的模型也有很高的概率将连续的令牌作为HED_{entity}返回。如果返回离散的HED_{entity}，那么每个连续的部分将被视为独立 的头部实体名称。应该注意的是，HED_{entity}可能只是正确头部实体名 称的一部分。因此，所有与HED_{entity}相同或包含HED_{entity}的实体都将 作为候选头部实体被包括在内，这可能仍然很大，因为许多实体将在 大KG中共享相同的名称。

[0077] 4.嵌入空间上联合搜索的实施方式

[0078] 对于每一个新的简单问题，它的谓词和头部实体表示为$\hat{\mathbf{p}}_\ell$和$\hat{\mathbf{e}}_h$，并且它的候选头部实体正在被预测，目标是找到$\mathcal{G}$中最匹配这些学习到 的表示和候选的事实。

[0079] 联合距离度量。如果事实的头部实体属于候选头部实体，则其被 命名为候选事实。假设C成为收集所有候选事实的集合。为了测量候 选事实(h,l,t)和预测表示$(\hat{\mathbf{e}}_h,\hat{\mathbf{p}}_\ell)$之间的距离，直观的解决方案是将 (h,l,t)表示为(e_h,p_ℓ)，并将距离度量定义为e_h和$\hat{\mathbf{e}}_h$之

间的距离与p_l和$\hat{\mathbf{p}}_\ell$之间的距离之和。然而,这个解决方案没有考虑KG嵌入表示中保留 的有意义的关系信息。

[0080] 在一个或多个实施方式中,使用了利用关系信息$e_t \approx f(e_h, p_l)$的联 合距离度量。数学上,建议的联合距离度量可以定义为:

[0081]
$$\underset{(h,\ell,t)\in C}{\text{minimize}} \|\mathbf{p}_\ell - \hat{\mathbf{p}}_\ell\|_2 + \beta_1\|\mathbf{e}_h - \hat{\mathbf{e}}_h\|_2 + \beta_2\|f(\mathbf{e}_h, \mathbf{p}_\ell) - \hat{\mathbf{e}}_t\|_2 - \beta_3 sim[n(h), \mathrm{HED}_{\mathbf{entity}}] - \beta_4 sim[n(\ell), \mathrm{HED}_{\mathbf{non}}] \quad (9)$$

[0082] 其中$\hat{\mathbf{e}}_t = f(\hat{\mathbf{e}}_h, \hat{\mathbf{p}}_\ell)$。函数n(•)返回实体或谓词的名称。$\mathrm{HED}_{entity}$和 HED_{non}表示被HED模型分类为实体名称和非实体名称的令牌。函数 sim[•,•]测量两个字符串的相似性。β_1、β_2、β_3和β_4是预定义的权重,用 于平衡每个项的贡献。在一个或多个实施方式中,l^2范数用于测量距 离,并且可以直接扩展到其他向量距离测量。

[0083] 前三项(在等式(9)中可以称为向量距离项)在KG嵌入空间中 测量事实(h,l,t)和预测之间的距离。在一个或多个实施方式中,$f(e_h, p_l)$ 用于表示尾部实体的嵌入向量,而不是e_t。换句话说,使用由KG定 义的定义函数f(•),从候选事实的头部实体嵌入向量和谓词嵌入向量计 算联合距离度量中使用的候选事实的尾部实体嵌入向量。这是因为在 KG中,可能有几个事实具有相同的头部实体和谓词,但是尾部实体 不同。因此,单个尾部实体e_t可能无法回答这个问题。同时,$f(e_h, p_l)$匹 配预测的尾部实体$\hat{\mathbf{e}}_t$,因为它也是基于f(•)推断的。其倾向于选择这样 一个事实,其头部实体名称与HED_{entity}完全相同,并且谓词名称被问 题提及。在一个或多个实施方式中,这两个目标分别通过第四项和第 五项(在等式(9)中称为字符串相似性项)来实现。在一个或多个实 施方式中,字符串相似性项被合并到联合距离度量中,以帮助选择头 部实体名称与HED_{entity}完全相同并且谓词名称被问题提及的事实。返 回最小化目标函数的事实(h^*, l^*, t^*)。

[0084] 基于知识嵌入的问题回答。方法1总结了KEQA实施方式的整个 过程。给定KG$\mathcal{G}$和具有相应答案的问题集Q,谓词学习模型、头部实 体学习模型和HED模型被训练,如从第1行到第9行所示。然后,对 于任何新的简单问题Q,其被输入到训练的谓词学习模型、头部实体学习模型和HED模型中,以学习其预测的谓词表示$\hat{\mathbf{p}}_\ell$、头部实体表示$\hat{\mathbf{e}}_h$、实体名称令牌HED_{entity}和非实体名称令牌HED_{non}。基于HED_{entity}中的 学习到的(多个)实体名称,搜索整个$\mathcal{G}$以找到候选事实集合C。对于 C中的所有事实,它们到预测表示$(\hat{\mathbf{e}}_h, \hat{\mathbf{p}}_\ell, \hat{\mathbf{e}}_t)$的联合距离基于等式(9)中的目标函数来计算。选择具有最小距离的事实(h^*, l^*, t^*)。最后,头部 实体h^*和谓词l^*作为Q的答案返回。

[0085] 方法1:KEQA框架实施方式

[0086]
```
输入: $\mathcal{G}$, 谓词和实体的名称, P, E, $\mathcal{Q}$, 新的简单问题 $Q$。
输出: 标题实体$h^*$和谓词$\ell^*$。
/*训练谓词学习模型:                                   */
1  对于 $\mathcal{Q}$ 中的 $Q_i$, 进行
```

```
2       将 $Q_i$ 的 $L$ 个令牌作为输入，将其谓词 $\ell$ 作为要训练的标签，如
        图 3 所示；
3       更新权重矩阵 $\{\mathbf{W}\}$、$\mathbf{w}$、$\{\mathbf{b}\}$ 和 $b_q$ 以最小化谓词目标函数
        $\left\|\mathbf{p}_\ell - \frac{1}{L}\sum_{j=1}^{L}\mathbf{r}_j^\top\right\|_2$
    /* 训练头部实体学习模型:                    */
4   对于 $Q$ 中的 $Q_i$，进行
5       将 $Q_i$ 的 $L$ 令牌作为输入，将它的头部实体 $h$ 作为标签进行训练，如图 3
        所示；
6       更新权重矩阵和偏置项以最小化头部实体目标函数
        $\left\|\mathbf{e}_h - \frac{1}{L}\sum_{j=1}^{L}\mathbf{r}_j^\top\right\|_2$
    /* 训练 HED 模型:
    */
```

[0087]

```
7   对于 $Q$ 中的 $Q_i$，进行
8       以 $Q_i$ 的 $L$ 令牌作为输入，以其头部实体名称位置作为标签进行
        训练；
9       更新权重矩阵和偏置项，如图 5 所示
    /* 问题回答过程:
    */
10  将 $Q$ 输入谓词学习模型中以学习 $\hat{\mathbf{p}}_\ell$；
11  将 $Q$ 输入头部实体学习模型中以学习 $\hat{\mathbf{e}}_h$；
12  将 $Q$ 输入 HED 模型中以学习 $\mathrm{HED}_{\mathrm{entity}}$ 和 $\mathrm{HED}_{\mathrm{non}}$；
13  基于 $\mathrm{HED}_{\mathrm{entity}}$ 从 $\mathcal{G}$ 中找到候选事实集 $C$；
14  对于 $C$ 中的所有事实，计算使等式（9）中的目标函数最小化的事实
    $(h^*, \ell^*, t^*)$。
```

[0088]　作为与步骤12(上文)相关的示例,从HED模型来看,图5中 的结果将是“澳大利亚”将具有作为实体名称令牌的高概率。作为另 一个例子,在一个或多个实施方式中,包含“亚伯拉罕·林肯总统”的 短语将返回结果,其中单词“亚伯拉罕”和“林肯”中的每一个都具有被组合的高概率,至少因为连续的令牌和/或多个令牌作为一个实体 是名称相关的。

[0089]　图7描绘了根据本公开的一个或多个实施方式的步骤13(以上) 的实施方式实现方案。图7示出了根据本公开实施方式的使用由HED 模型识别的头部实体名称搜索KG中的头部实体同义词的方法。$\mathrm{HED}_{\mathrm{entity}}$可以是单个实体,也可以包含多个实体。在一个或多个实施 方式中,由HED模型识别为头部实体的一个或多个实体被输入(710) 到KG中,KG包括实体、谓词、它们的唯一代码、同义词集及其嵌 入。实体可以包括一个或多个令牌,例如“亚伯

拉罕·林肯总统”。因 此，在一个或多个实施方式中，对于包括多个令牌的候选实体，可以例如通过实体的每个令牌的实体向量的点积来形成实体向量。在一个 或多个实施方式中，搜索策略包括利用每个识别的头部实体的嵌入比 较、字符串匹配或两者来搜索KG。

[0090] 在一个或多个实施方式中，在确定(715)每个识别的头部实体是 否存在直接字符串匹配时，该过程或者返回(720)结果，该结果可以 包括匹配字符串的实体代码和一组或多组同义词。在一个或多个实施 方式中，如果没有找到直接字符串匹配，则搜索可以被扩展以尝试识 别(725)是否存在一个或多个部分字符串匹配。例如，“亚伯拉罕·林 肯总统”和“美国内战时期的总统”这两个字符串是部分匹配的，也 被认为是同一个实体。如果识别出一个或多个部分字符串匹配，则搜 索过程返回(730)结果，对于每个部分匹配，该结果可以包括一组或 多组同义词的实体代码。在一个或多个实施方式中，响应于没有找到 直接或部分字符串匹配，用嵌入相似性识别(735)每个识别的头部实 体的头部实体同义词。经由直接字符串匹配、部分字符串匹配和嵌入 相似性的所识别的头部实体的所有同义词被收集在一起，以建立(740) 一个或多个所识别的头部实体的候选事实集。

[0091] 在一个或多个实施方式中，对于每个搜索策略(字符串匹配和嵌 入比较)，可以使用一个或多个阈值来决定是否存在足够的相似性或匹 配。字符串匹配中的阈值可能与嵌入比较的阈值相同，也可能不同。

[0092] 作为与步骤13和14(上文)相关的进一步说明，一旦找到一组 候选头部实体(例如，从如图7所示的搜索过程)，可以基于一组找到 的头部实体、来自训练中找到的谓词Q以及从训练数据中已知的尾部 实体来构建候选事实集。给定从训练数据中构建的具有已知尾部实体(或真实事实)的候选事实集，可以将候选事实集输入等式(9)中， 以联合训练KEQA框架中的模型。一旦训练完成，可以使用KEQA框 架预测测试数据中新问题Q的尾部实体。

[0093] 应当注意，这些训练实施方式和结果是以举例说明的方式提供的， 并且是在特定条件下使用一个或多个特定实施方式来执行的；因此， 这些训练实施方式及其结果都不应用于限制当前专利文献的公开范 围。

[0094] 通过整体概述的方式，所公开的框架KEQA实施方式享有几个好 的属性。首先，通过执行基于KG嵌入的问题回答，KEQA实施方式 能够使用不同于训练数据中所有谓词和实体的谓词和实体来处理问 题。其次，通过利用KG嵌入表示中保留的结构和关系信息，KEQA实施方式可以联合执行头部实体、谓词和尾部实体预测。这三个子任 务将相互补充。第三，KEQA框架可推广到不同的KG嵌入算法。因 此，KEQA实施方式的性能可以通过更复杂的KG嵌入算法进一步提 高。

[0095] E.一些实验

[0096] 应当注意，这些实验和结果是以举例说明的方式提供的，并且是 在特定条件下使用一个或多个特定实施方式进行的；因此，这些实验 及其结果都不应用于限制当前专利文献的公开范围。

[0097] 在本节中，评估了在大型QA-KG基准上所公开的框架KEQA的 测试实施方式的有效性和普遍性。在一个或多个实验中，研究了以下 三个研究问题：

[0098] Q1：与最先进的QA-KG方法相比，KEQA实施方式在不同的自 由基子集方面有多有效？

[0099] Q2:当采用不同的KG嵌入算法时,KEQA实施方式的性能有什 么不同?

[0100] Q3:KEQA实施方式的目标函数包括五个项,如等式(9)所示。

[0101] 每一项贡献多少?

[0102] 1.数据集的实施方式

[0103] 本节首先介绍了实验中使用的知识图子集和问题回答数据集。所 有的数据都是公开的。它们的统计数据见表2。

[0104] 表2:问题回答数据集的统计数据

[0105]

	FB2M	FB5M	简单问题
#训练	14,174,246	17,872,174	75,910
#验证	N.A.	N.A.	10,845
#测试	N.A.	N.A.	21,687
#谓词（M）	6,701	7,523	1,837
#实体（N）	1,963,130	3,988,105	131,681
词汇大小	733,278	1,213,205	61,336

[0106] FB2M和FB5M:自由基通常被认为是可靠的KG,因为它主要由 社区成员收集和整理。本文采用了自由基的两个大子集,即FB2M和 FB5M。它们的谓词编号M和实体编号N列于表2中。重复的事实已 被删除。自由基应用编程接口(API)不再可用。因此,实体名称集 合可以用于建立实体与其名称之间的映射。

[0107] 简单问题(Borders,等人,利用存储网络扩展简单的w问题回答 (Scale Simple Question Answering with Memory Networks).2015 arXiv preprint:1506.02075):其包含一万多个与相应事实相关的简单问 题。所有这些事实都属于FB2M。所有的问题都是由讲英语的人根据 事实和上下文来表述的。它被用作最近各种QA-KG方法的基准。

[0108] 2.实验设置

[0109] 在一个或多个实施方式中,为了评估QA-KG方法的性能,使用 传统的设置并使用最初在简单问题中提供的相同的训练、验证和测试 部分。FB2M或FB5M用作KG。然后应用KG嵌入算法(例如TransE 和TransR)来学习P和E。应该注意P和E不是额外的信息源。然后，使用QA-KG方法预测测试部分中每个问题的头部实体和谓词。其性 能是通过正确预测头部实体和谓词的精确度来衡量的。

[0110] 正如形式问题定义中所要求的,评估标准被定义为正确预测新问 题的头部实体和谓词的精确度。KG嵌入表示d的维度设置为250。使 用了基于GloVe的预先训练的单词嵌入。在一个或多个实施方式中， 为了测量两个字符串的相似性,即,为了建立函数sim[•,•],使用实现 方案Fuzzy(模糊)。如果其不是特定的,KG嵌入算法TransE将被用 来学习所有谓词P和实体E的嵌入表示。

[0111] 3.测试的KEQA实施方式的有效性

[0112] 本节开始时提出的第一个研究问题,即KEQA有多有效,现在得 到了回答。在一个或多个实施方式中,包括7种最先进的QA-KG算 法和KEQA的一种变化作为基线。

[0113] •Bordes等人在arXiv preprint发表的“带有存储网络的大规模 简单问题回答”(Large Scale Simple Question Answering with Memory Networks.arXiv preprint 1506.02075):其基于训练问题学习单词、谓 词和实体的潜在表示,以便新问题和候选事实可以被投射到相同的空 间并进行比较。

[0114] •Dai等人在arXiv preprint arXiv发表的“CFO:利用大型知识 库条件聚焦的神经问题回答”(CFO:Conditional Focused Neural Question Answering with Large-Scale Knowledge Bases.arXiv preprint arXiv:1606.01994):其采用基于双向门控递归单元的神经网络对候选 谓词进行排序。使用了来自自由基API的建议。

[0115] •Yin等人在2016COLING.发表的“细观卷积神经网络的简单 问题回答”(Simple Question Answering by Attentive Convolutional Neural Network,2016COLING.1746-1756.):其使用字符级卷积神经 网络来匹配问题和谓词。

[0116] •Golub和He在EMNLP发表的“注意字符级问题回答” (Character-Level Question Answering with Attention.In EMNLP. 1598-1607):其设计了字符级和基于注意的LSTM来编码和解码问题。

[0117] •Bao等人在COLING.发表的“利用知识图的基于约束的问题回 答”(Constraint-Based Question Answering with Knowledge Graph.In COLING.2503-2514):其手动定义几种类型的约束条件,并执行约束 条件学习来处理复杂的问题,其中每个问题都与几个事实相关。使用 额外的训练问题和免费的自由基API。

[0118] •Lukovnikov等人在WWW.发表的“基于神经网络的单词和字符 级别的知识图问题回答”(Neural Network-Based Question Answering over Knowledge Graphs on Word and Character Level.In WWW. 1211-1220):其利用字符级门控循环单元神经网络将问题和谓词/实体 投射到同一空间。

[0119] •Mohammed等人在NAACL-HLT.发表的“具有和不具有神经网 络的知识图简单问题回答的强基线”(Strong Baselines for Simple Question Answering over Knowledge Graphs with and without Neural Networks,In NAACL-HLT.291-296):其将谓词预测视为分类问题,并 使用不同的神经网络来解决它。其基于Fuzzy执行实体链接。

[0120] •KEQA_noEmbed:不使用KG嵌入算法。相反,其随机生成谓 词和实体嵌入表示P和E。

[0121] 如上文导言所示,所有基线都利用深度学习模型来推进其方法。它们在相应的论文或作者的实现方案中报告的结果被使用。表3中列 出了针对FB2M和FB5M的SimpleQuestions的不同方法的性能。

[0122] 表3:关于SimpleQuestions的所有方法的性能

[0123]

	FB2M（Accuracy）	FB5M
Bordes 等人	0.627	0.639
Dai 等人	N.A.	0.626
Yin 等人	0.683 (+8.9%)	0.672
Golub 和 He	0.709 (+13.1%)	0.703
Bao 等人	0.728 (+16.1%)	Entire Freebase
Lukovnikov 等人	0.712 (+13.6%)	N.A.
Mohammed 等人	0.732 (+16.7%)	N.A.
KEQA_noEmbed	0.731 (+16.6%)	0.726
KEQA	**0.754 (+20.3%)**	**0.749**

[0124] 正如Lukovnikov等人和Mohammed等人的其他几项工作所提到 的，一些算法实现了高精确度，但是它们要么使用额外的信息源，要 么没有可用的实现方案。额外的训练数据自由基API建议、自由基实 体链接结果和训练的分段模型。这些依赖于不再可用的自由基API。相反，所呈现的框架KEQA实施方式使用实体名称集合。因此，对于 Dai等人和Yin等人，当不使用额外的训练数据时，报告他们的结果。有两项工作声称具有很高精确度，但没有公开可用的实现方案。因此，不可能复制它们，其他工作也指出了这一点。

[0125] 根据表3中的结果，进行了三次观察。首先，提议的框架KEQA 优于所有基线。与发布SimpleQuestions时相比，KEQA的精确率提高 了20.3％。其次，与KEQA_noEmbed相比，KEQA实现了20.3％的提 高。这表明单独的任务KG嵌入确实有助于问题回答任务。第三，当应用于FB5M时，KEQA的性能下降0.7％。因为所有真实事实都属于 FB2M，FB5M的事实比FB2M多26.1％。

[0126] 通过联合预测问题的谓词和实体，KEQA达到了0.754的精确度。在谓词预测子任务中，KEQA在验证部分上达到0.815的精确度，这 比Mohammed等人最近达到的0.828的差。这一差距表明，本专利文 献中提出的KEQA框架可以通过更复杂的模型进一步改进。然而，在 简单问题回答任务中，KEQA仍然优于Mohammed等人的方案。这证 实了所提出的联合学习框架的有效性。通过联合学习，KEQA在预测 头部实体时达到0.816的精确度，在预测头部实体和谓词时达到0.754 的精确度，在预测整个事实、测试部分和FB2M时达到0.680的精确度。这意味着FB2M中不存在一些真实事实。

[0127] 4.<u>普遍性和鲁棒性评估的实施方式</u>

[0128] E.4.1KEQA的普遍性。在一个或多个实施方式中，为了研究当使 用不同的KG嵌入算法时KEQA的普遍性程度，在比较中包括三种可 缩放的KG嵌入方法。详细介绍列举如下：

[0129] •KEQA_TransE:TransE用于执行KG嵌入。这是一种典型的基 于翻译的方法。其将关系函数定义为$e_t \approx f(e_h, p_l) = e_h + p_l$，然后执行 基于边距的排序，使所有事实趋近满足关系函数。

[0130] •KEQA_TransH:TransH用于执行KG嵌入。TransH类似于 TransE，并将关系函数定

义为$\mathbf{e}_t^{\perp} \approx \mathbf{e}_h^{\perp} + \mathbf{p}_\ell$,其中$\mathbf{e}_t^{\perp} = \mathbf{e}_t - \mathbf{m}_\ell^{T}\mathbf{e}_t\mathbf{m}_\ell$和$m_l$是 谓词l的超平面。

[0131] •KEQA_TransR:TransR类似于TransE,并将关系函数定义为 $e_tM_l \approx e_hM_l + p_l$,其中$M_l$是l的变换矩阵。

[0132] 当不使用KG嵌入和使用不同的KG嵌入算法时,KEQA的性能 如表4所示。从结果中,获得三个主要的观察结果。首先,KG嵌入 算法提高了KEQA的性能。例如,与KEQA_noEmbed相比,KEQA 在基于TransE的基础上实现了3.1%的改进。其次,当使用不同的KG 嵌入算法时,KEQA具有相似的性能。其证明了KEQA的普遍性。第 三,即使不使用KG嵌入,KEQA仍然可以获得与表3所示的最先进 的QA-KG方法相当的性能。其验证了KEQA的鲁棒性。随机生成的 P和E能够实现相当的性能的原因是,其倾向于使所有p_l均匀分布并 且彼此远离。这将表示预测问题转换成类似于分类任务的问题。

[0133] 表4:不同知识图嵌入算法在FB2M上的KEQA性能

[0134]

	简单问题	SimpleQ_Missing
KEQA_noEmbed	0.731	0.386
KEQA_TransE	0.754 (+3.1%)	0.418 (+8.3%)
KEQA_TransH	0.749 (+2.5%)	0.411 (+6.5%)
KEQA_TransR	0.753 (+3.0%)	0.417 (+8.0%)

[0135] 4.2KEQA的鲁棒性。为了进一步验证KEQA的鲁棒性, SimpleQuestions中的所有108,442个问题都被重新改组,并获得名为 SimpleQ_Missing的新数据集。在一个或多个实施方式中,为了执行重 新改组,所有类型的谓词被随机分成三组,并基于谓词将问题分配给 这些组。因此,在SimpleQ_Missing中,测试部分中的问题的所有对 应谓词从未在训练和验证部分中提到过。最后,在训练部分中得到了 75,474个问题,在验证部分中得到了11,017个问题,在测试部分中 得到了21,951个问题,它们的比率与SimpleQuestions中的比率大致 相同。不同KG嵌入算法对SimpleQ_Missing的KEQA性能如表4所 示。

[0136] 从表4中的结果可以看出,借助于TransE,KEQA仍然可以实现 0.418的精确度。KG嵌入表示P和E中保存的全局关系和结构信息使 KEQA的性能比Random(随机)好8.3%。这些观察结果证明了KEQA 的鲁棒性。

[0137] 5.参数分析的实施方式

[0138] 在本节中,将调查KEQA目标函数中的每一项可以贡献多少。目 标函数中有五项,如等式(9)所示。在一个或多个实施方式中,研究 了KEQA相对于三组不同项的组合的性能。为了研究等式(9)中每 一项的贡献,在第一组,即Only_Keep(仅保留)中,五个项中只有 一个被保留作为新的目标函数。为了研究五个项中缺失一个的影响, 在第二组,即Remove(移除)中,五个项中的一个被移除。为了研究 累积贡献,在第三组,即Accumulate(累积),项区域作为新的目标函 数一个接一个地增加。表5总结了KEQA相对于FB2M上不同组目标 函数的性能。

[0139] 表5:在FB2M上具有不同目标函数的KEQA实施方式的性能

[0140]

	Only_Keep(仅保留)	Remove(移除)	Accumulate(累积)
$\|\mathbf{p}_\ell - \hat{\mathbf{p}}_\ell\|_2$	0.728	0.701	0.728
$\|\mathbf{p}_h - \hat{\mathbf{e}}_h\|_2$	0.195	0.751	0.745
$\|f(\mathbf{e}_h, \mathbf{p}_\ell) - \hat{\mathbf{e}}_t\|_2$	0.730	0.753	0.745
$sim[n(h), \mathrm{HED}_{\mathrm{entity}}]$	0.173	0.754	0.746
$sim[n(\ell), \mathrm{HED}_{\mathrm{non}}]$	0.435	0.746	0.754

[0141] 从表5中的结果可以看出三个主要的观察结果。首先，预测谓词 表示p_l在给出的框架中具有最重要的贡献。第一项独立达到0.728的精 度。这是因为谓词的数量1,837比训练问题的数量75,910少得多。第 二，预测的头部实体表示$\hat{\mathbf{e}}_h$可以在联合学习中补偿p_l。使用$\hat{\mathbf{e}}_h$时，精 度从0.728提高到0.745。第二项独立地实现了低精度，因为实体总数 N太大，例如FB2M中的N＝1,963,115。第三，谓词名称n(l)将KEQA 的性能提高了1.1％。这可以用一些话语与相应的谓词名称共享几个单 词来解释。

[0142] F.一些结论

[0143] 基于知识图的问题回答是一个关键问题，因为其使普通用户能够 通过自然语言轻松地访问大型知识图中有价值但复杂的信息。这也是 一个具有挑战性的问题，因为谓词可以有不同的自然语言表达。机器 很难捕捉它们的语义信息。此外，即使假设问题的实体名称被正确识 别，实体名称和部分名称的模糊性仍然会使候选实体的数量很大。

[0144] 为了弥补这一差距，本文公开了基于知识图嵌入的新型的问题回 答问题的实施方式，并且给出了简单有效的KEQA框架的实施方式。KEQA框架旨在解决简单的问题，即QA-KG中最常见的问题类型。KEQA没有直接推断头部实体和谓词，而是在KG嵌入空间中联合恢复问题的头部实体、谓词和尾部实体表示。在一个或多个实施方式中，基于注意的双向LSTM模型被用来执行谓词和头部实体表示学习。由 于与KG中的所有实体进行比较既昂贵又有噪声，所以头部实体检测 模型被用来选择问题中的连续令牌作为头部实体的名称，使得候选头 部实体集将被减少到具有相同或相似名称的多个实体。给定预测事实$\hat{\mathbf{e}}_h$、$\hat{\mathbf{p}}_\ell$、$\hat{\mathbf{e}}_t$，精心设计的联合距离度量的实施方式用于测量其到所有 候选事实的距离。具有最小距离的事实作为答案返回。进行综合实验 来评估所呈现的KEQA框架实施方式的性能。在大型基准上的实验表 明，KEQA实施方式比最先进的方法获得更好的性能。

[0145] 在一个或多个实施方式中，KEQA框架实施方式可以在各种场景 中扩展。扩展包括但不限于(i)KEQA实施方式基于预先训练的KG 嵌入执行问题回答。KEQA可以通过联合进行KG嵌入和问题回答来 推进。(ii)现实世界的知识图和训练问题经常动态更新。KEQA框架实施方式可以被扩展以处理这样的场景。

[0146] G.系统实施方式

[0147] 在实施方式中，本专利文献的各方面可涉及、可包括一个或多个 信息处理系统/计算系统或可在一个或多个信息处理系统/计算系统上 实现。计算系统可以包括可操作来运算、计算、确定、分类、处理、 发送、接收、检索、发起、路由、切换、存储、显示、通信、展示、

检测、记录、再现、处置或利用任何形式的信息、情报或数据的任何 手段或手段的集合。例如,计算系统可以是或可以包括个人计算机(例 如笔记本计算机)、平板计算机、平板电脑、个人数字助理(PDA)、 智能电话、智能手表、智能包、服务器(例如刀片服务器或机架式服务器)、网络存储设备、照相机或任何其他合适的设备,并且可以在尺 寸、形状、性能、功能和价格上变化。计算系统可以包括随机存取存 储器(RAM)、一个或多个处理资源,例如中央处理单元(CPU)或 者硬件或软件控制逻辑、ROM和/或其他类型的存储器。计算系统的 附加组件可以包括一个或多个磁盘驱动器、用于与外部设备以及诸如 键盘、鼠标、触摸屏和/或视频显示器的各种输入和输出(I/O)设备 通信的一个或多个网络端口。计算系统还可以包括一条或多条总线, 可操作为在各种硬件组件之间发送通信。

[0148] 图8描绘了根据本公开实施方式的计算设备/信息处理系统(或计 算系统)的简化框图。应当理解,用于系统800所示的功能可以操作 来支持计算系统的各种实施方式—尽管应当理解,计算系统可以被不 同地配置并包括不同的组件,包括具有如图8所示的更少或更多的组 件。

[0149] 如图8所示,计算系统800包括提供计算资源并控制计算机的一 个或多个中央处理单元(CPU)801。CPU 801可以用微处理器等实现, 并且还可以包括一个或多个图形处理单元(GPU)819和/或用于数学 计算的浮点协处理器。系统800还可以包括系统存储器802, 其可以 是随机存取存储器(RAM)、只读存储器(ROM)或两者的形式。

[0150] 还可以提供多个控制器和外围设备,如图8所示。输入控制器803 表示到各种输入设备804(例如键盘、鼠标、触摸屏和/或触笔)的接 口。计算系统800还可以包括用于与一个或多个存储设备808接口连 接的存储控制器807,每个存储设备808包括诸如磁带或磁盘的存储 介质,或者可以用于记录操作系统、实用程序和应用程序的指令程序 的光学介质,该操作系统、实用程序和应用程序可以包括实现本发明 各个方面的程序的实施方式。存储设备808也可用于存储已处理数据 或根据本发明待处理的数据。系统800还可以包括显示控制器809, 用于提供到显示设备811的接口,显示设备811可以是阴极射线管 (CRT)、薄膜晶体管(TFT)显示器、有机发光二极管、电致发光面 板、等离子面板或其他类型的显示器。计算系统800还可以包括一个 或多个外围设备806的一个或多个外围设备控制器或接口805。外围 设备的示例可以包括一个或多个打印机、扫描仪、输入设备、输出设 备、传感器等。通信控制器814可以与一个或多个通信设备815接口 连接,这使得系统800能够通过包括互联网、云资源(例如以太网云、 以太网光纤通道(FCoE)/数据中心桥接(DCB)云等)、局域网(LAN)、 广域网(WAN)、存储区域网(SAN)在内的各种网络中的任何一种 或通过包括红外信号的任何合适的电磁载波信号连接到远程设备。

[0151] 在图示的系统中,所有主要系统组件可以连接到总线816,总线 816可以代表一条以上的物理总线。然而,各种系统组件可能在物理 上彼此接近,也可能不接近。例如,输入数据和/或输出数据可以从一 个物理位置远程发送到另一个物理位置。此外,可以通过网络从远程 位置(例如,服务器)访问实现本发明各个方面的程序。这种数据和/ 或程序可以通过各种机器可读介质中的任何一种来传送,包括但不限 于:诸如硬盘、软盘和磁带的磁介质;诸如CD-ROM和全息设备等光 学介质;磁光介质;以及专门配置为存储或存储并执行程序代码的硬 件设备,例如专用集成电路(ASIC)、可编程逻辑设备(PLD)、闪存 设备以及ROM和RAM设备。

[0152] 本发明的各方面可以用指令编码在一个或多个非暂时性计算机可 读介质上，该指令用于一个或多个处理器或处理单元以使得步骤得以 执行。应当注意，一个或多个非暂时性计算机可读介质应当包括易失 性和非易失性存储器。应当注意，替代实现方案是可能的，包括硬件 实现方案或软件/硬件实现方案。硬件实现的功能可以使用ASIC、可 编程阵列、数字信号处理电路等来实现。因此，任何权利要求中的术 语“装置”旨在涵盖软件和硬件实现方案。类似地，本文使用的术语 “计算机可读介质”包括其上包含指令程序的软件和/或硬件，或者它 们的组合。考虑到这些实施方案，应当理解，附图和所附描述提供了 本领域技术人员编写程序代码(即软件)和/或制造电路(即硬件)以 执行所需处理所需的功能信息。

[0153] 应当注意，本发明的实施方式还可以涉及具有非暂时性、有形计 算机可读介质的计算机产品，该计算机可读介质上具有用于执行各种 计算机实现的操作的计算机代码。介质和计算机代码可以是为本发明 的目的专门设计和构造的那些，或者它们可以是相关领域技术人员已 知或可获得的类型。有形计算机可读介质的例子包括但不限于：诸如 硬盘、软盘和磁带的磁介质；诸如CD-ROM和全息设备等的光学介质； 磁光介质；以及专门配置为存储或存储并执行程序代码的硬件设备， 例如专用集成电路(ASIC)、可编程逻辑设备(PLD)、闪存设备以及 ROM和RAM设备。计算机代码的例子包括机器代码，例如由编译器 产生的代码，以及包含由计算机使用解释器执行的高级代码的文件。本发明的实施方式可以全部或部分实现为机器可执行指令，其可以在 由处理设备执行的程序模块中。程序模块的示例包括库、程序、例程、 对象、组件和数据结构。在分布式计算环境中，程序模块可以物理地 位于本地、远程或两者兼有的环境中。

[0154] 本领域技术人员将认识到，没有任何计算系统或编程语言对本发 明的实践是至关重要的。本领域技术人员还将认识到，上述许多元件 可以在物理上和/或功能上分成子模块或组合在一起。

[0155] 本领域技术人员将会理解，前面的示例和实施方式是示例性的， 而不限制本公开的范围。本领域技术人员在阅读说明书和研究附图后 显而易见的所有置换、增强、等同、组合和改进都包含在本公开的真 实精神和范围内。还应当注意，任何权利要求的元素可以用不同的方 式布置，包括具有多个从属性、配置和组合。

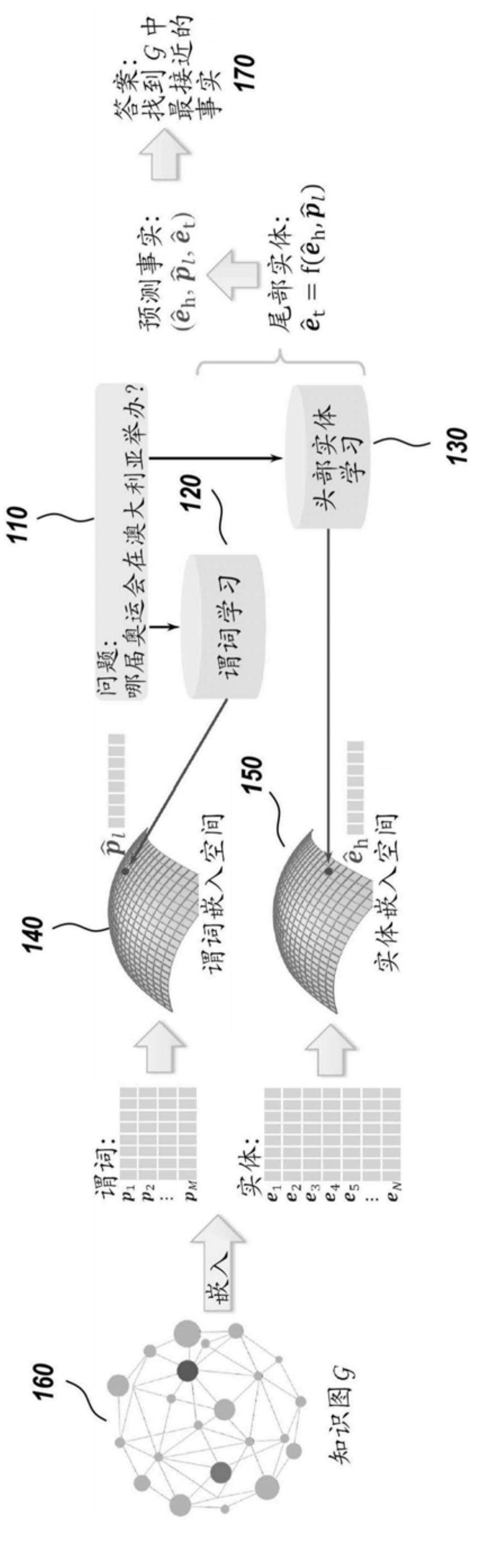

100
110
问题:
哪届奥运会在澳大利亚举办?
120
谓词学习
130
头部实体学习
140
$\hat{p}_l$
谓词嵌入空间
150
$\hat{e}_h$
实体嵌入空间
160
知识图 $\mathcal{G}$
嵌入
谓词:
p_1
p_2
⋮
p_M
实体:
e_1
e_2
e_3
e_4
e_5
⋮
e_N
预测事实:
$(\hat{e}_h, \hat{p}_l, \hat{e}_t)$
尾部实体:
$\hat{e}_t = f(\hat{e}_h, \hat{p}_l)$
答案:
找到 $\mathcal{G}$ 中最接近的事实
170

图1

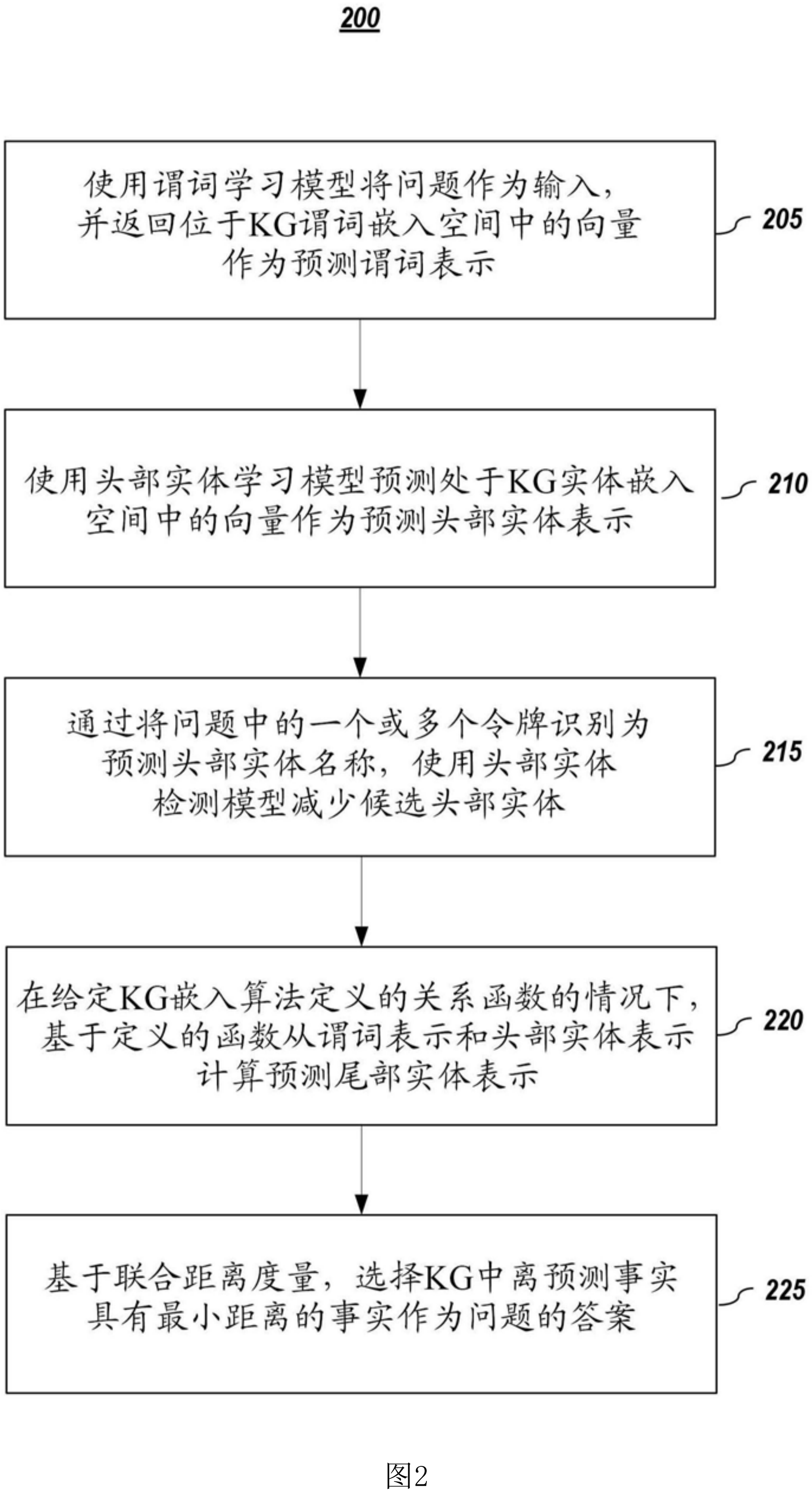

200
使用谓词学习模型将问题作为输入，并返回位于KG谓词嵌入空间中的向量作为预测谓词表示
205
使用头部实体学习模型预测处于KG实体嵌入空间中的向量作为预测头部实体表示
210
通过将问题中的一个或多个令牌识别为预测头部实体名称，使用头部实体检测模型减少候选头部实体
215
在给定KG嵌入算法定义的关系函数的情况下，基于定义的函数从谓词表示和头部实体表示计算预测尾部实体表示
220
基于联合距离度量，选择KG中离预测事实具有最小距离的事实作为问题的答案
225

图2

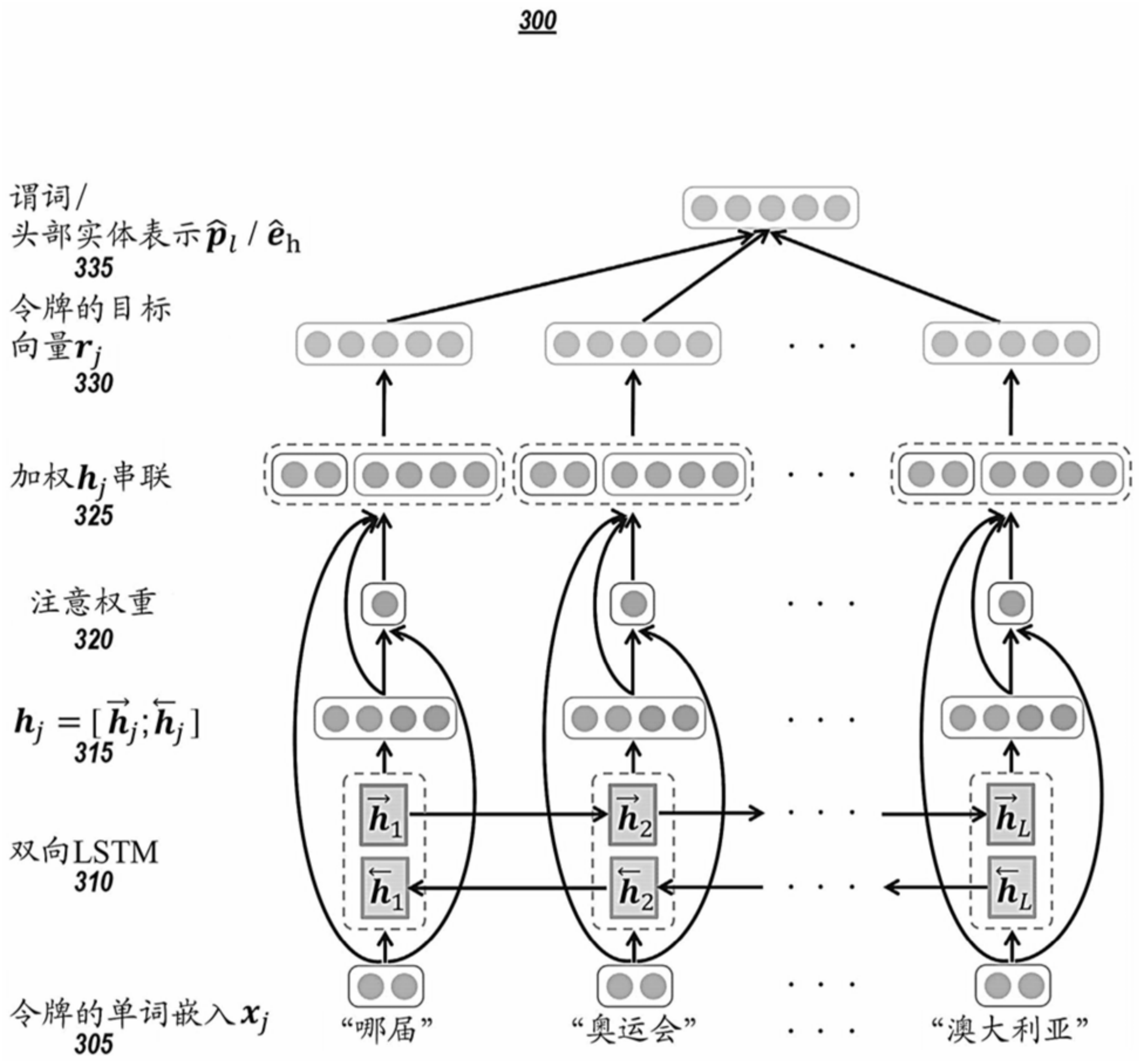

300
谓词/
头部实体表示$\hat{p}_l$ / $\hat{e}_h$
335
令牌的目标
向量r_j
330
加权h_j串联
325
注意权重
320
$h_j = [\overrightarrow{h}_j; \overleftarrow{h}_j]$
315
双向LSTM
310
令牌的单词嵌入x_j
305
$\overrightarrow{h}_1$
$\overleftarrow{h}_1$
$\overrightarrow{h}_2$
$\overleftarrow{h}_2$
$\overrightarrow{h}_L$
$\overleftarrow{h}_L$
“哪届”
“奥运会”
“澳大利亚”

图3

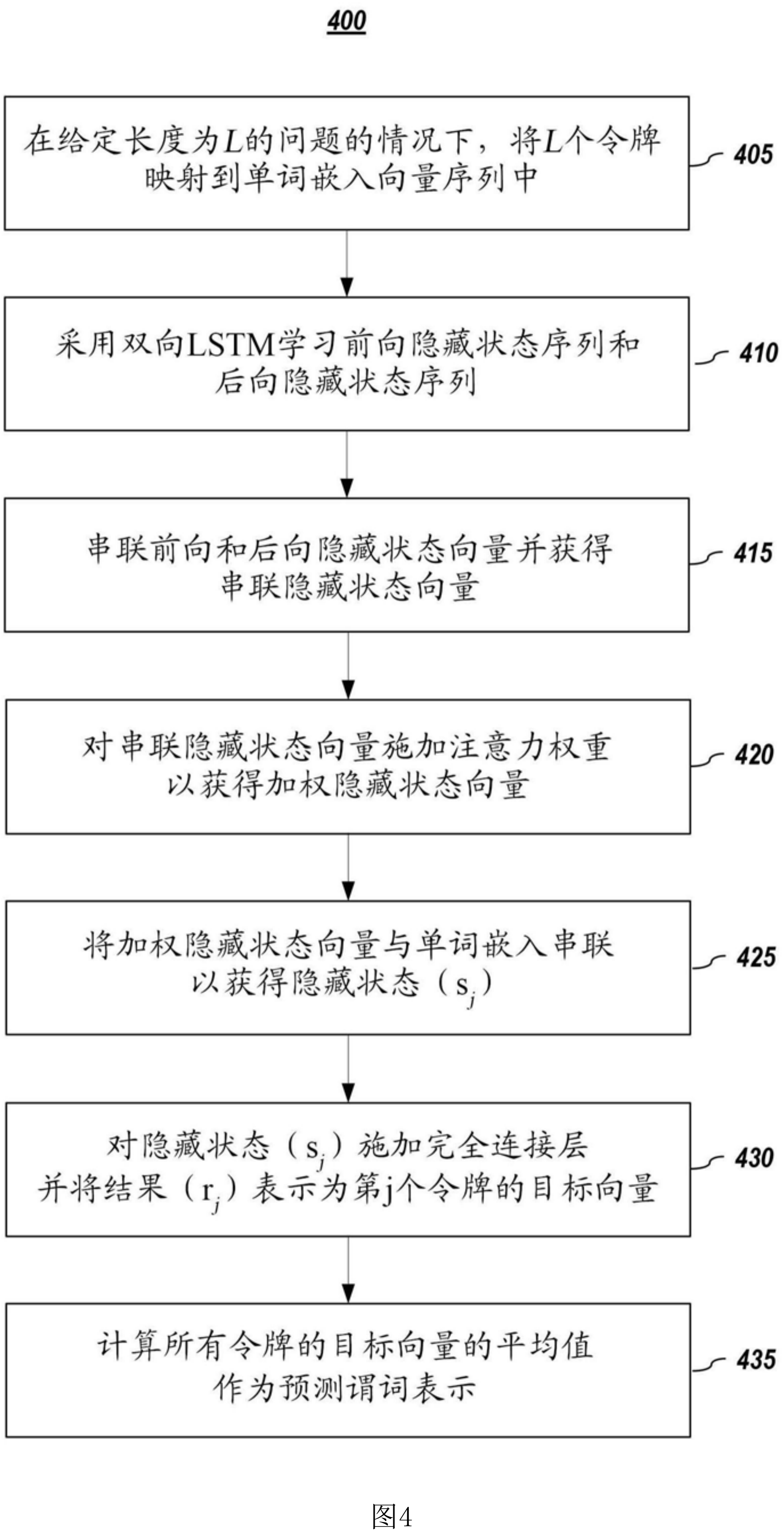

400
在给定长度为L的问题的情况下，将L个令牌映射到单词嵌入向量序列中
405
采用双向LSTM学习前向隐藏状态序列和后向隐藏状态序列
410
串联前向和后向隐藏状态向量并获得串联隐藏状态向量
415
对串联隐藏状态向量施加注意力权重以获得加权隐藏状态向量
420
将加权隐藏状态向量与单词嵌入串联以获得隐藏状态（s_j）
425
对隐藏状态（s_j）施加完全连接层并将结果（r_j）表示为第j个令牌的目标向量
430
计算所有令牌的目标向量的平均值作为预测谓词表示
435

图4

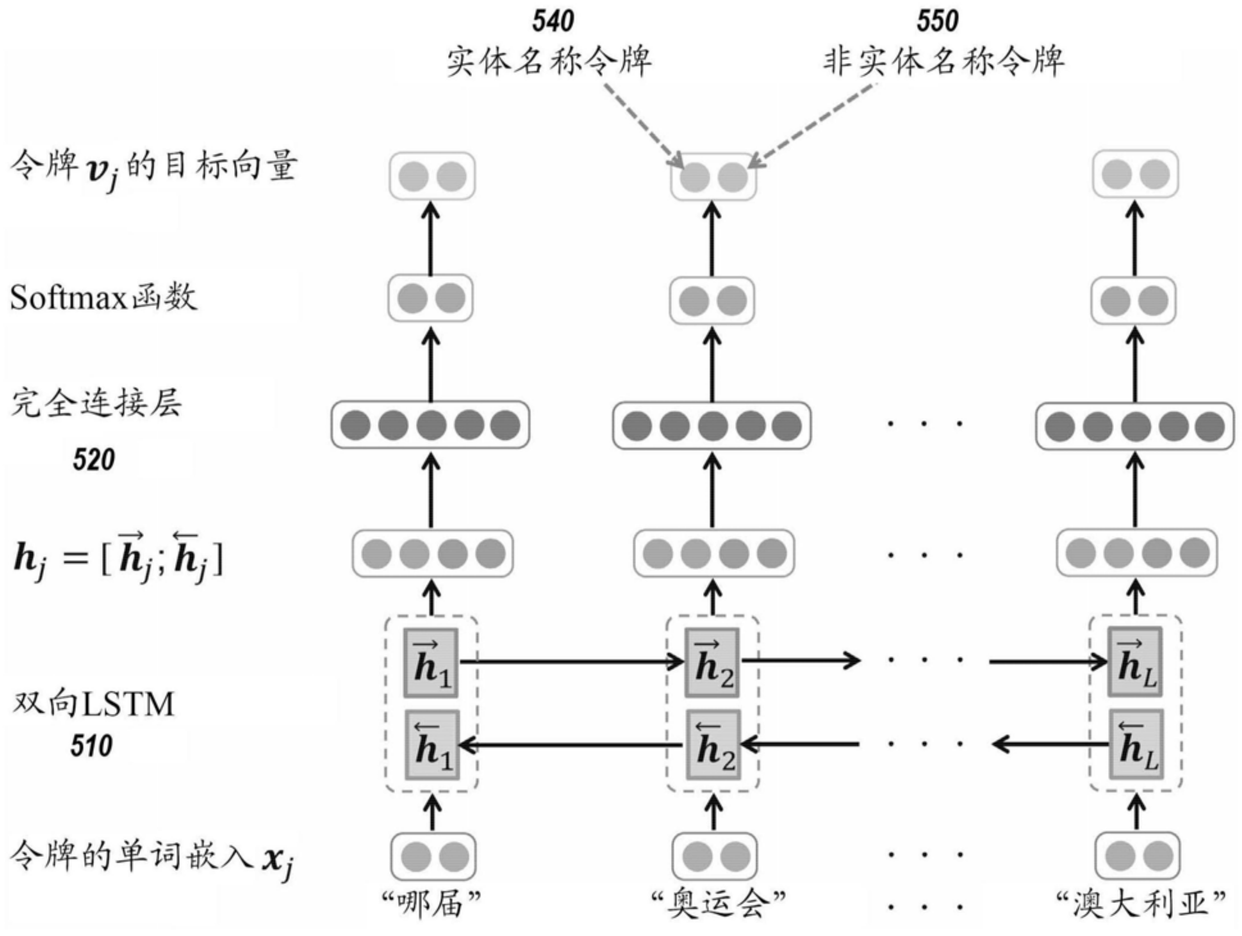

500
540
实体名称令牌
550
非实体名称令牌
令牌 $\boldsymbol{v}_j$ 的目标向量
Softmax函数
完全连接层
520
$\boldsymbol{h}_j = [\overrightarrow{\boldsymbol{h}}_j; \overleftarrow{\boldsymbol{h}}_j]$
双向LSTM
510
令牌的单词嵌入 $\boldsymbol{x}_j$
$\overrightarrow{\boldsymbol{h}}_1$
$\overleftarrow{\boldsymbol{h}}_1$
$\overrightarrow{\boldsymbol{h}}_2$
$\overleftarrow{\boldsymbol{h}}_2$
$\overrightarrow{\boldsymbol{h}}_L$
$\overleftarrow{\boldsymbol{h}}_L$
“哪届”
“奥运会”
“澳大利亚”

图5

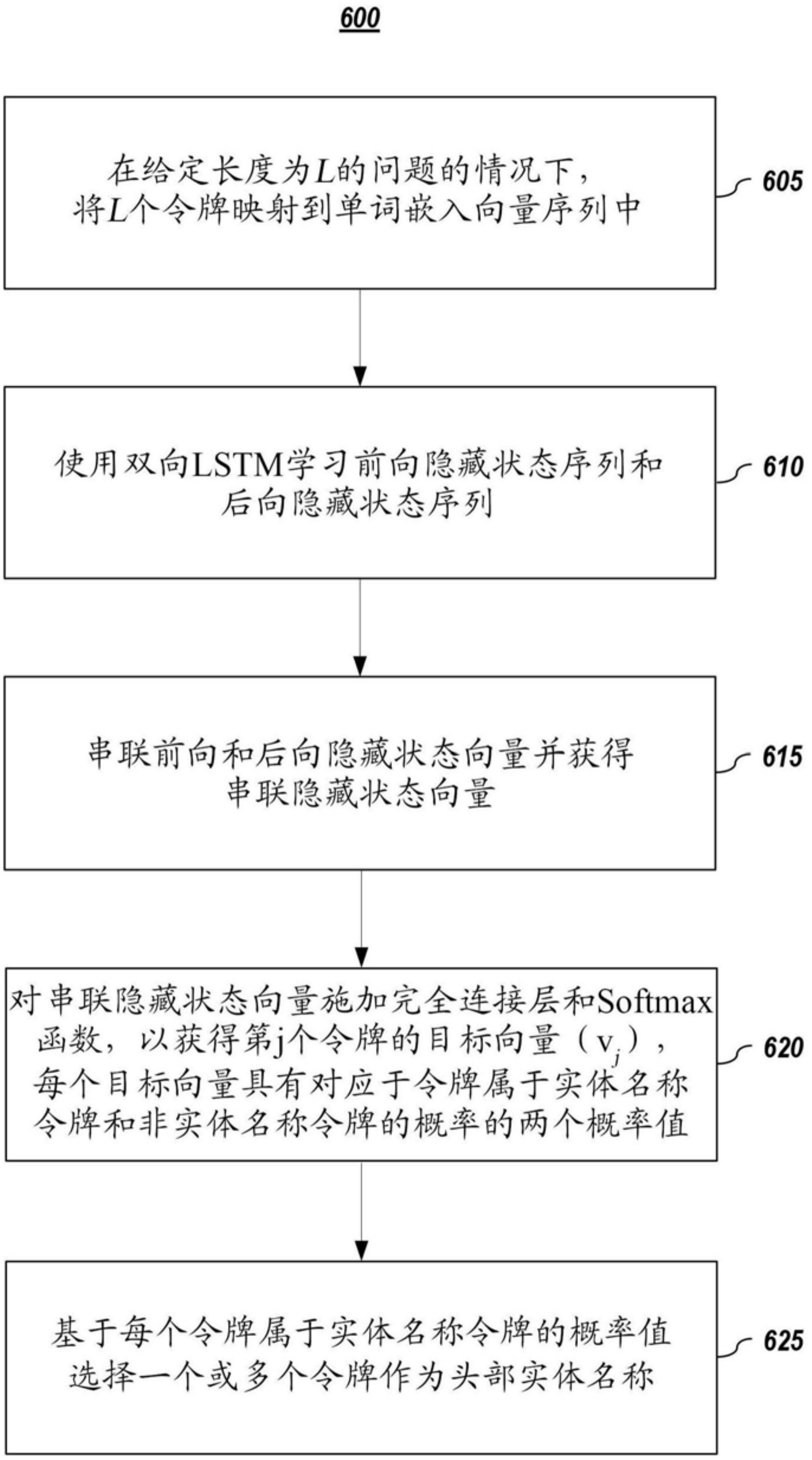

600
在给定长度为L的问题的情况下，将L个令牌映射到单词嵌入向量序列中
605
使用双向LSTM学习前向隐藏状态序列和后向隐藏状态序列
610
串联前向和后向隐藏状态向量并获得串联隐藏状态向量
615
对串联隐藏状态向量施加完全连接层和Softmax函数，以获得第j个令牌的目标向量（v_j），每个目标向量具有对应于令牌属于实体名称令牌和非实体名称令牌的概率的两个概率值
620
基于每个令牌属于实体名称令牌的概率值选择一个或多个令牌作为头部实体名称
625

图6

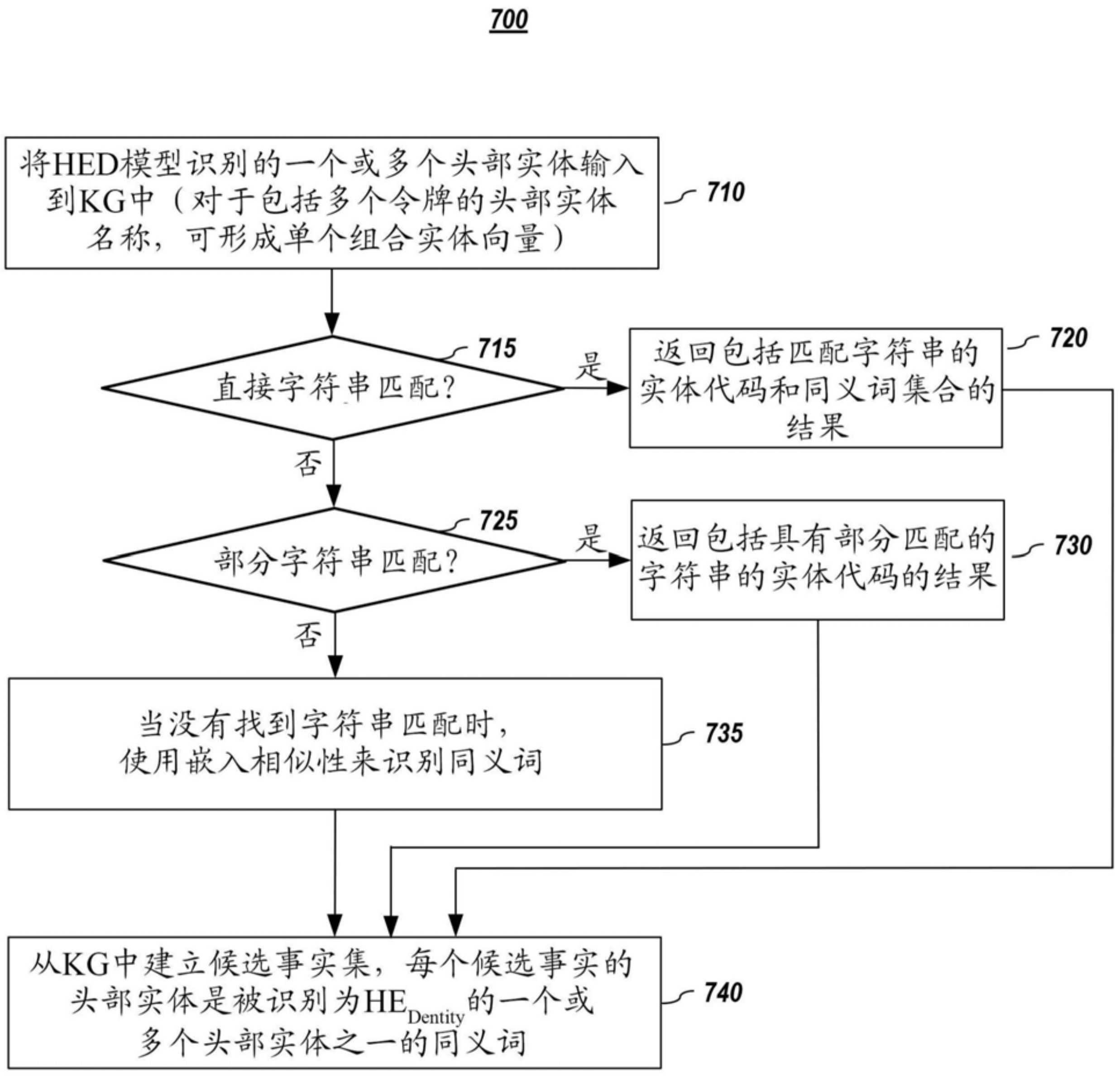

700
将HED模型识别的一个或多个头部实体输入到KG中（对于包括多个令牌的头部实体名称，可形成单个组合实体向量）
710
直接字符串匹配？
715
是
返回包括匹配字符串的实体代码和同义词集合的结果
720
否
部分字符串匹配？
725
是
返回包括具有部分匹配的字符串的实体代码的结果
730
否
当没有找到字符串匹配时，使用嵌入相似性来识别同义词
735
从KG中建立候选事实集，每个候选事实的头部实体是被识别为$HE_{Dentity}$的一个或多个头部实体之一的同义词
740

图7

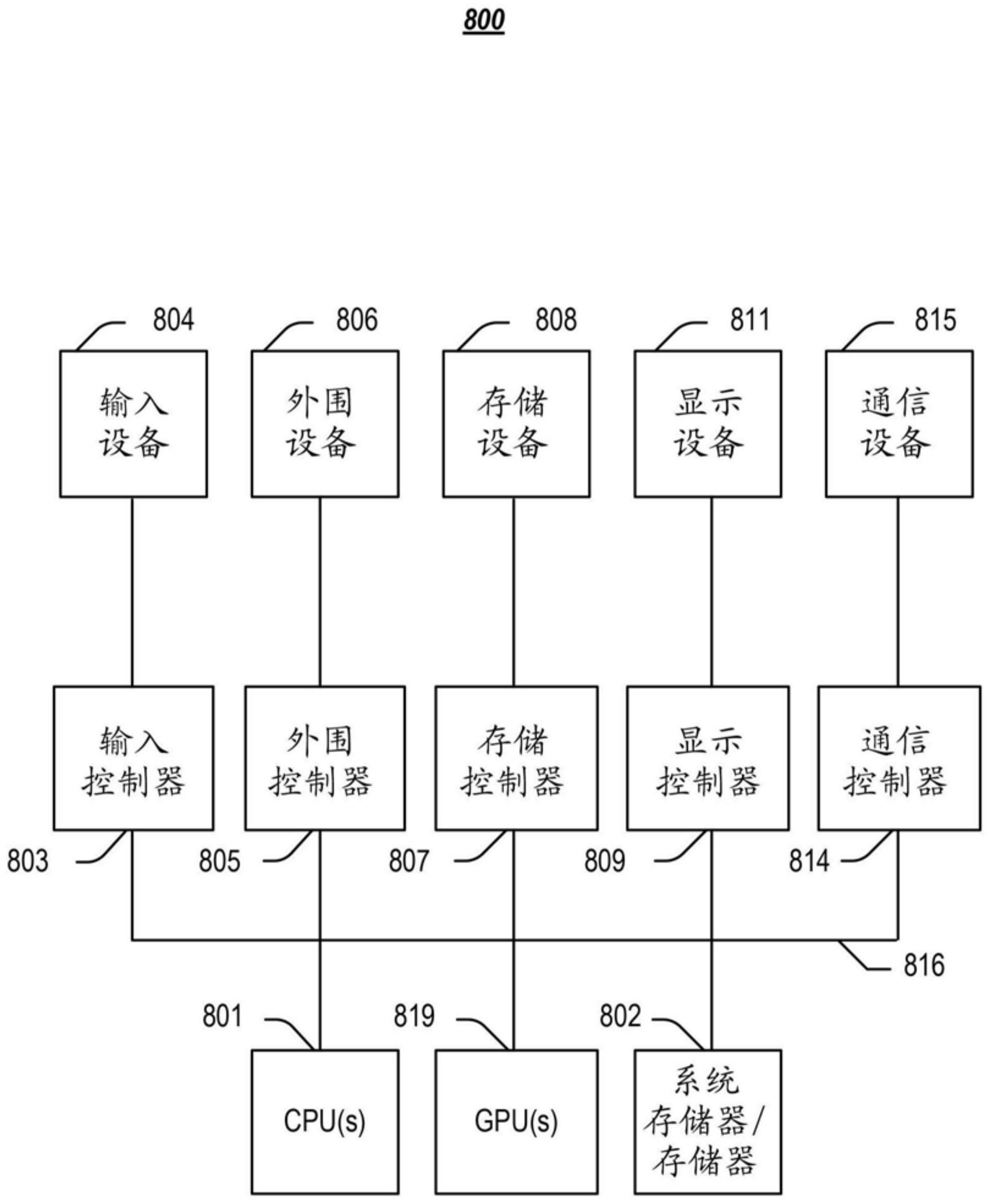

800
804
输入设备
806
外围设备
808
存储设备
811
显示设备
815
通信设备
输入控制器
外围控制器
存储控制器
显示控制器
通信控制器
803
805
807
809
814
816
801
819
802
CPU(s)
GPU(s)
系统存储器/存储器

图8