Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Knowledge-based question answering using the semantic embedding space

Min-Chul Yang[a], Do-Gil Lee[b], So-Young Park[c], Hae-Chang Rim[a,*]

[a] *Department of Computer & Radio Communications Engineering, Korea University, Seoul, Republic of Korea*
[b] *Research Institute of Korean Studies, Korea University, Seoul, Republic of Korea*
[c] *Department of Game Design and Development, Sangmyung University, Seoul, Republic of Korea*

A R T I C L E I N F O

Keywords:
Question answering
Knowledge base
Embedding model
Neural networks
Labeled-LDA
Distributional semantics

A B S T R A C T

Semantic transformation of a natural language question into its corresponding logical form is crucial for knowledge-based question answering systems. Most previous methods have tried to achieve this goal by using syntax-based grammar formalisms and rule-based logical inference. However, these approaches are usually limited in terms of the coverage of the lexical trigger, which performs a mapping task from words to the logical properties of the knowledge base, and thus it is easy to ignore implicit and broken relations between properties by not interpreting the full knowledge base. In this study, our goal is to answer questions in any domains by using the semantic embedding space in which the embeddings encode the semantics of words and logical properties. In the latent space, the semantic associations between existing features can be exploited based on their embeddings without using a manually produced lexicon and rules. This embedding-based inference approach for question answering allows the mapping of factoid questions posed in a natural language onto logical representations of the correct answers guided by the knowledge base. In terms of the overall question answering performance, our experimental results and examples demonstrate that the proposed method outperforms previous knowledge-based question answering baseline methods with a publicly released question answering evaluation dataset: WebQuestions.

1. Introduction

Question answering (QA) is concerned with building systems that automatically answer questions posed by humans in a natural language by exploiting the techniques of natural language processing (NLP) and information retrieval (IR). In general, QA systems can retrieve and extract answers from natural language documents in unstructured Web-data by using a structured query that is semantically associated with a given question.

As an alternative form of QA implementation, knowledge-based question answering (KB-QA) requires a structured database called a knowledge base (KB) because KB-QA systems simply extract answers from the structured knowledge base instead of the unstructured Web-data. In other words, these systems can use information to resolve a query without having to navigate to other sites and assemble the information themselves. A KB is a technique used to store large volumes of factual information in a structured format, which is constructed based on well-written textual data, such as Wikipedia[1] and human contributions. In particular, Freebase,[2] one of the publicly available databases, allows users to add new information and to modify incorrect information even if the users are not experts. Freebase comprises a huge volume of facts as multi-relational data in a triple format: $<$ *subject entity, logical predicate(= relation), object entity* $>$. For KB-QA, the central problem is how to transform the input question into its corresponding structured query for KB as the logical form. This task involves the transformation of various natural language representations in an unstructured format into semantically similar KB-properties in the structured and canonical format.

The latest KB-QA systems [2–4,9,13,21,24,37] employ the semantic parsing technique to exploit the mappings between the lexical phrases and logical predicates in the KB. Semantic parsing is a learning task that maps natural language statements onto formal meaning representations of their underlying meanings. In KB-QA, this technique is used to transform given natural language questions into structured queries for KB. However, previously proposed methods have the following three limitations. (1) The meaning of a logical

* Corresponding author. Tel.: +82 2 3290 3195; fax: +82 2 929 7914.
E-mail addresses: mcyang@nlp.korea.ac.kr (M.-C. Yang), motdg@korea.ac.kr (D.-G. Lee), ssoya@smu.ac.kr (S.-Y. Park), rim@nlp.korea.ac.kr (H.-C. Rim).

1 http://en.wikipedia.org.
2 https://www.freebase.com.

http://dx.doi.org/10.1016/j.eswa.2015.07.009
0957-4174/© 2015 Elsevier Ltd. All rights reserved.

predicate often shares different natural language expression (NLE) forms, so the lexical representations linked with a predicate may be limited in size with respect to the user inputs. (2) Entities detected by the named entity recognition (NER) component are used to combine the logical forms with the logical predicates, and thus their types should also be consistent with the predicates. However, most of the NER components used in existing KB-QA systems are independent of the NLE-to-predicate mapping procedure. (3) Semantic parsing-based approaches have difficulty fully representing various properties of the KB because the appropriate knowledge information may be found for given lexical statements by exploring the lexicon rather than by providing comprehensive descriptions.

In general, embedding models in the NLP area, such as Word2Vec [22], are used to represent the meanings of words as low-dimensional vectors. These distributed representations of words can also be used to memorize many linguistic regularities and patterns. The basic idea of this approach is that the vector of a word has a similar weight to the vectors of its surrounding words because co-occurring words within a limited range are likely to share similar semantics and contexts. In the same manner, we propose a semantic embedding space that jointly encodes words and KB-properties based on their semantic relationships. Joint embeddings [34] have been used to learn various representations of items with different types, but we focus on building semantic mappings for KB-QA over the embedding space. The semantic embedding space has three roles, as follows. (1) Semantic embedding jointly encodes words and KB-properties into the same space based on their semantic associations. (2) We can simply compute the semantic similarities of two given features (a word or KB-property) using the dot product operation between two embedding vectors connected by the features. (3) The semantics of words or KB-properties can be represented as distributed values.

The remainder of this paper is organized as follows. We examine previous research related to this area in Section 2 and we investigate KB-QA in Section 3. In Section 4, we introduce the setup of the proposed method and the three stages of our KB-QA system are described in Sections 5–Section 7. In Section 8, we present our experimental results and analysis. Finally, Section 9 gives our conclusions.

2. Related work

Semantic parsers can be used to transform natural language sentences into their machine interpretable corresponding logical forms in a fully formal language. Supervised semantic parsers [24,38,39] are highly reliant on < *sentence, semantic annotation* > pairs for lexical trigger extraction and model training to map natural language expressions onto formal meaning representations. Due to their requirement for data annotation, these methods are usually restricted to specific domains (such as GEO, ATIS, and JOBS) and they struggle with coverage issues caused by the limited size of lexical triggers. Thus, other studies have employed weakly supervised semantic parsers to reduce the amount of human supervision by using question–answer pairs [21] or *distant supervision* [19] instead of full semantic annotations. These approaches can automatically extract the < *sentence, semantic annotation* > pairs supported by the structured database, although some incorrect pairs may be obtained due to the redundant information in KBs.

As conventional QA approaches, when given a question statement, IR-based QA systems [14,18,30,32] try to retrieve, extract, and assemble answer information for a given question based on a large volume of unstructured data such as WIKIPEDIA, before generating an answer statement. Similarly, community-based question answering (CQA) systems [1,15] aim to find answers from past question–answer pairs in online social sites such as YAHOO! ANSWERS.[3]

Recently, researchers have developed open-domain systems based on large-scale KBs such as FREEBASE. Thus, semantic parsers for Open-QA may be learned using manually prepared schema [3,9], a comprehensive ontology [20], paraphrased questions that are semantically similar [4,12,13], a machine translation-perspective model [2], pairs of graph structures of the question statement and KBs with QA-pairs [37], a formalized knowledge representation [27] and ontology-based inferences of question's syntactic structure and context [26]. The semantic parsers employed are usually unified, formal, and scalable, where they allow a question statement to be mapped onto the appropriate logical form based on precise manually prepared lexicons or schema matching. These methods might provide correct answers, but some responses cannot be provided for complex questions because the size of the lexicon may be limited (low recall). Similarly, our method also aims to obtain similar logical forms but we only use low-dimensional embeddings of *n*-grams and the KB-properties are learned from a huge volume of texts and KBs. In previous studies of KB-QA, Cai and Yates and Berant, Chou, Frostig, and Liang produced evaluation datasets where the QA-pairs were annotated by humans based on FREEBASE, i.e., FREE917 and WEBQUESTIONS, respectively. These two standard datasets have been utilized for QA evaluations in recent KB-QA systems, and we use WEBQUESTIONS for QA evaluations in the present study.

The pioneering studies of QA using embedding models [8] aimed to learn low-dimensional vector representations of words and KB-properties in the same space. This type of QA model obtains answers via candidate QA paths that are linked directly with each other in the KB by scoring the paths based on their similarities in the learnt embedding space. In our study, the QA paths are defined as being equal to the answer derivations. However, previous embedding models were restricted to capturing various QA paths, which could not handle various complex types of questions. The most similar KB-QA systems to our proposed method [7,36] focused on the semantic associations between words and the KB-properties of candidate answers, where the method proposed by Yang et al. was the initial version of our method. Similar to the method suggested by [8], Bordes et al. also aimed to learn embeddings of questions and answers based on question–answer pairs in ReVerb [11] and those of paraphrased questions in YAHOO ANSWERS. However, these previous methods used highly sophisticated inferences to handle long QA paths, thereby obtaining rich representations of the training QA paths and the surrounding subgraphs of the KB, whereas our method obtains high-quality semantic links (QA paths) directly from a large-scale corpus instead of using ReVerb, which was constructed previously. In the method described in Yang et al., the aim is to construct the joint relational embeddings of words and KB-properties based on semantic links obtained directly from WIKIPEDIA and SATORI,[4] which is a KB produced by Microsoft Research. Unlike [36], we employ the following features: (1) we use question category features to identify expected answer types, (2) we filter out unassociated semantic links with distributional semantics during post-processing, and (3) feature-level representations are used to identify semantic links before training the embedding space. We consider that feature-level semantic links provide more robust semantic mappings for KB-QA than pattern-level versions.

3. Knowledge-based question answering

3.1. Challenges in KB-QA

Basically, KB-QA systems interpret the given question statement and then map it onto a logical representation of the correct answer in

[3] https://answers.yahoo.com/.

[4] http://searchengineland.com/library/bing/bing-satori.

the KB. In this study, we focus on the following challenges that need to be solved during KB-QA.

KB interpretation. KBs are large-scale structured databases, which include large volumes of multi-relational data. For KB-QA, the KB is a collection that is used to fetch answer entities, which must interpret the knowledge information fully. However, there are hidden and broken relations between the properties of the KB, especially in Freebase. Thus, the KB cannot be exploited without further processing. To alleviate this problem, we exploit embedding models that can utilize the relations between any properties, but we also recover any incomplete relations based on the observed relations and other superordinate relations in the hierarchical structure.

Question category detection. Open-domain QA aims to provide an answer statement to a given question statement in any category. For example, *"what movies has tom hanks starred in?"* and *"how many movies has tom hanks starred in?"* are two questions that share a similar context but their goals are different. The first question requests a list of the movies starring Tom Hanks and the second question requires the number of movies in which Tom Hanks has starred. Similarly, we build separate properties for question categories to understand the user's needs. Note that these properties are not present in the KB, so we need to examine them in training QA-pairs.

Entity span identification. A question statement usually includes at least one entity. To detect this entity, conventional QA systems mainly use an NER tool or external resource to compare candidate entities with their descriptions in Wikipedia. However, the target entity identification tasks used in these methods are independent of other tasks in KB-QA, which leads to the error propagation problem. Therefore, we cannot find the correct answer if we detect incorrect target entities. For example, the question *"where is the city of david"* contains ambiguous entities, i.e., David, The City, and The City of David. Most NER tools recognize David as an entity because David is a commonly used personal name. Thus, all of the possible entity spans of the question statement (David, The City, and The City of David) should be considered as candidate entities, rather than determining David as the incorrect target entity.

Predicate mapping. Semantic parsing-based KB-QA systems exploit syntax-based grammar formalisms, such as Combinatory Categorial Grammar [31] and Dependency-based Compositional Semantics [21], to map a natural language statement onto a logical form. As a mapping table, a lexicon is used to link each lexical representation to its corresponding properties in the KB. Most lexical triggers are based on rule matching or schema matching, which are limited to answering various different question statements. However, we can find the corresponding logical predicate if discriminative words are identified. For example, the appropriate logical predicate for the question *"when was barack obama born?"*, can be found (people.person.place_of_birth) if we analyze the semantics of the significant unigrams (*when* and *born*) without any syntactic analysis.

3.2. *Knowledge graph (base):* Freebase

In this study, we use Freebase [6] as a KB for extracting answer information. Freebase is a large collaborative KB that comprises metadata produced mainly by its community members.

Freebase comprises 47 million entities and 2.7 billion facts (December 2014). Each fact is defined by a triple format that contains two entities (subject entity and object entity) and a relation (logical predicate) between them, which has the form of $<$ *subject entity, logical predicate, object entity* $>$. The *logical predicate* can be regarded as the canonical form for relation phrases that comprise at least one verb word. The *subject entity* should be a Freebase entity, whereas the *object entity* can be allocated to a number, a date-time, and a text, as well as a Freebase entity. Freebase entities refer to materials that exist in real life, such as people, locations, and organizations. However, some *object entities* cannot be transferred to one property if the entity includes too much information. These entities are defined as *event entities* in Freebase, which are compounds of some entities.

For each KB-fact, two nodes on the knowledge graph, *subject entity* and *object entity*, represent real world concepts, and an edge, *logical predicate*, represents a relation, thereby forming a bidirectional graph-like structure based on many KB facts. A *subject entity* and *logical predicate* pair can sometimes have multiple *object entities* such as the people.person.children predicate if someone has several children in real life, which produce multiple edges from one node. For example, Fig. 1 depicts graphical representations of many facts for the Barack Obama entity. As a *subject entity*, Barack Obama is on the left-hand side of all facts, whereas *object entity* is on the right-hand side and it can be assigned different values according to the *logical predicates*. In practice, the entities are recorded by a unique Freebase identifier because some entities can share an identical name, such as Forrest Gump(film) $\rightarrow$ /m/0bdjd and Forrest

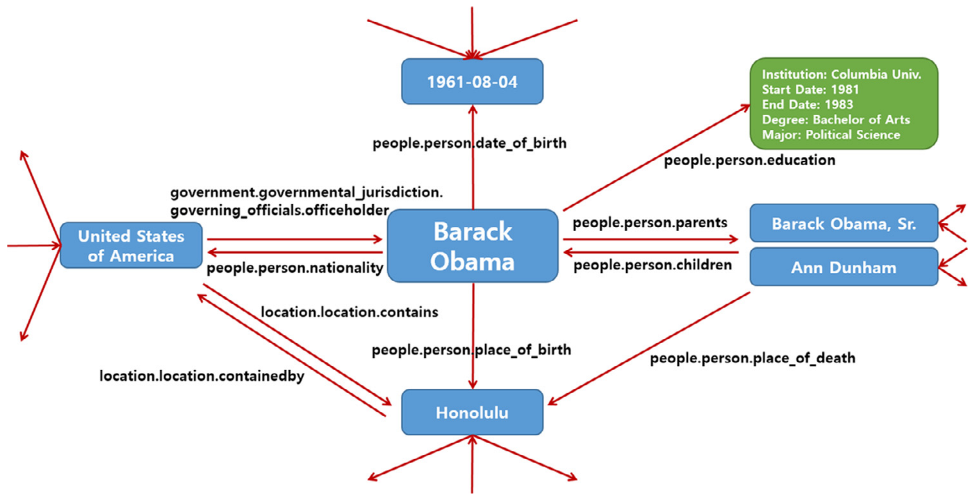

1961-08-04
people.person.date_of_birth
Institution: Columbia Univ.
Start Date: 1981
End Date: 1983
Degree: Bachelor of Arts
Major: Political Science
people.person.education
government.governmental_jurisdiction.
governing_officials.officeholder
United States of America
Barack Obama
people.person.parents
Barack Obama, Sr.
people.person.nationality
people.person.children
Ann Dunham
location.location.contains
people.person.place_of_birth
people.person.place_of_death
location.location.containedby
Honolulu

Fig. 1. Graph representations of the Barack Obama entity in Freebase. (For interpretation of the references to colour in the text, the reader is referred to the web version of this article.)

Gump(film_character) → /m/0264f7h. In FREEBASE, the *logical predicate* is usually organized according to three hierarchical levels (3-hop): the top level is the *domain* (topic), the middle level is the *type* (entity type of an entity), and the bottom level is the *attribute* (specific sense of a relation between entities). Above these levels, the *logical predicates* for an *event entity*, people.person.education, are too complex to take sub-entities because they comprise the common *logical predicates* (3-hop) as well as sub-attributes (4-hop). In terms of the types of *logical predicates*, the rectangles in blue indicate common entities linked by 3-hop *logical predicates* and the rectangle in green denotes the *event entity* connected by 4-hop *logical predicates*.

4. Setup

4.1. Problem statement

For natural language understanding, semantic transformation (parsing) aims to find semantic-related logical forms for given natural language statements. The following issues related to embedding model-based semantic transformation need to be addressed.

- All of the features of the KB (KB-properties) are represented as logical and canonical forms, and thus no information is available that links a lexical feature (word) with them. A KB-property that is a semantic concept can be related to many lexical representations posed in a natural language, while a lexical representation can also be linked to several properties of the KB. To solve this problem, we must obtain the semantic links between a relation phrase and its corresponding KB-properties from the unstructured textual data (Section 5.1). We also extract question patterns with the appropriate KB-properties to tackle question statements that start with interrogative words (Section 5.2). Next, we can learn the low-dimensional embeddings of a word and a logical property, which comprise a semantic-associated pair extracted from unstructured textual data using *distant supervision*, such that vector representations of semantically similar features are close to each other in the semantic embedding space. Thus, the meaning representations of words can be specified collaboratively using their relations with the logical properties based on the KB of conceptual data in the hierarchical and multi-relational structure (Section 6). Fig. 2 shows the procedure used for joint semantic embedding.
- KB-QA assumes that the correct answer should be present in the KB. However, examining a vast amount of knowledge information carefully is an intensive process. To reduce the computational space required for QA, we focus on ranking potential answers based on the semantic similarities between embeddings of the bag-of-words represented in the given question and those of logical representations of the potential answers. The set of candidate answers is generated by the facts as the KB constituents of possible entities that appear in the question statement. Fig. 3 shows a QA procedure with joint semantic embeddings.

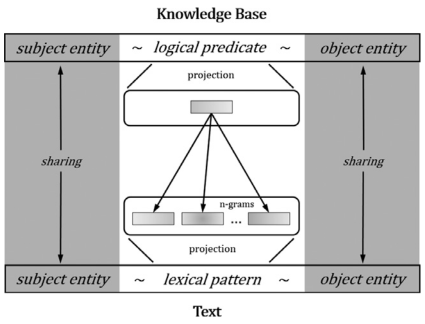


Fig. 2. Process of joint semantic embedding between the text space and knowledge base space.

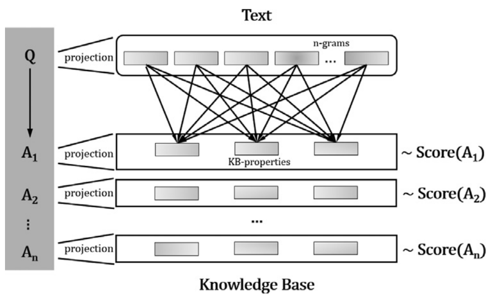


Fig. 3. Process used to answer a question Q via joint semantic embeddings between the text space and knowledge base space.

Table 1
Lexical features and logical features for the question: *where was barack obama born?*

Lexical features	
Word (*w*)	{*1w_where*, . . . ,
	{*2w_where be*, . . . ,}
	{*3w_where be #entity*, . . . ,}
Logical features	
Question category (*c*)	list
Topic domain (*d*)	people
Entity type (*t*)	people.person
Predicate (*p*)	people.person.place_of_birth

4.2. Relational features for KB-QA

Our method focuses on the semantic relationships between lexical features and logical features. The lexical features represent natural language expressions that are used commonly in people's statements, whereas the logical features indicate the logical properties of the KB, which are concept representations in the canonical forms. Table 1 shows the corresponding lexical and logical features for the question "*where was barack obama born?*" and we introduce five types of features, which we define in this study.

Lexical feature. Bag-of-words models are used to express question statements in KB-QA, as follows.

- **Word (*w*)** indicates the bag-of-words (segment) of lexical patterns in natural language statements. As lemmatized *n*-grams, the bag-of-words model is a simplifying representation used in the NLP. Therefore, frequent and common *n*-grams can be components of a lexical trigger that plays a role in mapping to KBs. For QA tasks, we focus on collecting *n*-grams of relation phrases (predicates), which are the main linking verbs between subject and object noun phrases in natural language sentences, because the relation phrases can be semantically related to logical predicates (*p*) in the logical features.
  In this study, we use three types of *n*-grams: uni-, bi-, and trigrams, as shown in Table 1. The larger *n*-grams are mainly expected to be discriminative features used to link logical features. Note that we retain the positional information for entity spans as a placeholder, *#entity*, because the entity position is important for identifying the neighbor context of the entity. According to the candidate entities, we can construct several transformed question contexts. Table 2 shows pairs of candidate entities and transformed questions for the entity-ambiguous question: "*where is the city of david?*"

Logical feature. Canonical representations of the KB are used to generate logical forms (answer derivations) for the given question statements in KB-QA. The feature set is described as follows.

Table 2
Candidate entities and the remaining statements for the question: *where is the city of david?*

Where is the city of David?	
Candidate entity	Transformed question
David	*Where is the city of #entity?*
The City	*Where is #entity of David?*
The City of David	*Where is #entity?*

- **Question category (*c*)** represents the expected answer types of question statements. We use a fixed set of question categories: {list|count} (two values). The answer types differ mainly depending on interrogative words in the question. As mentioned earlier, the question *"what movies has tom hanks starred in?"* is expected to be answered by a list of types, and the count of the list will be the answer to the question "*how many movies has tom hanks starred in?*." The first question starts with "*what*" whereas the second question's interrogative phrase is "*how many*."
- **Topic domain (*d*)** depicts a topical representation of question statements. As the first part of a *logical predicate*, Freebase domains can be used to classify entities in the NER task such as people, organization, and location. The top-level property can accommodate other lower level properties, and thus the higher class allows us to bind broken connections that are semantically associated with the lower class. It is assumed that these broad semantics can smooth over the inevitable incompleteness of the KB on a large scale. This domain information usually represents the semantic generality of the target entities for QA.
- **Subject entity (*e*; entity type:*t*)** indicates the target entity of question statements. Among the words that may occur in a question, the proper noun is likely to be an entity. Some questions have ambiguous entity spans, so the entity span selection task is important for the question analysis. This task plays a supporting role when solving the other main tasks, but it is the first stage in understanding question statements. In this study, we use the Freebase types (entity types) of a subject entity to reduce the errors in our loss function. The Freebase type refers to the abstract expressions of entities, such as Barack Obama → {people.person, . . . , government.politician}.
- **Logical predicate (*p*)** denotes the canonical forms of relation phrases between a subject entity and an object entity. In contrast to the relation phrases posed in natural language, the predicate should be an unambiguous and clear representation of the specific sense. A Freebase predicate, people.person.place_of_birth, has three hierarchical levels, domain: people, type: person, attribute: place_of_birth, and it defines the relational links between entities, many of which can regard Freebase as a graphical structure. Popular entities such as celebrities and notable things may have many predicates linked to object entities because the information is rich for these entities.

4.3. Framework

Our aim is to build a semantic parser that maps a natural language question onto its corresponding logical form, which can be executed directly with KBs to obtain answer entities. Our QA system comprises two stages: 1) semantic embedding space construction and 2) embedding-based QA.

Fig. 4 shows the system architecture of our method. The left-hand diagram in yellow explains the semantic linking stage, the middle diagram in orange illustrates the construction stage of semantic embedding space, and the right-hand diagram in green depicts the embedding-based QA stage. First, we extract NL-entries with the form < *subject entity, lexical pattern, object entity* > using KB-facts with the form < *subject entity, logical predicate, object entity* >. The two tuples can share entity pairs, < *subject entity, object entity* >, whereas the attributes in the middle, *lexical pattern* and *logical predicate*, are different. This structure allows us to perform *weak (distant) supervision* [16,17,23,25,29] to map between lexical features and logical features (semantic links). In the present study, *distant supervision*

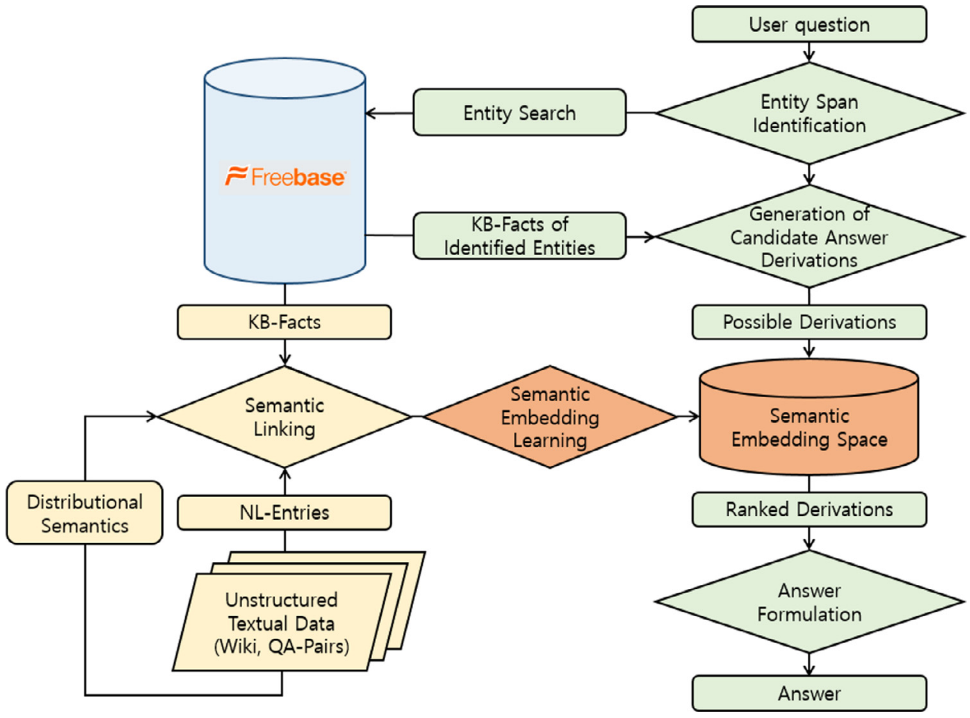
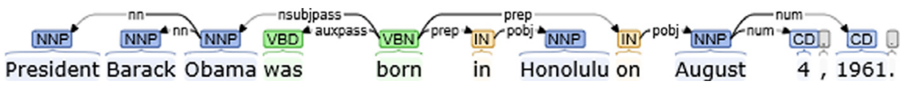


Fig. 4. System architecture of our method. (For interpretation of the references to colour in the text, the reader is referred to the web version of this article.)



NNP NNP NNP VBD VBN IN NNP IN NNP CD CD
nn nn nsubjpass auxpass prep pobj prep pobj num num
President Barack Obama was born in Honolulu on August 4 , 1961.

Fig. 5. Example of a dependency-parsed sentence: *President Barack Obama was born in Honolulu on August 4, 1961.*

is a learning scheme in which a classifier is learned given a weakly labeled training data based on the two target entities of KB-facts. During post-processing, we also filter out tuples of < *lexical pattern, logical predicate* > that are wrongly paired using distributional semantics. Based on the semantic links of < lexical feature, logical feature >, we can obtain robust embedding vectors using ranking-based relational embedding learning. As a QA step, we first generate all possible derivations from the input question statement and from its candidate entity-driven facts, before scoring each derivation using the learnt embedding models. Based on the top-scoring derivation, we obtain the answer entities from the KB and we then formulate an answer statement that depends on the predicted question category.

5. Semantic linking between lexical features and logical features

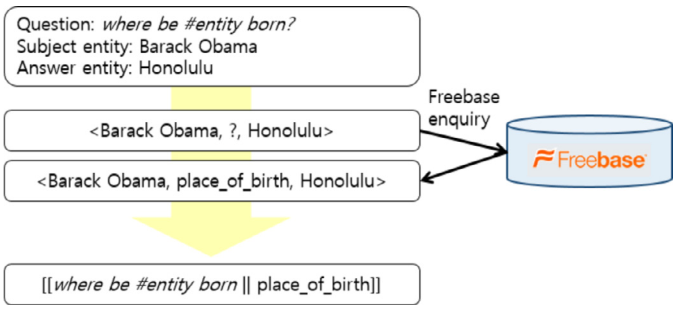
In this section, we introduce the linking process for building a set L of < lexical feature, logical feature > links as training pairs for use in relational embedding learning. First, we extract tuples of [[*lexical pattern* ||FREEBASE*predicate*]] from: (1) WIKIPEDIA, as described in Section 5.1; and (2) question–answer pairs, as described in Section 5.2. Next, we filter out the inappropriate FREEBASE*predicate* for the given context, as described in Section 5.3. As a result, we construct a set L of < lexical feature, logical feature > links and compute a score set S for the semantic links, as described in Section 5.4.

5.1. Relation phrase extraction from raw textual data

Relation phrases denote lexical representations of an existing logical predicate in the KB. Based on previous relation extraction methods with *distant supervision* [16,17,23,25,29], we extract tuples of [[relation phrase || FREEBASE *predicate*]] from the English WIKIPEDIA, which is connected closely to our KB. This task is similar to linking the properties of WIKIPEDIA and those of FREEBASE.

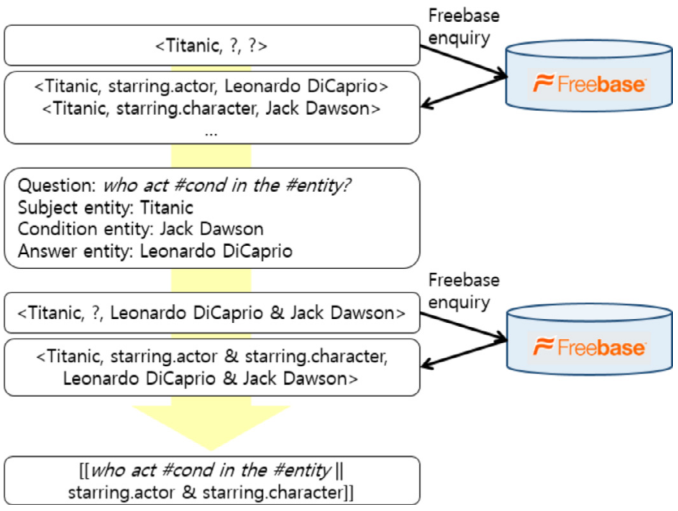
Our relation phrase extraction method with *distant supervision* is described as follows. Given a KB-fact < *subject entity,*FREEBASE *predicate, object entity* >, we extract NL-entries < *subject entity, relation phrase, object entity* >, where *relation phrase* is the shortest path between *subject entity* and *object entity* in the dependency tree of the sentences. It is assumed that any *relation phrase* in the NL-entry containing the entity pairs that occur in the KB-fact is likely to express the FREEBASE *predicate* of that fact.

For example, we first collect sentences containing at least two entities, such as the sentence in Fig. 5, which includes three entities: Barack Obama, Honolulu, and 1961-08-04. Similar to common natural language statements, the sentences in WIKIPEDIA are also affected by the entity disambiguation problem due to variations in their mentions. For an entity linking task, we only use proper nouns with hyper-links that indicate other WIKIPEDIA articles. Next, we extract two NL-entries, < Barack Obama, *be born in*, Honolulu > and < Barack Obama, *be born on*, 1961-08-04 >, from the dependency parsing results. On the opposite side, there are two KB-facts in FREEBASE, < Barack Obama, place_of_birth, Honolulu > and < Barack Obama, date_of_birth, 1961-08-04 >, each of which shares entity pairs with the NL-entries above, respectively. Note that the *subject entity* is replaced with the symbol *#entity* to keep its position, whereas the position of *object entity* is left unchanged because it is not important for analyzing question statements. Finally, we obtain two tuples: [[*relation phrase* ||FREEBASE *predicate*]]: [[*#entity be born in* || people.person.place_of_birth]] and [[*#entity be born on* || people.person.date_of_birth]].



Question: where be #entity born?
Subject entity: Barack Obama
Answer entity: Honolulu
Freebase enquiry
<Barack Obama, ?, Honolulu>
<Barack Obama, place_of_birth, Honolulu>
Freebase
[[where be #entity born || place_of_birth]]

Fig. 6. Example of the FREEBASE*predicate* linking process for a single-related question in WEBQUESTIONS: *where was obama born?*



Freebase enquiry
<Titanic, ?, ?>
<Titanic, starring.actor, Leonardo DiCaprio>
<Titanic, starring.character, Jack Dawson>
...
Freebase
Question: who act #cond in the #entity?
Subject entity: Titanic
Condition entity: Jack Dawson
Answer entity: Leonardo DiCaprio
Freebase enquiry
<Titanic, ?, Leonardo DiCaprio & Jack Dawson>
<Titanic, starring.actor & starring.character, Leonardo DiCaprio & Jack Dawson>
Freebase
[[who act #cond in the #entity || starring.actor & starring.character]]

Fig. 7. Example of the FREEBASE*predicate* linking process for a multi-related question in WEBQUESTIONS: *who plays dawson in titanic?*

5.2. Question pattern extraction from QA-pairs

WIKIPEDIA articles include no information to facilitate the utilization of interrogative words and the question category (c) depends greatly on the object entity (answer); therefore, it is difficult to distinguish some questions that only comprise different *5W1H* words, e.g., "{*when*|*where*} *was barack obama born?*." Hence, we need to extract question patterns from QA-pairs in the training set of FREE917 [9] and the WEBQUESTIONS [3] evaluation datasets.

Fig. 6 shows how to link the given single-related question to the FREEBASE *predicate*. In this case, the question does not include the exact spans matched to the Barack Obama entity because common questions may ignore the sub-spans of entities (in this case, the family name). To solve this problem, we first segment the given question into a set of string spans with different sizes and we then compare them with an entity name (type.object.name) as well as sub-spans of the name and the aliases (common.topic.alias) of the *subject entity* in the entity span matching task. If entity spans are detected, they are replaced with *#entity*. The next step is KB-fact retrieval based on *subject entity* and *object entity*, which allows us to find the relations between two entities as the FREEBASE *predicate*. As a result, we obtain [[*question pattern* ||FREEBASE *predicate*]] tuple ([[*where be #entity born* || people.person.place_of_birth]], as shown in Fig. 6).

Fig. 7 illustrates another solution for multi-related questions which are mainly related to the 4-hop FREEBASE *predicate*. Thus, their

Table 3
Positive and negative cases of lexical pattern extractions with distant supervision.

WIKIPEDIA	
Relation phrase	*#entity be born*
Positive	people.person.place_of_birth
Negative	people.deceased_person.place_of_death
Question–answer pairs	
Question pattern	*which country border #entity*
Positive	location.location.adjoin_s.adjoins
Negative	location.statistical_region.places_exported_to.exported_to

answer entities (*object entities*) are likely to be found in *event entity*, which contains other related entities as well as answer entities. We then detect *conditional entities* in the question based on the above entity span matching solution. By using another place holder *#cond*, the positional information for a condition entity can also be retained. The query for FREEBASE can be transformed into $<$ *subject entity*, ?, *object entity* & *condition entity* $>$. Since there are two FREEBASE entities for the object, the two FREEBASE *predicates* for each entity are matched. Consequently, the tuple [[*question pattern* ‖FREEBASE *predicate* & *conditional*FREEBASE *predicate*]] can be extracted. Fig. 7 shows that all the KB-facts for the subject entity (Titanic) are crawled during the first FREEBASE enquiry. We then check whether other entities (except the answer entity; Leonardo Di Caprio) appear in the question. As a result, Jack Dawson is selected as the *condition entity*, and the final tuple ([[*who plays #cond in the #entity* ‖ film.film.starring.actor-&-film.film.starring.character]]) is acquired during the second FREEBASE enquiry.

5.3. FREEBASE *predicate selection with distributional semantics*

Linking task swith *weak supervision* (*distant supervision*) sometimes causes errors due to the redundancy of entity pairs across many KB-facts. In particular, when some KB-facts share identical entity pairs (*subject entity* and *object entity*), the semantic sense of the relations between them (FREEBASE *predicates*) can be quite different. As shown in Table 3, if someone's birth place and death place are the same, some of the extracted lexical patterns cannot be associated with a given FREEBASE *predicate* (upper cases). Similarly, members of an entity pair that are connected to each other across many KB-facts (e.g., country-to-country) can lead to incorrect linking tasks (lower cases). We employ labeled-LDA [28] to solve these problems because the distributional semantics between lexical patterns and FREEBASE *predicates* can distinguish semantic-associated tuples. Next, we describe the post-processing step for semantic links based on topical analysis.

Predicate filtering using labeled-LDA with FREEBASE *types.* Labeled-LDA [28] is a probabilistic graphical model that describes a process for generating a labeled document collection as a modification of LDA [5]. In contrast to the standard LDA, both the label set Λ and the Dirichlet parameter α contribute to the topic distributions for a document d (θ_d) as two directed edges from Λ and α to θ in Fig. 8. In particular, labeled-LDA models treat each document as a mixture of underlying topics to generate each word from the assigned topic, as well as integrating supervision simply by constraining the topic model to the use of those topics that correspond to a document's (observed) label set Λ.

When a *lexical pattern* (*relation phrase* or *question pattern*) is linked to multiple FREEBASE *predicates*, as shown in Table 3, we must try to select the most appropriate FREEBASE *predicate* using distributional semantic analysis. Based on the anchors of WIKIPEDIA articles that correspond to FREEBASE entities, we can assign the available FREEBASE types to each WIKIPEDIA article. Because we use FREEBASE types as the label set for labeled-LDA, the word probability distributions of each FREEBASE type (label) can then be obtained. As shown in Fig. 9, the three entities, Steve Jobs, Bill Gates, and Barack Obama, are popular people, but their assigned FREEBASE types are somewhat different, such as their careers, e.g., Steve Jobs and Bill Gates: founder and Barack Obama: politician. In particular, Steve Jobs is a deceased person, so the terms for deceased_person type are used frequently in the WIKIPEDIA paragraphs of Steve Jobs. For this reason, we apply labeled-LDA to the WIKIPEDIA paragraphs, each of which is labeled with the appropriate FREEBASE type, i.e., for each WIKIPEDIA article (FREEBASE entity), the labeled FREEBASE types used in the label set Λ are determined by all the facts of the entity. Table 4 shows the result obtained after using labeled-LDA to capture the distributional semantics between FREEBASE types and the terms used in WIKIPEDIA paragraphs.

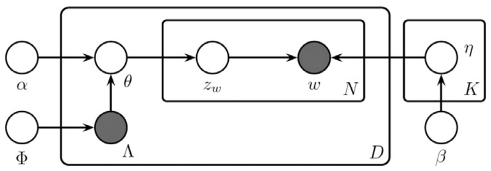


Fig. 8. Plate notation for labeled-LDA: the Dirichlet parameter α and a label set Λ influence the topic distribution (θ) of document D, which affects the topic for word w (z_w). In addition, the word (w) depends on the word distribution (η) according to another Dirichlet parameter β.

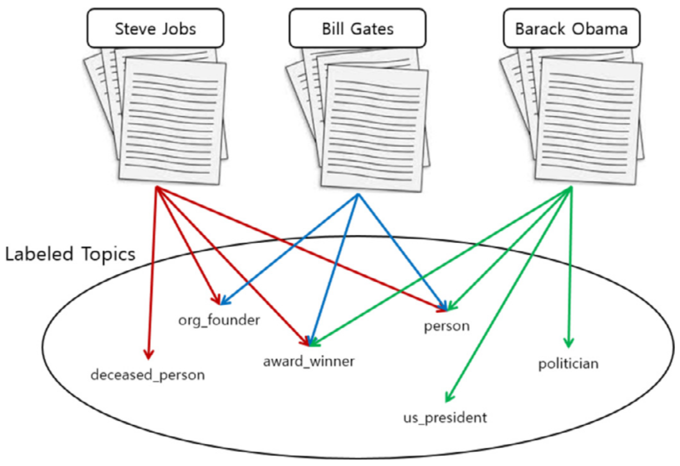


Fig. 9. Set of labeled topics (FREEBASE types) linked with WIKIPEDIA articles (FREEBASE entities): Steve Jobs, Bill Gates, and Barack Obama. The directed arrows indicate the labeled FREEBASE types of the entities.

Table 4
Examples of FREEBASE types and their top-scoring terms in labeled-LDA based on WIKIPEDIA paragraphs.

FREEBASE type	High probability terms
people.person	Work, serve, member, school, include, study, born, ...
people.deceased_person	Die, son, death, family, time, father, daughter, return, ...
sports.sport	Team, play, game, player, sport, use, event, ball, time, ...
book.author	Book, write, work, publish, novel, include, writer, ...
music.artist	Music, song, perform, band, play, album, record, ...
location.location	Area, village, build, located, include, building, use, ...
film.film	Film, direct, release, star, make, movie, write, story, ...

Algorithm 1 Predicate selection based on the FREEBASE type-to-word distribution (labeled-LDA).

```
Input: lexical pattern (lex), set of predicates linked with lex ($P_{lex}$),
    probability distribution of FREEBASE types and words ($Z$; $P(t|w)$)
1: for all $p_i \in P_{lex}$ do
2:     Get a FREEBASE type $t_i = pre2type(p_i)$;
3:     $Score\ s_i \leftarrow 0\ (TopicSim(p_i) = s_i)$;
4:     for all word $w_j \in W_{lex}$ do
5:         $s_i \leftarrow s_i + P(t_i|w_j)$;
6:     end for
7: end for
Output: predicate $\hat{p} = \arg\max_{p_i \in P_{lex}} TopicSim(p_i)$
```

Algorithm 1 describes the predicate filtering process when a lexical pattern *lex* is linked with many predicates P_{lex}. Since a FREEBASE type corresponds to the sub-hop of FREEBASE *predicates*, we first bind the given predicate p_i to FREEBASE type t_i using the dictionary *pre2type* (Line 2). Next, we measure the score s_i for p_i by summing the probabilities of t_i given word w_j, i.e., in W_{lex} ($P(t_i|w_j)$), where W_{lex} is the set of words that occurs in the lexical pattern *lex* (Lines 4 and 5). Finally, we select the predicate assigned the highest score $\hat{p}$, i.e., the most appropriate predicate for the given lexical pattern based on the semantic analysis.

In Table 3, for the upper case, people.person.place_of_birth is selected because the word "*born*" is highly associated with the predicate, as shown in Table 4. For the lower case, the words "*which*" and "*border*" are assigned with high probabilities to location.location.adjoin_s.adjoins.

5.4. Semantic linking model

At this point, we have examined how to extract and select tuples of [[*lexical patterns* ||FREEBASE *predicates*]] from the unstructured data. Table 5 shows some examples of tuples obtained by the extraction methods described in Section 5.1 (WIKIPEDIA) and Section 5.2 (QA-pairs). The *lexical patterns* and FREEBASE *predicates* are highly semantically associated and they treat various semantic relations. *Relation phrases* generally have simple lexical representations, whereas *question patterns* are usually related to the structure types of common questions.

Next, we introduce training pairs for joint embedding learning and a feature-level scoring model to filter out the low quality semantic links.

Feature representation for semantic links. Given a tuple of [[*lexical pattern* || FREEBASE *predicate*]], a lexical pattern *lex* (*relation phrase* or *question pattern*) is divided into several *n*-grams. First, we designate a placeholder *#out* at the boundary of *lex* and we then extract uni-, bi-, and trigrams from the word sequence of *lex*. W_{lex}lex denotes the set of *n*-grams (w_{lex}) that occur in *lex*. On the opposite side, the question category c_{lex} for a pattern *lex* is assigned (only from QA-pairs), and a FREEBASE predicate *pre* is converted into a topic domain d_{pre}, an entity type t_{pre}, and a logical predicate p_{pre} by dividing the three-level hierarchical structure of the *pre*. In addition to the extraction method, the question category is obtained based on a simple rule-based approach. For example, if a question starts with *"how many"* or its answer type is a numeric value, we label the question category as count. The topic domain, entity type, and logical predicate can be obtained directly from a FREEBASE *predicate*, which contains the FREEBASE domain (the beginning-hop), FREEBASE type (sub-hop from the beginning), and FREEBASE predicate (full-hop). Table 6 shows an example of the feature representation task for a tuple of [[*lexical pattern* || FREEBASE *predicate*]]. Finally, we can generate the feature-level semantic links of each element in a set W_{lex} with each logical feature (c_{pre}, d_{pre}, t_{pre}, or p_{pre}), and thus a tuple can build multiple feature pairs (< *lexical feature, logical feature* >). These pairs are designed as training pairs for joint relational embeddings, as described in Section 6.

Based on the features extracted above, the expected effects across their pair types are defined as follows.

- < **Word, question category** > **pairs** ($\mathcal{WC}$) denote the semantic associations between an *n*-gram (w) and a question category (c). These relations are used mainly for answer type prediction-to-question statements, so they have a central role in the answer formulation step in KB-QA (Section 7.3). We expect that interrogative *n*-grams (e.g., "*how many*" or "*when*") will be quite well related to the question categories.
- < **Word, topic domain** > **pairs** ($\mathcal{WD}$) state the semantic associations between an *n*-gram (w) and a FREEBASE domain (d). With

Table 5
Statistics and samples of tuples obtained from WIKIPEDIA and QA-pairs.

WIKIPEDIA	
# Sentences (≥ two entities): 32.8 M	# Tuples: 1,949,043
[[*#entity graduate from* || people.person.education.institution]]	
[[*#entity be town in* || location.location.containedby]]	
[[*#entity be son of* || people.person.parents]]	
[[*be son of #entity* || people.person.children]]	
Question–answer pairs	
# QA-pairs: 4327	# Tuples: 4414
[[*what to do in #entity with family* || travel.travel_destination.tourist_attractions]]	
[[*what language they speak in #entity* || location.country.languages_spoken]]	
[[*who be #cond of #entity* || governing.office_holder & governing.basic_title]]	

Table 6
< Lexical feature, Logical feature > relations obtained from the given tuple.

[[*where be #entity born* || people.person.place_of_birth]]	
$\mathcal{UC}$	< *1g_where*, list >, . . . , < *3g_#entity born #out*, list >
$\mathcal{UD}$	< *1g_where*, people >, . . . , < *3g_#entity born #out*, people >
$\mathcal{UT}$	< *1g_where*, people.person >, . . . , < *3g_#entity born #out*, people.person >
$\mathcal{UP}$	< *1g_where*, people.person.place_of_birth >, ... < *3g_#entity born #out*, people.person.place_of_birth >

respect to the aim of a question statement, we assume that the statement covers only one topic, and thus these relations are the initial point for understanding the statement. We also expect that the broad semantics at the top-level of a hierarchical structure can help to design a robust KB by recovering broken and hidden relations in the lower levels based on the shallow semantics.

- $<$ **Word, entity type** $>$ **pairs** ($\mathcal{WT}$) indicate the semantic associations between an n-gram (w) and a FREEBASE type (t) of an entity (e) in question statements, which are important for the entity span selection step (Section 7.1). These relations allow us to analyze the subject entity's attributes, such as its positional information and relevant entity types for the given context, which may address the entity disambiguation problem in KB-QA. As placeholders for entities, the *#entity* and *#cond* symbols in n-grams allow us to analyze their neighboring context.
- $<$ **Word, logical predicate** $>$ **pairs** ($\mathcal{WP}$) represent the semantic associations between an n-gram (w) and a FREEBASE predicate (p). They are suitable for the predicate mapping step (Section 7.2), which aims to map question statements onto their corresponding predicates. Similar to typical semantic parsers, these relations can exploit the semantic overlap between the question contexts and logical predicates of the KB. Finding discriminative n-gram features to link logical predicates, such as interrogative words and main verbs, is crucial for our method. Moreover, with respect to the coverage issues in the QA task, more relations provide greater insights to help understand the broad question context.

Scoring model for $<$ lexical feature, logical feature $>$ links. By obtaining high quality $<$ lexical feature, logical feature $>$ pairs, we can retain only the lexical features that are highly associated with logical features. To determine the correlations among lexical and logical features, we calculate the score s_l for a link l based on PMI computation [10], as follows:

$$\begin{aligned} s_l &= \alpha \cdot \mathrm{PMI}(\mathcal{E}_{f\mathrm{lex}}; \mathcal{E}_{f\mathrm{log}}) \\ &\quad + \beta \cdot \mathrm{PMI}(\mathcal{U}_{f\mathrm{lex}}; \mathcal{U}_{f\mathrm{log}}) \\ &\quad + \gamma \cdot \mathrm{PMI}(\mathcal{P}_{f\mathrm{lex}}; \mathcal{P}_{f\mathrm{log}}), \end{aligned} \tag{1}$$

where $\mathcal{E}$ and $\mathcal{U}$ denote the total set and the unique set of entity pairs, respectively, and $\mathcal{P}$ represents a set of tuples of [[*question pattern* || FREEBASE *predicate*]]. More precisely, $\mathcal{E}$ is the set of all pairs of $<$ *subject entity, object entity* $>$ that participate in *relation phrases* or *logical predicates*, which are extracted as described in Section 5.1, and the $\mathcal{U}$ means $<$ *subject entity, object entity* $>$ pairs after removing the duplicated pairs in $\mathcal{E}$. The unique set is expected to complement the total set by reducing the amount of redundant information. We aggregate the PMI scores between $\mathcal{E}_{f\mathrm{lex}}$ and $\mathcal{E}_{f\mathrm{log}}$, and $\mathcal{U}_{f\mathrm{lex}}$ and $\mathcal{U}_{f\mathrm{log}}$ are controlled by parameters α and β respectively, where $\mathcal{E}_f$ indicates the total set of entity pairs that co-occur with a feature *f*, and $\mathcal{U}_f$ indicates the set of entity pairs, which are the repeated pairs removed in $\mathcal{E}_f$, as described above. The PMI score for the tuples $\mathcal{P}$ is also combined with two other PMI scores, which are constrained by a parameter γ. Similar to $\mathcal{E}_f$ and $\mathcal{U}_f$, $\mathcal{P}_f$ indicates the total set of tuples that occur with a feature *f*. The set *S*, which is assigned scores to any links in the set *L*, is finally obtained by Eq. (1), where the parameters α, β, and γ are selected empirically.

For the implementation, Eq. (1) is transformed into the following equation in terms of PMI computation with the frequency-based probabilities:

$$\begin{aligned} s_l &= \alpha \cdot \log \frac{|\mathcal{E}_{f\mathrm{lex}} \bigcap \mathcal{E}_{f\mathrm{log}}| \cdot |\mathcal{E}|}{|\mathcal{E}_{f\mathrm{lex}}| \cdot |\mathcal{E}_{f\mathrm{log}}|} \\ &\quad + \beta \cdot \log \frac{|\mathcal{U}_{f\mathrm{lex}} \bigcap \mathcal{U}_{f\mathrm{log}}| \cdot |\mathcal{U}|}{|\mathcal{U}_{f\mathrm{lex}}| \cdot |\mathcal{U}_{f\mathrm{log}}|} \\ &\quad + \gamma \cdot \log \frac{|\mathcal{P}_{f\mathrm{lex}} \bigcap \mathcal{P}_{f\mathrm{log}}| \cdot |\mathcal{P}|}{|\mathcal{P}_{f\mathrm{lex}}| \cdot |\mathcal{P}_{f\mathrm{log}}|}, \end{aligned} \tag{2}$$

where $\mathrm{PMI}(x; y) = \log \frac{P(x,y)}{P(x)P(y)}$ can be re-written as $\mathrm{PMI}(x; y) = \log \frac{|x \bigcap y| \cdot C}{|x| \cdot |y|}$. This function is derived partly from the *support* score [16], but we focus on the correlations of shared entity pairs in terms of lexical features and logical features using the PMI computation.

In contrast to the frequency-based scoring method, our correlation-based scoring function tends to prevent biased scoring. For example, a unigram *"born"* co-occurs with many logical features in the person.birth class, and thus it may be biased toward the person.birth class if a frequency-based method is used. However, by considering information for shared entities and by reducing the redundancy of the counts of the entities, the *"born"* feature may have an opportunity to be associated with other logical features. In fact, *"born"* is related to the person.birth class as well as organization.found, artist.origin, etc.

Finally, for each semantic link of the lexical features (uni-, bi-, and trigrams) and logical features (question category, entity type, and logical predicate), we normalize the scores *S* to allow effective relational learning. Fundamentally, a score s_l for a link *l* of $<$ lexical feature, logical feature $>$ is utilized to filter out the low quality links and to assign its margin score in the training step, as described in Section 6.3.

6. Semantic embedding space construction

In this section, we describe how to learn the joint semantic embeddings using the semantic links of $<$ lexical feature, logical feature $>$ with their scores (Section 5). Similar to the typical semantic parser, the goal is to learn the semantic associations between lexical features and logical features. These associations are crucial for obtaining a logical representation of the KB that is semantically relevant to the given question statement.

First, we describe how to represent features in the embedding space $\mathbb{E}$ in Section 6.1, and we then introduce a similarity measure between embedding vectors in Section 6.2. Section 6.3 describes the learning step for the embedding space based on the semantic-associated links of $<$ lexical feature, logical feature $>$ (L) and their scores (S), which are obtained as explained in Section 5.

6.1. Feature representation in the embedding space

For our feature space, each lexical (w) and logical (t, e, or p) feature is encoded in the distributed representation, which comprises n-dimensional embedding vectors $\mathbb{E}^n$:

$$\forall x, \quad x \overset{\text{encode}}{\Rightarrow} \mathbb{E}(x) \in \mathbb{E}^n. \tag{3}$$

For the implementation, a lookup table manages the connections between features and their aligned embedding vectors. This feature representation approach is inspired by a previous study of embedding-based relation extraction [35], but it differs in the following respects. (1) In our method, a pattern-level tuple of the same form as that considered by Weston et al. [35] is divided into each lexical feature, which is paired with a logical feature. We consider that semantic links of the form $<$ lexical feature, logical feature $>$ can avoid redundant links and facilitate more effective relation learning. (2) Entity information (d and t) is represented by the separate embedding vectors but its positional information remains as the symbol *#entity*.

The numeric vector (embedding) that encodes a feature represents the feature's semantic information because each numeric element in the vector is specified as the weight in some semantic sense.

6.2. Semantic similarity computation via embedding

In our joint relational approach, we focus on the set of paired relationships $<$ lexical feature, logical feature $>$, which can be utilized semantically. Formally, these features are embedded in the same latent space $\mathbb{E}^n$ and their semantic similarities can be computed by a dot product operation:

$$\mathrm{Sim}(l) = \mathrm{Sim}(a, b) = \mathbb{E}(a)^{\mathsf{T}}\mathbb{E}(b), \tag{4}$$

where l is a link of $<$ a, b $>$, and a or b are the features described in Section 4.2. In mathematics, this function is the same as the matrix operation between $1 \times k$ and $k \times 1$ matrices. We expect that the distance between the vectors of two features will be closer if the two features are more strongly semantically associated. Thus, closer numeric values for the same indexes in the embeddings can produce higher semantic similarity scores.

6.3. Ranking-based joint relational embedding learning

Our ranking-based relational learning approach is based on a ranking loss function [33,35], which supports the idea that the similarity scores of observed pairs in the training set (positive instances) should be higher than those of any other pairs (negative instances), i.e.,

$$\forall l^{\mathrm{pos}} \in L,\ \forall l^{\mathrm{neg}} \notin L, \quad \mathrm{Sim}(l^{\mathrm{pos}}) > m + \mathrm{Sim}(l^{\mathrm{neg}}), \tag{5}$$

where a set L denotes the training set of semantic links, and l^{pos} and l^{neg} indicate an observed link (positive relation) and an unobserved link (negative relation) in the set L, respectively. We assume that the set L is the group of gold standard links, and thus the unseen links are regarded as incorrect links. In addition, a margin score m is utilized to denote the significance (weight) of the given positive link l^{pos}. In summary, the rank (similarity score) of a correct link should be higher than that of an incorrect link as much as the weight on the correct link (margin score).

A reasonable equation for selecting negative links is as follows:

$$\begin{aligned} \forall l_i \in L, \quad & \forall l_i^{\mathrm{neg}} \notin L, \\ & \mathrm{Sim}(l_i) > m_i + \mathrm{Sim}(l_i^{\mathrm{neg}}) \\ & \mathrm{Sim}(f_i^{\mathrm{lex}}, f_i^{\mathrm{log}}) > m_i + \mathrm{Sim}(f_i^{\mathrm{lex}}, f_{\mathrm{neg}}^{\mathrm{log}}), \end{aligned} \tag{6}$$

where $l_i^{\mathrm{neg}} =< f_i^{\mathrm{lex}}, f_{\mathrm{neg}}^{\mathrm{log}} >$ denotes a negative link with respect to a link l_i (positive link), and f^{lex} and f^{log} indicate a lexical feature and a logical feature, respectively. The lexical feature of a link l_i^{neg} is equal to that of the positive link l_i (f_i^{lex}), whereas its logical feature $f_{\mathrm{neg}}^{\mathrm{log}}$ is determined by one that is unrelated to f_i^{lex} in the set L. The selection of negative links in this manner can be more efficient than Eq. (5) because the constraint that only considers logical features in negative terms by excluding lexical features helps to speed up the convergence of embedding during training. As a consequence, the similarity scores of correct links should be higher than those of links where their logical features are only altered to those in the negative set due to the weights on the correct links.

In practice, our embedding vectors are trained under the soft ranking criterion on links of $<$ lexical feature, logical feature $>$, which conducts stochastic gradient descent (SGD). Thus, we aim to minimize the following:

$$\begin{aligned} \forall l_i \in L, \quad & \forall l_i^{\mathrm{neg}} \notin L, \\ & \max\left(0,\ m_i - \mathrm{Sim}(l_i) + \mathrm{Sim}(l_i^{\mathrm{neg}})\right) \\ & \max\left(0,\ m_i - \mathbb{E}(f_i^{\mathrm{lex}})^{\mathsf{T}}\mathbb{E}(f_i^{\mathrm{log}}) + \mathbb{E}(f_i^{\mathrm{lex}})^{\mathsf{T}}\mathbb{E}(f_{\mathrm{neg}}^{\mathrm{log}})\right), \end{aligned} \tag{7}$$

where l_i^{neg} represents a negative link of $< f_i^{\mathrm{lex}}, f_{\mathrm{neg}}^{\mathrm{log}} >$. SGD is a gradient descent optimization method for minimizing an objective function, which can reduce the computational cost of each iteration. We set -1 as the minimum score for SGD, and thus the element values in the embedding vectors can be represented by negative numbers. We expect that the use of negative numbers in vectors will allow embedding to indicate various semantic meanings.

Algorithm 2 describes our learning strategy. First, we initialize the embedding space $\mathbb{E}^n$, n-dimensional vector, by randomly allocating a mean of 0 and a standard deviation $1/n$ to each vector (Line 1). Next, for each link of $l_i =< f_i^{\mathrm{lex}}, f_i^{\mathrm{log}} >$ in the training set L, we select a negative link l_{neg} where the lexical feature f_i^{lex} is retained as that of l_i, but the logical feature ($f_{\mathrm{neg}}^{\mathrm{log}}$) is selected based on another that is unrelated to f_i^{lex} in the set L, according to Eq. (6) (Line 6). The right-hand part is the SGD constraint on the ranking criterion: $\mathrm{Sim}(l_i) > m_i + \mathrm{Sim}(l_i^{\mathrm{neg}})$. Finally, we run a stochastic gradient step to minimize Eq. (7) and update $\mathbb{E}^n$ at each step (Line 8).

Algorithm 2 Ranking-based learning process for joint semantic embeddings.

```
Input: set of training relations ($L$) and set of their scores ($S$), parameters (e.g., $MaxItr$, $\lambda$, $n$)
 1: Initialize embedding space $\mathbb{E}^n$
 2: for $itr := 1$ to $MaxItr$ do
 3:     for all $r_i \in R$ do
 4:         semantic link $l_i =< f_i^{lex}, f_i^{log} >$, its score $s_{l_i}$;
 5:         Run a stochastic gradient step to minimize:
 6:         $\forall l_i^{neg} \notin L,\ \max(0,\ m_i - \mathrm{Sim}(l_i) + \mathrm{Sim}(l_i^{neg}))$;
 7:                         ▷ $l_i^{neg} =< f_i^{lex}, f_{neg}^{log} >$ and $m_i = s_{l_i}$
 8:         Update $\mathbb{E}(f_i^{lex})$, $\mathbb{E}(f_i^{log})$, and $\mathbb{E}(f_{neg}^{log})$;
 9:     end for
10: end for
Output: embedding space $(\mathbb{E}^n)^{MaxItr}$
```

In particular, if the ranking criterion for given positive and negative pairs is false ($m_i + \mathrm{Sim}(l_i^{\mathrm{neg}}) > \mathrm{Sim}(l_i)$), then the embeddings of the semantic links l_i and l_i^{neg} ($\mathbb{E}(f_i^{\mathrm{lex}})$, $\mathbb{E}(f_i^{\mathrm{log}})$, and $\mathbb{E}(f_{\mathrm{neg}}^{\mathrm{log}})$) are learnt incorrectly. Thus, we perform back-propagation for two links (three embedding vectors) in order to satisfy the ranking criterion by increasing the similarity score of the positive pair ($\mathrm{Sim}(l_i)$) and by decreasing the similarity score of the negative pair ($\mathrm{Sim}(l_i^{\mathrm{neg}})$) as much as the learning rate (λ). Next, the revised embeddings are updated continually.

We expect that joint relational learning of the semantic associations between lexical features and logical features will have the following advantages. (1) The embeddings of logical features can discover hidden and incomplete relations as well as the observed relations between the properties of the KB, which comprises hierarchical and multi-relational data. The robust structure of the embeddings in the KB is suitable for mapping any lexical features. (2) Although FREEBASE is incomplete in terms of particular semantics, our joint embeddings can bind the narrow semantics into broader semantics. For example, a predicate user.alexander.misc.murdered_person.murdered_by is made by an ordinary user *alexander* as individual contributions of FREEBASE, which can used to understand specific questions even if it is not the base predicate. In this case, FREEBASE does not provide appropriate FREEBASE types for the predicate because only the base predicates are officially approved by FREEBASE. There are similar problems with inversed predicates for some base predicates in FREEBASE where the exclamation mark indicates the opposite meaning, such as people.person.nationality and !people.person.nationality. In fact, the second predicate should be linked with the type location.country. Hence, our embeddings should link unapproved predicates (user.alexander.misc.murdered_person.murdered_by) to semantically associated FREEBASE types (people.person) because invisible types (user.alexander.misc. murdered_person) co-occur with the FREEBASE types (people.person) in training pairs. (3) Our joint relational learning approach can smooth the surface (lexical) features for semantic parsing by using the aligned logical features. In other words, semantically similar lexical features can be clustered

Table 7
Semantically similar logical features for the given lexical features based on learnt joint embeddings.

1w_born	
c	list / count
d	people / location / zoos / dataworld / common ...
t	people.person / fictional_universe.fictional_setting / zoos.zoo_animal ...
p	people.person.places_lived.location / people.person.date_of_birth ...
2w_be born	
c	list / count
d	people / location / zoos / fictional_universe / dataworld ...
t	people.person / fictional_universe.fictional_setting / zoos.zoo_animal ...
p	people.person.date_of_birth / people.person.places_lived.location ...
3w_be born in	
c	list / count
d	people / location / fictional_universe / biology / common ...
t	people.person / fictional_universe.fictional_setting / location.location ...
p	location.location.people_born_here / biology.organism.place_of_birth ...

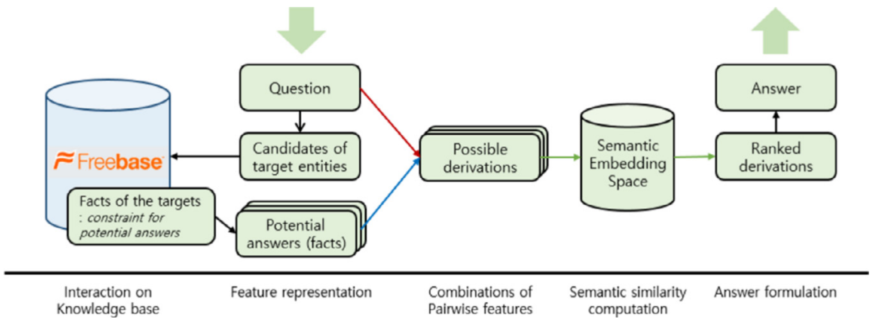

Question
Candidates of target entities
Freebase
Facts of the targets : constraint for potential answers
Potential answers (facts)
Possible derivations
Semantic Embedding Space
Ranked derivations
Answer
Interaction on Knowledge base
Feature representation
Combinations of Pairwise features
Semantic similarity computation
Answer formulation

Fig. 10. Sequential stages of our question answering approach based on the semantic embedding space.

if they are connected by only one logical feature that denotes one semantic concept.

Table 7 shows the top ranked logical features for the given lexical features in the trained embedding vectors ($\mathbb{E}^n$). The unigram "*1w_born*" and the bigram "*2w_be born*" are related to the broad semantics of the birth domain. However, the trigram "*3w_be born in*" shows the more specific (narrow) semantic, which is the location of the birth domain. In particular, the highly associated semantics with "*1w_born*" and "*2w_be born*" are more similar to that of "*3w_be born at*", the time information of birth (people.person.date_of_birth), rather than that of "*3w_be born in*".

7. Knowledge-based question answering with semantic embeddings

Our algorithm for finding the answers for a given question statement using the semantic embedding space. The learnt embeddings obtained as described in Section 6.3 are used to map the lexical features of the question statement onto the logical representation of the correct answer in the KB.

The QA stage is as follows. Given a question statement q, we first identify the candidate entities that occur in the statement q, and we generate all possible derivations K_q based on the lexical features of the statement q and the logical features of KB-facts that are linked with the candidate entities, as described in Section 7.1. We rank the derivations via joint semantic embeddings $\mathbb{E}^n$, as described in Section 7.2, to predict the answer derivation $\hat{k}$. Finally, as described in Section 7.3, we formulate the final answer statement ans_q using the answer knowledge $\hat{A}_q$ guided by the predicted answer derivation $\hat{k}$. These stages are shown in Fig. 10.

FREEBASE *API interaction.* Before examining QA tasks using FREEBASE APIs, we first introduce the following two functions for the FREEBASE interactions.

Search-in- $\mathcal{KB}(\cdot)$ executes entity retrieval in FREEBASE for a given string span via Search API.[5] For the names and aliases of entities, we can obtain appropriate FREEBASE entities without preparing our own database. We select returning properties using the filter constraint for Search API. In our method, we use the Search API function to identify candidate entities in the question statement.

Get-Props- $\mathcal{KB}(\cdot)$ provides retrieved facts in the triple format, $<$ *subject entity, logical predicate, object entity* $>$, in FREEBASE for a given input property via Topic API.[6] Hence, this function can be placed in two types of input properties: $<$ *subject entity,* $*, * >$ and $<$ *subject entity, logical predicate,* $* >$, where $*$ indicates a *wildcard* that can substitute for any of the properties in the KB. Thus, we obtain complete KB-facts based on the input constraints of the knowledge required by the user. For a question statement, a constraint with the form $<$ *subject entity, logical predicate,* $* >$ is equivalent to its structured query for the KB, which yields the answer entities directly. In our study, the Topic API function is used to retrieve KB-facts for candidate entities using the constraint: $<$ *candidate entity,* $*, * >$.

7.1. Candidate answer derivations

We generate all possible answer derivations based on a given question statement and a candidate answer. In this case, a candidate answer derivation represents combinations of lexical features

[5] https://developers.google.com/freebase/v1/search-overview.
[6] https://developers.google.com/freebase/v1/topic-overview.

and logical features. The lexical features are extracted from the question statement and the logical features are inferred from the KB-facts of candidate entities, which can be identified among the string spans of the question statement. Based on the learnt embedding model described in Section 6, we can score any pairs of lexical features and logical features, and thus our method focuses on the similarity value between the pairwise features of individual derivations. We consider that summing the pairwise similarities with their weights helps to utilize the semantic associations between the given question statement and the potential answers.

Algorithm 3 describes the generation step for candidate answer derivations. Given a question statement q, we first find all of the possible entity spans E_q that appear in the statement q using the Search API, the *Search-in-$\mathcal{KB}$* function, based on FREEBASE, which obtains a mapping from the string spans of the question to FREEBASE entities. In FREEBASE, entities are recovered based on their unique identifications, i.e., the mid properties of FREEBASE. Specifically, we divide the statement into a set S_q (all of the string spans that appear in q) by collecting string spans with lengths that vary by one from that of the statement, and we then retrieve the candidate entities (E_q) by comparing each span s with the FREEBASE names and aliases of the entities (Line 3). When a FREEBASE entity matches with multiple spans, the longest span is selected. Similarly, if a string span matches with multiple FREEBASE entities, the most popular ones are selected. The popularity of entities is determined by the relevance score provided by FREEBASE.[7] Next, we extract the lexical features and logical features, as described in Section 5.4. A word sequence of the question statement is split into uni- bi-, and trigrams, and the placeholders, (*#entity*, *#cond*, and *#out*), which are recorded according to the identified entity e. W_e denotes a set of words (n-grams) for the question statement q where the subject entity is e. On the opposite side, candidate question categories are assigned based on all of the items in their set C. Furthermore, a FREEBASE entity is linked with a large number of predicates, which can generate many feasible groups of FREEBASE logical features (d, t, and p), depending on the identified entity e via Topic API, the *Get-Props-$\mathcal{KB}$* function: $< e, *, * >$, of FREEBASE (Line 6). We also obtain candidate answer knowledge ($\mathcal{A}$) from the retrieved object entities. As a result, a candidate answer derivation $k = (A; W_e, c, d, t, p)$ comprises a possible combination of lexical features and logical features for inferring a potential answer A (Lines 8 and 9). Note that a candidate answer A is selected based on a subject entity (e; candidate entity) and a logical predicate p (Line 10).

Algorithm 3 Generation of candidate answer derivations.

```
Input: question statement (q), set of question categories (C), set of
    topic domains (D), set of entity types (T), set of predicates (P),
    knowledge base ($\mathcal{KB}$)
 1: $K_q$: set of logical derivations for question statement q
 2: for all string span $s \in S_q$ do
 3:     $E_q \leftarrow E_q + Search\text{-}in\text{-}\mathcal{KB}(s)$; // $E_q$: candidate entities
 4: end for
 5: for all $e_i \in E_q$ do
 6:     $\mathcal{A}_{e_i}, D_{e_i}, T_{e_i}, P_{e_i} \leftarrow Get\text{-}Props\text{-}\mathcal{KB}(< e_i, *, * >)$; // $\mathcal{A}$: candidate
    answers
 7:     $E_q, P_{e_i} \leftarrow$ Add $multi\text{-}pre(P_{e_i}, E_q, e_i)$;
 8:     for all $A_j \in \mathcal{A}_{e_i}, c_j \in C, d_j \in D_{e_i}, t_j \in T_{e_i}, p_j \in P_{e_i}$ do
 9:         $K_q \leftarrow K_q +$ derivation $k_j = (A_j; W_{e_i}, c_j, d_j, t_j, p_j)$;
10:                     ▷ constrained by a KB-fact: $< e_i, p_j, A_j >$
11:     end for
12: end for
Output: $K_q$
```

Multi-related question. As mentioned in Section 5.2, some multi-related questions include a *condition entity* as well as a main entity. Both entities are crucial for understanding the question. Since the common derivations only include the main entities, the additional step (*multi-pre*) is required to insert a derivation that holds a *condition entity* and a *conditional logical predicate*, as described in Line 7 of Algorithm 3.

The *multi-pre* function is explained fully in Algorithm 4. For each predicate p in the set P_e, which contains predicates for an entity e, we first match it with a set P^{multi} that includes a predefined list of multi-related predicates (p-&-p') obtained by question pattern extraction, as described in Section 5.2 (Line 4). After executing *Get-Props-$\mathcal{KB}$*$(< e, p', * >)$, where p' is a *conditional predicate*, its object entities can be obtained (Line 5). Next, we will obtain the multi-related entity and predicate if the retrieved object entities occur in the question statement q (from Line 5 to Line 7). The multi-related entity and predicate comprise the main properties, which are concatenated with *condition entity* and *conditional predicate*, where both are then divided by the symbol &. Our solution allows a multi-related question to be transformed into a single-related question, which can yield the answer directly. Thus, we treat the result of the merging step as one of the candidate derivations in the decoding step.

Algorithm 4 Multi-related predicate detection for the expansion of K_q (*multi-pre*).

```
Input: set of predicates for e ($P_e$), all existing entities in q ($E_q$), set of
    multi-related predicates p-&-p' ($P^{multi}$), knowledge base ($\mathcal{KB}$)
 1: $E_q^{multi}$: set of multi-related entities in the question statement q
 2: $P_e^{multi}$: set of multi-related predicates for an entity e
 3: for all $p \in P_e$ do
 4:     if $\exists p \in P^{multi}$ then
 5:         if $\exists e' \in \{E_q \cap Get\text{-}Props\text{-}\mathcal{KB}(< e, p', * >)\}$ then
 6:             $E_q^{multi} \leftarrow E_q^{multi}$ + a multi-related entity e-&-e';
 7:             $P_e^{multi} \leftarrow P_e^{multi}$ + a multi-related predicate p-&-p';
 8:         end if
 9:     end if
10: end for
Output: $E_q^{multi}, P_e^{multi}$
```

7.2. Predication for correct answer derivation

Next, we explain how to rank the candidate answer derivations K_q using the learnt semantic embeddings of lexical features and logical features. The pairwise embeddings can yield a similarity value between the features and thus the final score of the derivation can be obtained.

The following prediction model finds the best derivation $\hat{k}$ and we explain the process used to estimate the weights θ for the pairwise feature vector $\Phi(\cdot)$.

Ranking model for derivations. Our method focuses on the semantic associations between natural language statements and KBs based on the semantic similarities computed in the semantic embedding space. Therefore, the final score of a candidate answer derivation k is computed based on the summed similarity values of pairwise features in k over the weight vector θ. The pairwise features used in our method are described in Table 8. The lexical features (W) are separated into uni-, bi-, and trigrams (W^u, W^b, and W^t) because we expect that the larger n-grams will mainly give us stronger contributions. In addition, there are four types of logical features (C, D, T, and P). We consider that these 12 pairs (3×4) are suitable for utilizing the semantic associations between natural language statements and the properties of KBs.

[7] https://developers.google.com/freebase/v1/search.

Table 8
Set of pairwise features used in our method.

	Weight	Description
$\mathcal{W}^u\mathcal{C}$	θ_1	Unigram words W^u and a question category c
$\mathcal{W}^b\mathcal{C}$	θ_2	Bigram words W^b and a question category c
$\mathcal{W}^t\mathcal{C}$	θ_3	Trigram words W^t and a question category c
$\mathcal{W}^u\mathcal{D}$	θ_4	Unigram words W^u and a topic domain d
$\mathcal{W}^b\mathcal{D}$	θ_5	Bigram words W^b and a topic domain d
$\mathcal{W}^t\mathcal{D}$	θ_6	Trigram words W^t and a topic domain d
$\mathcal{W}^u\mathcal{T}$	θ_7	Unigram words W^u and an entity type t
$\mathcal{W}^b\mathcal{T}$	θ_8	Bigram words W^b and an entity type t
$\mathcal{W}^t\mathcal{T}$	θ_9	Trigram words W^t and an entity type t
$\mathcal{W}^u\mathcal{P}$	θ_{10}	Unigram words W^u and a logical predicate p
$\mathcal{W}^b\mathcal{P}$	θ_{11}	Bigram words W^b and a logical predicate p
$\mathcal{W}^t\mathcal{P}$	θ_{12}	Trigram words W^t and a logical predicate p

First, we define the score of a derivation k based on the similarities of the pairwise features in k, as follows:

$$\forall k \in K_q, \quad \begin{aligned} \mathrm{Score}(k) &= \Phi(k)\cdot\theta \\ \mathrm{Score}(k) &= \Phi(W_k, c_k, d_k, t_k, p_k)\cdot\theta, \end{aligned} \tag{8}$$

where $\Phi(k)$ is a pairwise feature vector of a derivation k, and $\Phi(\cdot)$ and θ are 12-dimensional vectors. $\Phi(k)$ is generated as follows:

$$\Phi(k) = [\,\mathrm{Sim}(W_k^u, c_k), \mathrm{Sim}(W_k^b, c_k), \ldots, \mathrm{Sim}(W_k^t, p_k)\,], \tag{9}$$

where each element in the vector is assigned based on the similarity score of each pairwise feature defined above. In the cases where W_k^u, W_k^b, and W_k^t, the unit of the features is a set of features, so we expand Eq. (4) to the following measure:

$$\mathrm{Sim}(A, B) = \frac{\sum_{a\in A, b\in B}\mathbb{E}(a)^{\mathsf{T}}\mathbb{E}(b)}{|A|\cdot|B|}, \tag{10}$$

where A and B are feature sets, and the goal is to obtain the average value of the similarity scores between the features in sets A and B. This equation is fundamentally the same as Eq. (4) if both sets A and B contain one element. Note that we assign a value of 0 to a similarity $(\mathbb{E}(a)^{\mathsf{T}}\mathbb{E}(b))$ if any of the features (a or b) do not exist in the embedding space, i.e., we penalize rare features because most rare lexical features are unnatural in terms of natural language representations.

Based on the scoring function in Eq. (8), we try to rank the candidate answer derivations in a set K_q. The best answer derivation $\hat{k}$ for the given question statement q can be obtained by:

$$\begin{aligned} \hat{k_q} &= \arg\max_{k\in K_q} \mathrm{Score}(k) \\ \hat{k_q} &= \arg\max_{k\in K_q} \Phi(k)\cdot\theta. \end{aligned} \tag{11}$$

Finally, we obtain a set of answer entities $\hat{A}_q$ from the predicted answer derivation $\hat{k} = (\hat{A}_q; \hat{W}_e, \hat{c}, \hat{d}, \hat{t}, \hat{p})$. In the following section, we formulate an answer statement using the predicted answer knowledge A_q.

Weight vector estimation. In the scoring function shown in Eq. (8), we employ a weight vector θ to adjust the contributions of the pairwise features. The following equation explains the use of the weight vector θ in detail:

$$\forall k \in K_q, \quad \begin{aligned} \mathrm{Score}(k) &= \Phi(k)\cdot\theta \\ \mathrm{Score}(k) &= \Phi_1(k)\cdot\theta_1 + \cdots + \Phi_{12}(k)\cdot\theta_{12}, \end{aligned} \tag{12}$$

where all weights should be nonzero values, and they can depict the impact of each pairwise feature.

Algorithm 5 explains the weighting process for the pairwise features. For each QA-pair (q, a) in the training set of FREE917 and WEBQUESTIONS, we can obtain the gold standard derivations K_q^{pos} and negative derivations K_q^{neg}. K_q^{pos} is generated by traversing the derivations constructed using Algorithm 3 with correct QA-pairs (q, a), whereas K_q^{neg} is built from the incorrect QA-pairs (q, a′) (Lines 4 and 5). As described in Algorithm 2, we use a ranking loss-based approach to estimate the weight vector θ because we assume that the gold standard derivations (positive instances) should be ranked higher than any other derivations (negative instances). In summary, the weight vector is trained under the soft ranking criterion based on the derivations inferred from QA-pairs, which runs SGD as described in Section 6.3. Thus, we run a stochastic gradient step to minimize the equation in Line 8 based on the learning criterion in Line 9, and we then update the weight vector θ during each step (Line 10).

Algorithm 5 Estimating weights for pairwise features using training QA-pairs.

```
Input: training set of QA-pairs ($\mathcal{QA}$), knowledge base ($\mathcal{KB}$)
 1: $\theta$: weight vector for pairwise features
 2: for itr := 1 to MaxItr do
 3:     for all (q, a) $\in \mathcal{QA}$ do
 4:         $K_q^{pos}$: set of gold standard derivations using a and $\mathcal{KB}$
 5:         $K_q^{neg}$: set of other derivations except for $K_q^{pos}$ in $K_q$
 6:         for all $k^{pos} \in K_q^{pos}$ do
 7:             Run a stochastic gradient step to minimize:
 8:             $\forall k^{neg} \in K_q^{neg}$, max (0, $1 - \Phi(k^{pos})\cdot\theta + \Phi(k^{neg})\cdot\theta$);
 9:                                  $\triangleright$ $Score(k^{pos}) > 1 + Score(k^{neg})$
10:             Update $\theta$;
11:         end for
12:     end for
13: end for
Output: $\theta$
```

Fig. 11 shows the estimated weights of the pairwise vector $\Phi(\cdot)$. Among the properties of KB, logical predicates have stronger effects when predicting the correct answer derivation. As expected, the larger n-grams ($\theta_3, \theta_6, \theta_9$, and θ_{12}) make greater contributions compared with the other smaller n-grams.

7.3. Answer formulation

We obtain a set of answer entities $\hat{A}_q$, but the answer statement ans_q provided to a user should be organized in a specific manner according to the user's requirements. Thus, we formulate ans_q using $\hat{A}_q$ with respect to the following expected answer types.

- **List type** of an answer statement ans_q provides the list of name entities in the set $\hat{A}_q$. $\mathrm{ans}_q = \{a_1, a_2, \ldots, a_n\}$ is generated by:

$$\forall a \in \hat{A}_q,\ \mathrm{ans}_q \leftarrow \mathrm{ans}_q + a.name. \tag{13}$$

- **Count type** of an answer statement ans_q counts the set $\hat{A}_q$, where the expected answer type is a number. ans_q is obtained by:

$$\mathrm{ans}_q = |\hat{A}_q|. \tag{14}$$

7.4. Question answering example

The following paragraphs describe a sequential process of answering a single-related question, "*Where is the city of david?*".

Candidate entities identification. We first identify candidate entities (E_q) of the input question q via Search API on FREEBASE. From the longest string span to uni-string spans, all possible string spans (S_q) are candidates. As the Algorithm 3, *Search-in-$\mathcal{KB}$*$(\cdot)$ function matches candidate entities based on entity names and aliases in the FREEBASE. Table 9 shows a list of pairs of a candidate entity (e) and a transformed question (W_e) where a symbol *#entity* is substituted for the candidate entity. Note that we select top-3 popular FREEBASE entities (FREEBASE IDs) for each candidate entity string.

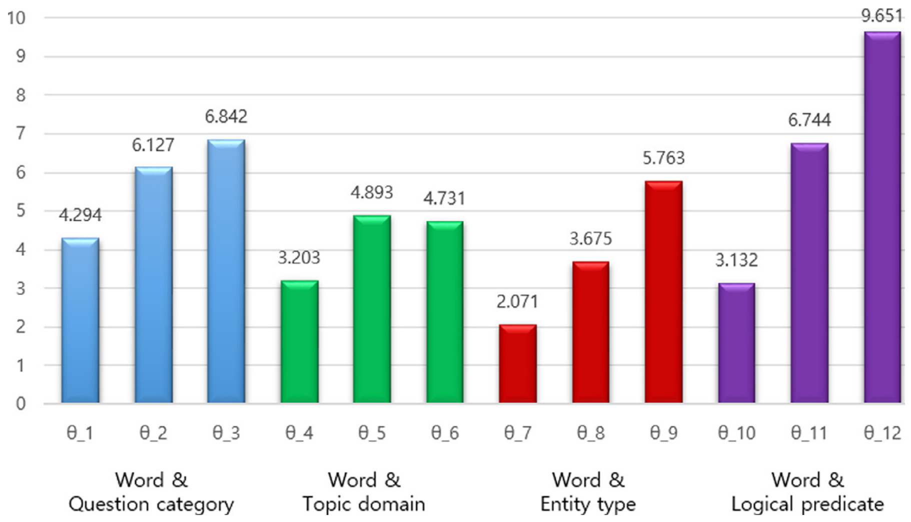


Fig. 11. Estimated weights for pairwise features.

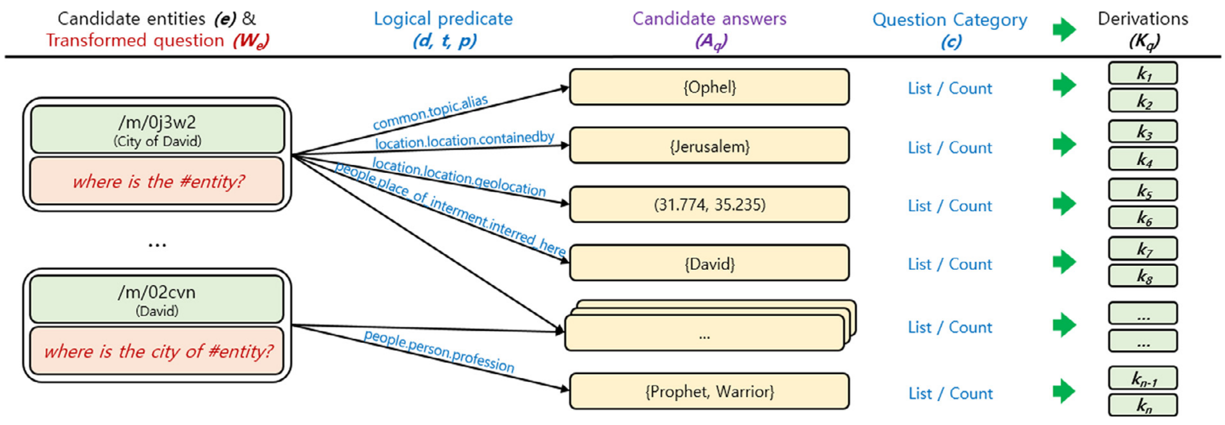


Fig. 12. Generation of answer derivations using the KB-facts of candidate entities. (For interpretation of the references to colour in the text, the reader is referred to the web version of this article.)

Table 9
Pairs of a candidate entity and a transformed question.

Candidate entity (E_q)		Transformed question (W_e)
Entity string	FREEBASE ID	
the city of david	/m/024gnn5	*where is #entity?*
	/m/0n8t6tk	
is the city	/m/0_rqlcx	*where #entity of david?*
city of david	/m/0j3w2	*where is the #entity?*
	/m/023x5yv	
	/m/0fw25ws	
the city	/m/0d6lp	*where is #entity of david??*
	/m/04myf9x	
	/m/038ydf	
city of	/m/073fqz	*where is the #entity david?*
city	/m/01n3	*where is the #entity of david?*
	/m/01h_yt	
	/m/07bvmh	
david	/m/02cvn	*where is the city of #entity?*
	/m/0953j	
	/m/01qwwx	

Generation of possible answer derivations. We crawl the KB-facts of candidate entities (E_q) via Topic API on FREEBASE. Given a FREEBASE ID, *Get-Props-$\mathcal{KB}$*(·) function in the Algorithm 3 provides FREEBASE facts. In Fig. 12, the left rectangles in green (subject entity) indicate each candidate entities (e), the right rectangles in yellow (object entity) represent candidate answers (A_q), and the arrows between entities show logical predicates (p) between entities. We can see that an answer derivation (K_q) is generated by a path from a candidate entity to an object entity (a KB-fact) with a question category (c). In this case, 246,364 derivations are achieved from the 16 candidate FREEBASE entities.

Prediction for the correct answer derivation. After all of answer derivations (K_q) are scored, the top-ranked derivation provide the structured query for KB: $<$ /m/0j3w2, location.location.containedby, $* >$. The symbol $*$ is converted to {Jerusalem} ($\hat{A}_q$) using the knowledge base interaction.

Result analysis. Table 10 shows the ranked list of answers. Each row represents a FREEBASE fact of the derivations. The short question is mostly linked with common.topic.alias because there is no discriminative lexical features: the main verb is "*is*". We also think that a transformed question "*where is #entity*" is highly associated with a logical predicate location.location.containedby in the embedding space, which solve the entity-ambiguous problem.

8. Experiments

8.1. Experimental setting

In this section, we describe the experimental setup used to assess the overall QA performance of our proposed KB-QA system.

Table 10
Top-5 scored answer derivations.

Where is the city of david?			
Rank	Subject entity (e)	Logical predicate (p)	Answer (A)
1	City of David	location.location.containedby	Jerusalem
2	David	common.topic.alias	David de Miquel ngel
3	City of David	common.topic.alias	{Ophel, Jerusalem}
4	City of David	symbols.namesake.named_after	David
5	David	common.topic.alias	King David

Table 11
Descriptive statistics of the relational features.

	No. features		No. features
Lexical feature	267,611	W^u: unigrams	9976
		W^b: bigrams	84,565
		W^t: trigrams	173,070
Logical feature	4734	C: question categories	2
		D: topic domains	85
		T: entity types	1177
		P: predicates	3470

Table 12
Statistics for the tuples of [[lexical pattern ‖ logical predicate]].

Source data	No. tuples	Accuracy (%)
Wikipedia	2,081,784	89.00
Free917	627	97.00
WebQuestions	3786	95.25

First, we performed preprocessing, including lowercase transformation, tokenization, lemmatization, and dependency parsing, using unstructured textual data, i.e., Wikipedia and QA-pairs. The preprocessing step was conducted using the Stanford CoreNLP toolkit.[8] To normalize the data, we removed stop words and question marks, and we transformed any possessive pronouns into the placeholder, *#poss*.

For the labeled-LDA described in Section 5.3, we used 1786 topic labels, i.e., Freebase types, and 223,225 words. Next, we ran the labeled-LDA process by setting the Dirichlet parameters α to 0.01 and β to 0.01, and the iteration number to 2000, which we performed using the Stanford Topic Modeling Toolbox.[9]

As unstructured data, we obtained Wikipedia articles from the 2014-06-15 version of Wikipedia dump[10] and QA-pairs as further unstructured data from two publicly released datasets. One of the datasets was used to measure the performance of our method and the other was employed as the baseline for the QA evaluation. The first dataset, Free917 [9], included the annotated lambda calculus forms for each question, where it covered 81 domains and 635 Freebase relations. The other dataset, WebQuestions [3], comprised 5810 question–answer pairs, which we constructed by collecting common questions from Web-query logs and by manually labeling the answers.

Table 11 shows the descriptive statistics of the features in the semantic embedding space, as described in Section 4.2. These features represent the high-frequency semantic links that remained when entity pairs ($\mathcal{E}$) that occurred at least three times in Wikipedia were used for PMI computations with Eq. (2). The counts of D, T, and P were higher because these features were subordinate to the others in the KB due to its hierarchical structure. The sum of features (272,345) represents the number of embeddings that had to be learned.

During embedding learning, we set the embedding dimension n to 200, the learning rate (λ) for SGD to 0.05, and the range of the margin score for SGD from −1.0 to 3.0. The semantic embedding space reached convergence after 2000 iterations of the learning process, which required 17 days. Fig. 13 shows the convergence flows according to the number of back-propagations and *approximate training losses* during semantic embedding space construction. The *training loss* was calculated by:

$$\text{training loss} = m_i - \text{Sim}(l_i) + \text{Sim}(l_i^{\text{neg}}), \tag{15}$$

where l_i is a semantic link i, l_i^{neg} is a negative pair compared with l_i, and m_i is the margin score between l_i and l_i^{neg}. The *approximate training loss* was computed based on the average *training losses* for all of the training pairs at each iteration, which indicates the degree of the learning process. The learning process was better when the value was close to 0, and therefore, the difference between the similarities of the positive and negative pairs ($\text{Sim}(l_i) - \text{Sim}(l_i^{\text{neg}})$) was equal to the margin score between the two pairs (m_i). After 2000 iterations, the two figures demonstrate that the semantic embedding space was learnt sufficiently well.

8.2. Quality of lexical pattern extraction

In preliminary experiments, we first examined the quality of the training tuples used for relational embedding learning. Table 12 shows the quality of the lexical pattern extraction task described in Sections 5.1 and 5.2. In this study, we determined the parameters used in Eq. (1), i.e., α, β, and γ, as 1, 1, and 2 respectively. In this task, the *lexical patterns* (*relation phrase* and *question patterns*) were extracted from unstructured data and mapped onto the logical predicate of the KB, so we labeled the sample tuples of [[*lexical pattern* ‖Freebase *predicate*]], which we selected randomly as semantically associated or not. The results show that the quality of the *question patterns* obtained from QA-pairs, Free917 and WebQuestions, was fairly good because the QA-pairs were mainly produced manually. In addition, we obtained the *relation phrases* from Wikipedia, which has acceptable quality and it provides a huge number of tuples. The errors were due mainly to incorrect mappings during *distant supervision*, which links *lexical patterns* and Freebase *predicates* based on the entity pairs that appear together. For example, semantically unassociated tuples were extracted when the relations of entity pairs were ambiguous such as [[*novel write by #entity* ‖ people.person.nationality]] (*people-to-country*), and the *lexical pattern* had insufficient semantics such as [[*have #entity* ‖ tv.tv_series_episode.writer]].

8.3. Overall QA performance analysis

Recent QA studies have focused on knowledge-based QA where the KBs are populated, thereby providing some benchmark systems to compare with the proposed method. In the first of our QA evaluations,

8 http://nlp.stanford.edu/software/corenlp.shtml.
9 http://www-nlp.stanford.edu/software/tmt/tmt-0.4/.
10 http://dumps.wikimedia.org/backup-index.html.

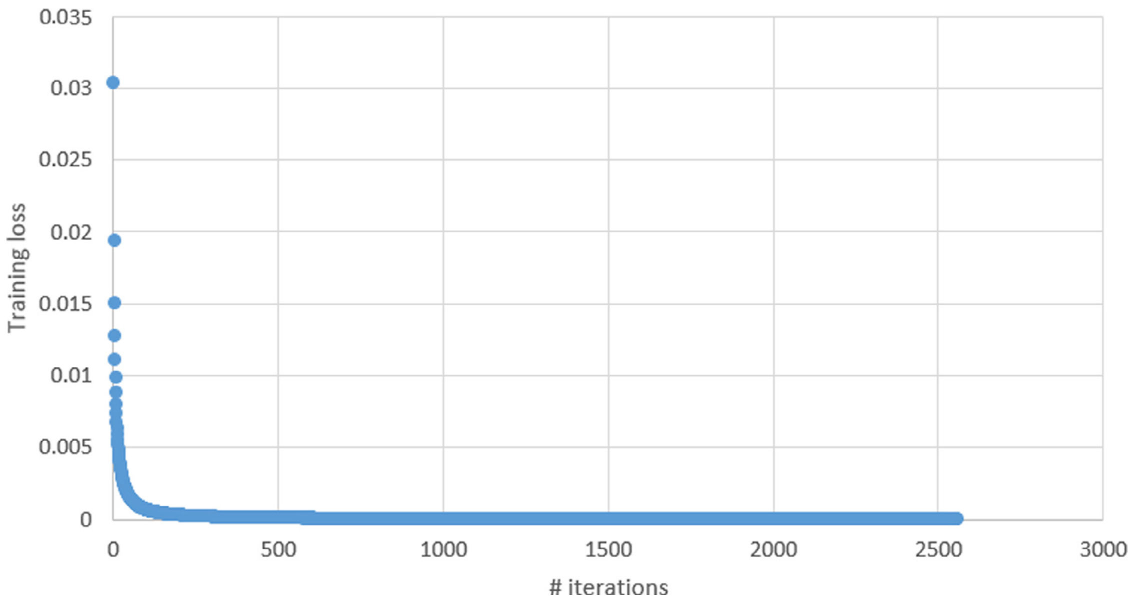


Fig. 13. Convergence of the *approximate training losses* (per iteration) during embedding learning with $n = 200$ and $\lambda = 0.05$.

Table 13
Accuracy of WebQuestions evaluation data.

Methods	Accuracy (%)
Berant et al.	31.40
Bao, Duan, Zhou, and Zhao	37.50
Yang et al.	41.34
Yao and Van Durme	42.00
Berant and Liang	43.00
Bordes et al.	43.20
Our method	44.89

we compared our embedding-based approach with six state-of-the-art methods [2–4,8,36,37], all of which are open-domain QA systems that use KBs.

We describe these baseline methods as follows Semantic parsing-based approaches [3,4,37] transform a natural language question into the corresponding logical form based on grammar formalisms such as Combinatory Categorial Grammar [31] and Dependency-based Compositional Semantics [21]. Next, they use a hand-crafted lexicon to map the parsed words onto the logical properties of a KB, which can build a logical form. The derived logical form predicts an answer using interactions based on the KBs. In particular, Berant and Liang examined a set of questions, which were paraphrased to build a robust lexicon, while Yao and Van Durme exploited the Freebase graph structure for schema matching. Similarly, a machine translation model-based approach [2] was used to translate a natural language question into the corresponding logical form with several features. Embedding model-based approaches [8,36] are quite similar to our method, but we focus on semantic linking between feature-level pairs (words and logical properties) rather than pattern-level pairs (lexical patterns and Freebase predicates). We consider that the feature-level approach provides a more robust embedding space. Furthermore, in contrast to Bordes et al., we construct our own semantic links from Wikipedia, which can build larger embedding spaces in terms of size when tackling natural language expressions.

Table 13 shows the overall performances of our proposed KB-QA method based on the evaluation dataset WebQuestions, where the results are compared with those obtained by the six baseline methods described above. Note that we did not re-implement the baseline systems, but instead we refer to the experimental results reported previously. To ensure that the KB-QA evaluations were fair, we also used Freebase as the KB, which was also the case for the baseline systems. Table 13 shows that our method outperformed the baseline methods with 2,032 QA-pairs in the WebQuestions dataset in terms of the accuracy of correct answer prediction for the given questions. The accuracy represents the average of the F1-scores for each QA-pair, as follows:

$$\begin{aligned}
\text{Accuracy} &= \text{Avg}(F1-\text{score}) = 2 \cdot \frac{\text{Avg}(P)\cdot\text{Avg}(R)}{\text{Avg}(P)+\text{Avg}(R)}\\
\text{Precision}(P) &= \frac{|\,\text{Gold}-\text{Standard}\cap\text{Predicted}\,|}{|\,\text{Predicted}\,|}\\
\text{Recall}(R) &= \frac{|\,\text{Gold}-\text{Standard}\cap\text{Predicted}\,|}{|\,\text{Gold}-\text{Standard}\,|},
\end{aligned} \tag{16}$$

where *Gold-Standard* is the set of correct answers for a given question statement in QA-pairs and *Predicted* is the set of answers returned (top-1) for a given question statement by a specific method. Our improved QA evaluations may be explained by the following reasons. (1) We consider that using the low-dimensional embeddings of *n*-grams rather than lexical triggers improves the coverage issue greatly. As shown in Table 11, a huge number of lexical features can address various natural language questions. (2) In contrast to the knowledge interpretation approach used in other methods based on a mapping table from words onto logical properties, our semantic embedding space obtains full interpretations of the KB. Thus, joint semantic embedding of the lexical and logical spaces can exploit the semantic associations between any features in the same vector space. (3) Syntax-based (grammar formalism) approaches such as Combinatory Categorial Grammar [9], Lambda Dependency-Based Compositional Semantics [3,4], and dependency graphs [37] may produce errors if a question contains grammatical errors. By contrast, the bag-of-words model-based approach employed in our method can handle any question provided that the question contains keywords that can help its understanding.

To assess the answer coverage (recall) of candidate answer derivations (K_q), we evaluated our method by comparing it with other methods in terms of the oracle accuracy. The oracle accuracy is defined as follows:

$$\text{Oracle Accuracy} = \frac{|\,\{q_o\,|\,\forall q\in Q,\ q_o:\ \text{if Gold}-\text{Standard}_q\in K_q\}\,|}{|\,Q\,|}, \tag{17}$$

Table 14
Oracle accuracy of WebQuestions evaluation data.

Methods	Oracle accuracy (%)
Berant et al.	48.00
Berant and Liang	63.00
Our method	87.45

Table 15
Impacts of lexical features on WebQuestions evaluation data.

Methods	P@1 (%)	R@1 (%)	F1 (%)
Unigram (W^u)	26.11	39.68	31.49
Bigram (W^t)	34.41	51.78	41.34
Trigram (W^b)	36.26	54.34	43.49
$W^u + W^b + W^t$	37.31	56.35	44.89

Table 16
Impacts of logical features on WebQuestions evaluation data.

Methods	P@1 (%)	R@1 (%)	F1 (%)
C + P	37.13	55.91	44.62
C + D + P	37.11	56.00	44.64
C + T + P	37.26	56.24	44.82
C + D + T + P	37.31	56.35	44.89

where *Gold-Standard$_q$* is the set of correct answers for a question statement q and Q is the set of QA-pairs in WebQuestions. Thus, we considered that the derivation obtained (K_q) was correct (q_o) if at least one gold standard answer was present in K_q. Table 14 shows that the oracle accuracy of our method was higher than that of the other two methods. The other two methods never found most of the correct answers because there were no appropriate candidate derivations for the correct answer in the previous stage due to the error propagation problem. By contrast, our method had more opportunities to find the correct answers.

8.4. Individual components analysis

In the second part of the QA evaluations, we determined the QA performance of the individual components (the individual features and training pairs used for embedding learning) of our method. We evaluated the QA performance by excluding individual components to analyze the impact of each component based on the degree of QA performance degradation.

Table 15 shows the QA contributions of the lexical features, i.e., unigram (W^u), bigram (W^b), and trigram (W^t), using three evaluation metrics: P@1 (*precision at top-1*), R@1 (*recall at top-1*), and F1 (*F1-score*). For each row, we only used the similarity between each n-gram and all of the logical features in the pairwise feature set to predict the best derivation, as described in Section 7.2. The results show that the performance of individual n-grams tended to be similar to their estimated weight, as shown in Fig. 11. We consider that trigram and bigram features make greater contributions compared with other n-grams (Fig. 11), but they are also discriminatory by themselves. By contrast, unigram features were not effective for identifying the semantics of natural language utterances, because unigrams are sometimes ambiguous. Thus, some unigram features share multiple semantics such as homonyms.

Table 16 shows the QA contributions of the logical features: logical predicate (P), topic domain (D), and entity type (T). In each row, we only used the similarity between a combination of logical features (with or without the symbols) and all of the lexical features in the pairwise feature set to predict the best derivation, as described in Section 7.2. The question category (C) was not evaluated because C is not used when the candidate derivations are ranked. The results show that the performance of individual logical features tended to be similar to their estimated weights, as shown in Fig. 11. Furthermore, the logical predicate P played a crucial role when determining the correct answers for a natural language question, whereas the other two logical features (D and T) were only slightly helpful.

Table 17
Excluding individual steps using WebQuestions evaluation data.

Methods	P@1 (%)	R@1 (%)	F1 (%)
– Relation phrase extraction (Section 5.1)	32.12	51.10	39.45
– Question pattern extraction (Section 5.2)	18.12	30.16	22.64
– Semantic link filtering (Section 5.3)	35.30	53.97	42.68
All (our method)	37.31	56.35	44.89

Table 17 shows the QA performance obtained after excluding lexical pattern extraction, as described in Sections 5.1 and 5.2, and post-processing in the semantic linking step, as described in Section 5.3. Among the *lexical patterns*, we found that *question patterns* were more discriminatory lexical features compared with *relation phrases*. The QA performance was higher after removing the relation phrase extraction step compared with that after excluding the question pattern extraction step. Post-processing with distributional semantics (labeled-LDA) was effective because the QA performance was lower when semantic link filtering was not executed. We consider that post-processing helps to filter out semantically unassociated training pairs from the semantic embeddings.

8.5. Error analysis

As a QA evaluation dataset, WebQuestions is a challenging dataset because it was created by non-experts and constructed using a wide vocabulary. Thus, our method could not find the correct answers for many questions because the majority of the questions in WebQuestions tend to be natural and diverse. These errors may be explained by any of the following reasons.

- Sometimes, KBs might not include the answer information for complex questions, such as multi-related questions. For example, given the QA-pair, *Q: what country did germany invade first in ww1?, A:*Belgium, it is difficult to identify the target entities in Q as well as capturing the logical predicate for Q. In particular, the appropriate logical predicate for Q does not exist in the KB. In this case, we can never find the correct answer even if the question is fully understood.
- Most of the errors obtained with our method occurred because we could not map a question statement to the corresponding logical predicate, i.e., the predicted logical predicates were wrong. For example, we obtained Europe as the suggested answer for a QA-pair: *Q: where is french spoken most?, A:*France. Our answer is not incorrect, but there is a more appropriate answer in the KB. To address this problem, we should focus on the adverb word, *most*, to predict the corresponding logical predicate: language.human_language.main_country, not language.human_language.region.
- More advanced inference is required when formulating answer statements. For example, given the QA-pair: *Q: which dawkins books was written first?, A:*The Selfish Gene, we obtained a list of books written by Dawkins, but the user only wanted his first book, i.e., The Selfish Gene. In this case, if an ordinal number is detected in the question statements, we should formulate the answer statements according to the identified ordinal number based on inference rules.
- Some questions are too trivial, e.g., the QA-pair: *Q: what did romo do?, A:*Baseball player. The word Romo is too ambiguous and

he is not a popular person for people who are not interested in baseball. In order to answer this question, we need more specific information about the target entity, such as Romo's first name.

9. Conclusion and future work

In this study, we proposed a novel method that answers natural language questions in any domains by using the semantic embedding space. Our method employs the following three steps. (1) We extract a large number of lexical patterns from unstructured textual data, such as Wikipedia. The high quality semantic links required to learn the semantic embedding space can be obtained based on *distant supervision*, labeled-LDA, and a PMI computation-based scoring model. (2) We jointly learn the semantic embeddings of words and KB-properties. For the QA task, a question statement can be analyzed by using the word embedding and the answer can be inferred from the knowledge embedding. The semantic embedding space can also exploit the semantic relationships between words and the logical properties of the KB. (3) We obtain answer statements for factoid questions using the semantic embedding space. In particular, we can compute the semantic similarities between the words of a given question and a logical representation of potential answers over the embedding space. Our experimental results demonstrated that the proposed method performed better than six state-of-the-art KB-QA baseline systems in terms of the overall QA performance and the validity of the candidate answers.

In our future research, we will conduct further experiments to examine the impacts of the parameters employed by our method such as the embedding dimension. We will also apply advanced inference to obtain the correct answers in terms of the user's requirements. Thus, we will construct training QA-pairs with various question patterns and expected answer types. In addition, we will combine our current method with a semantic parsing technique, which can parse and analyze a question statement.

Acknowledgments

This research was supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT and Future Planning (NRF-2012M3C4A7033344).

References

Atkinson, J., Figueroa, A., & Andrade, C. (2013). Evolutionary optimization for ranking how-to questions based on user-generated contents. *Expert Systems with Applications, 40*(17), 7060–7068. doi:10.1016/j.eswa.2013.06.017.

Bao, J., Duan, N., Zhou, M., & Zhao, T. (2014). Knowledge-based question answering as machine translation. In *Proceedings of the 52nd annual meeting of the association for computational linguistics* (pp. 967–976). Association for Computational Linguistics.

Berant, J., Chou, A., Frostig, R., & Liang, P. (2013). Semantic parsing on Freebase from question–answer pairs. In *Proceedings of the 2013 conference on empirical methods in natural language processing* (pp. 1533–1544). Seattle, Washington, USA: Association for Computational Linguistics. http://www.aclweb.org/anthology/D13-1160

Berant, J., & Liang, P. (2014). Semantic parsing via paraphrasing. In *Proceedings of the 52nd annual meeting of the association for computational linguistics* (pp. 1415–1425). Association for Computational Linguistics.

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet allocation. *The Journal of Machine Learning Research, 3*, 993–1022.

Bollacker, K., Evans, C., Paritosh, P., Sturge, T., & Taylor, J. (2008). Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on management of data*. In *SIGMOD '08* (pp. 1247–1250). New York, NY, USA: ACM. doi:10.1145/1376616.1376746.

Bordes, A., Chopra, S., & Weston, J. (2014). Question answering with subgraph embeddings. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 615–620). Doha, Qatar: Association for Computational Linguistics. http://www.aclweb.org/anthology/D14-1067.

Bordes, A., Weston, J., & Usunier, N. (2014). Open question answering with weakly supervised embedding models. In *Machine learning and knowledge discovery in databases—European conference, ECML PKDD 2014, Nancy, France, September 15–19, 2014. proceedings, part I* (pp. 165–180). doi:10.1007/978-3-662-44848-9_11.

Cai, Q., & Yates, A. (2013). Large-scale semantic parsing via schema matching and lexicon extension. In *Association for computational linguistics (ACL)* (pp. 423–433). The Association for Computer Linguistics.

Church, K. W., & Hanks, P. (1990). Word association norms, mutual information, and lexicography. *Computational Linguistics, 16*(1), 22–29. http://dl.acm.org/citation.cfm?id=89086.89095.

Fader, A., Soderland, S., & Etzioni, O. (2011). Identifying relations for open information extraction. In *Proceedings of the conference on empirical methods in natural language processing*. In *EMNLP '11* (pp. 1535–1545). Stroudsburg, PA, USA: Association for Computational Linguistics.

Fader, A., Zettlemoyer, L., & Etzioni, O. (2014). Open question answering over curated and extracted knowledge bases. In *Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining*. In *KDD '14* (pp. 1156–1165). New York, NY, USA: ACM. doi:10.1145/2623330.2623677.

Fader, A., Zettlemoyer, L. S., & Etzioni, O. (2013). Paraphrase-driven learning for open question answering. In *Association for computational linguistics (ACL)* (pp. 1608–1618). The Association for Computer Linguistics.

Ferrucci, D. A., Brown, E. W., Chu-Carroll, J., Fan, J., Gondek, D., Kalyanpur, A., … Welty, C. A. (2010). Building watson: An overview of the DEEPQA project. *AI Magazine, 31*(3), 59–79. http://www.aaai.org/ojs/index.php/aimagazine/article/view/2303

Figueroa, A., & Neumann, G. (2014). Category-specific models for ranking effective paraphrases in community question answering. *Expert Systems with Applications, 41*(10), 4730–4742. doi:10.1016/j.eswa.2014.02.004.

Gerber, D., & Ngonga Ngomo, A.-C. (2011). Bootstrapping the linked data web. In *1st workshop on web scale knowledge extraction @ ISWC 2011*.

Hoffmann, R., Zhang, C., Ling, X., Zettlemoyer, L., & Weld, D. S. (2011). Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies—Volume 1*. In *HLT '11* (pp. 541–550). Stroudsburg, PA, USA: Association for Computational Linguistics. http://dl.acm.org/citation.cfm?id=2002472.2002541.

Kolomiyets, O., & Moens, M.-F. (2011). A survey on question answering technology from an information retrieval perspective. *Information Sciences, 181*(24), 5412–5434. doi:10.1016/j.ins.2011.07.047.

Krishnamurthy, J., & Mitchell, T. M. (2012). Weakly supervised training of semantic parsers. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*. In *EMNLP-CoNLL '12* (pp. 754–765). Stroudsburg, PA, USA: Association for Computational Linguistics. http://dl.acm.org/citation.cfm?id=2390948.2391030.

Kwiatkowski, T., Choi, E., Artzi, Y., & Zettlemoyer, L. (2013). Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the 2013 conference on empirical methods in natural language processing* (pp. 1545–1556). Seattle, Washington, USA: Association for Computational Linguistics. http://www.aclweb.org/anthology/D13-1161.

Liang, P., Jordan, M. I., & Klein, D. (2011). Learning dependency-based compositional semantics. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies—Volume 1*. In *HLT '11* (pp. 590–599). Stroudsburg, PA, USA: Association for Computational Linguistics.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems 26: 27th annual conference on neural information processing systems 2013. Proceedings of a meeting held December 5–8, 2013, Lake Tahoe, Nevada, United States* (pp. 3111–3119).

Mintz, M., Bills, S., Snow, R., & Jurafsky, D. (2009). Distant supervision for relation extraction without labeled data. In *Proceedings of the joint conference of the 47th annual meeting of the ACL and the 4th international joint conference on natural language processing of the AFNLP: Volume 2*. In *ACL '09* (pp. 1003–1011). Stroudsburg, PA, USA: Association for Computational Linguistics.

Mooney, R. (2007). Learning for semantic parsing. In A. Gelbukh (Ed.), *Computational linguistics and intelligent text processing*. In *Lecture notes in computer science: 4394* (pp. 311–324). Berlin/Heidelberg: Springer. doi:10.1007/978-3-540-70939-8_28.

Nakashole, N., Weikum, G., & Suchanek, F. (2012). PATTY: A taxonomy of relational patterns with semantic types. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*. In *EMNLP-CoNLL '12* (pp. 1135–1145). Stroudsburg, PA, USA: Association for Computational Linguistics. http://dl.acm.org/citation.cfm?id=2390948.2391076.

Paredes-Valverde, M. A., Rodrguez-Garca, M. Á., Ruiz-Martnez, A., Valencia-Garca, R., & Alor-Hernndez, G. (2015). ONLI: An ontology-based system for querying {DBpedia} using natural language paradigm. *Expert Systems with Applications*, (0). doi:10.1016/j.eswa.2015.02.034.

Pavli, M., Han, Z. D., & Jakupovi, A. (2015). Question answering with a conceptual framework for knowledge-based system development node of knowledge. *Expert Systems with Applications*, (0). doi:10.1016/j.eswa.2015.02.024.

Ramage, D., Hall, D., Nallapati, R., & Manning, C. D. (2009). Labeled LDA: A supervised topic model for credit attribution in multi-labeled corpora. In *Proceedings of the 2009 conference on empirical methods in natural language processing* (pp. 248–256). Singapore: Association for Computational Linguistics. http://www.aclweb.org/anthology/D/D09/D09-1026.

Riedel, S., Yao, L., McCallum, A., & Marlin, B. M. (2013). Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 conference of the North American chapter of the association for computational linguistics: Human language technologies* (pp. 74–84). Atlanta, Georgia: Association for Computational Linguistics. http://www.aclweb.org/anthology/N13-1008.

Rodrigo, Á., Prez-Iglesias, J., Peas, A., Garrido, G., & Araujo, L. (2013). Answering questions about European legislation. *Expert Systems with Applications, 40*(15), 5811–5816. doi:10.1016/j.eswa.2013.05.008.

Steedman, M. (2000). *The syntactic process*. The MIT Press.

Unger, C., Bühmann, L., Lehmann, J., Ngonga Ngomo, A.-C., Gerber, D., & Cimiano, P. (2012). Template-based question answering over RDF data. In *Proceedings of the 21st international conference on world wide web*. In *WWW '12* (pp. 639–648). New York, NY, USA: ACM. doi:10.1145/2187836.2187923.

Weston, J., Bengio, S., & Usunier, N. (2010). Large scale image annotation: Learning to rank with joint word-image embeddings. *Machine Learning, 81*(1), 21–35. doi:10.1007/s10994-010-5198-3.

Weston, J., Bengio, S., & Usunier, N. (2011). WSABIE: Scaling up to large vocabulary image annotation. In *Proceedings of the twenty-second international joint conference on artificial intelligence—Volume three*. In *IJCAI'11* (pp. 2764–2770). AAAI Press. doi:10.5591/978-1-57735-516-8/IJCAI11-460.

Weston, J., Bordes, A., Yakhnenko, O., & Usunier, N. (2013). Connecting language and knowledge bases with embedding models for relation extraction. In *Proceedings of the 2013 conference on empirical methods in natural language processing* (pp. 1366–1371). Seattle, Washington, USA: Association for Computational Linguistics. http://www.aclweb.org/anthology/D13-1136.

Yang, M.-C., Duan, N., Zhou, M., & Rim, H.-C. (2014). Joint relational embeddings for knowledge-based question answering. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 645–650). Doha, Qatar: Association for Computational Linguistics. http://www.aclweb.org/anthology/D14-1071.

Yao, X., & Van Durme, B. (2014). Information extraction over structured data: Question answering with freebase. In *Proceedings of the 52nd annual meeting of the association for computational linguistics* (pp. 956–966). Baltimore, Maryland: Association for Computational Linguistics.

Zelle, J. M., & Mooney, R. J. (1996). Learning to parse database queries using inductive logic programming. In *Proceedings of the thirteenth national conference on artificial intelligence—Volume 2*. In *AAAI'96* (pp. 1050–1055). AAAI Press. http://dl.acm.org/citation.cfm?id=1864519.1864543

Zettlemoyer, L. S., & Collins, M. (2005). Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Uai* (pp. 658–666). AUAI Press. http://dblp.uni-trier.de/db/conf/uai/uai2005.html#ZettlemoyerC05.