Answering Questions with Complex Semantic Constraints on Open Knowledge Bases

Pengcheng Yin[†] Nan Duan[‡] Ben Kao[†] Junwei Bao[§] Ming Zhou[‡]

[†] The University of Hong Kong [‡] Microsoft Research Asia [§] Harbin Institute of Technology

[†] {pcyin, kao}@cs.hku.hk [‡§] {nanduan, v-jubao, mingzhou}@microsoft.com

ABSTRACT

A knowledge-based question-answering system (KB-QA) is one that answers natural language questions with information stored in a large-scale knowledge base (KB). Existing KB-QA systems are either powered by *curated KBs* in which factual knowledge is encoded in entities and relations with well-structured schemas, or by *open KBs*, which contain assertions represented in the form of triples (e.g., ⟨*subject; relation phrase; argument*⟩). We show that both approaches fall short in answering questions with *complex* prepositional or adverbial constraints. We propose using n-tuple assertions, which are assertions with an arbitrary number of arguments, and n-tuple open KB (nOKB), which is an open knowledge base of n-tuple assertions. We present TAQA, a novel KB-QA system that is based on an nOKB and illustrate via experiments how TAQA can effectively answer complex questions with rich semantic constraints. Our work also results in a new open KB containing 120M n-tuple assertions and a collection of 300 labeled complex questions, which is made publicly available[1] for further research.

1. INTRODUCTION

A question-answering (QA) system is one that automatically answers questions posed in natural languages (NLs). Many of these systems are powered by knowledge bases (KBs), whose information is often encoded as entities and relations with well-structured schemas. With a knowledge-based QA (KB-QA) system, a natural-language question is typically answered in two steps: (1) the free-form natural-language question is transformed into a structured query (e.g., SPARQL query [18]); and (2) answers are retrieved by executing the structured query against a KB [4, 5, 13, 12, 3, 18]. Most of existing research on KB-QA systems focuses on answering questions with *simple semantic constraints*, which are often expressed by the verb phrase of one or multiple binary relations, such as:

single relation: *Who* $[invented]_{rel}$ *the telephone?* [4]

multiple relations: *Who* $[was\ married\ to]_{rel1}$ *an actor that* $[played\ in]_{rel2}$ *Forrest Gump?* [18]

Relatively few efforts, however, have been spent on answering questions with rich semantic constraints, such as those that are expressed via prepositional or adverbial modifiers. As an example, consider the following question:

Question: *What was the currency of Spain before 2002?*
Correct Answer: *Peseta* **Incorrect Answer:** *Euro.*

In this example, the prepositional phrase *"before 2002"* imposes a temporal constraint of the question. As we will elaborate shortly, such constraints are not conveniently handled by QA systems built on existing KB models. Very often, such constraints are lost during the transformation of questions into KB queries, resulting in incorrect answers (e.g., the answer "Euro" is returned instead of "Peseta"). The objective of this paper is to study how questions with complex semantic constraints, particularly those expressed via prepositional or adverbial phrases, can be answered on a KB-QA system. For convenient, we call those questions with complex semantic constraints *"complex questions"*, while those without *"simple questions"*.

To understand the difficulties of existing KB-QA systems in answering complex questions, let us first briefly describe their underlying KBs. KB-QA systems can be categorized into *curated KB-QA systems* and *open KB-QA systems*. While the former has a longer history with a number of implementations (e.g., DEANNA [18] and PARASEMPRE [5]), the latter has been recently proposed by Fader et al. (PARALEX [13] and OQA [12]). A curated KB-QA system is built upon a curated KB (e.g., Freebase [6] and DBpedia [1]), which is collaboratively and manually created. Information is modeled as entities and relations that strictly follow a predefined schema. Figures 1(a) and 1(c) show two snippets taken from Freebase. Since a curated KB is manually edited, it is generally accurate and precise. There are, however, two aspects of it that weaken its ability in answering complex questions.

[Query transformation] An open question has to be first transformed into a structured query (such as SPARQL query) that is compatible with the KB schema, after which the query is executed against the KB to obtain answers. For example, Figure 1(b) shows a SPARQL query (Y_1) for the question Q_1: *"What was James K. Polk?"* This query is obtained by observing that `government_position` is an occupation relation that semantically matches the verb *"was"* in Q_1. The transformation from Q_1 to Y_1 is straightforward considering that Y_1 involves only one relation in the KB. Now consider a complex question Q_2: *"What was James K. Polk before he was president?"* The additional semantic

[1]`http://taqa.pcyin.me`

CIKM'15, October 19–23, 2015, Melbourne, Australia.
© 2015 ACM. ISBN 978-1-4503-3794-6/15/10 ...$15.00.
DOI: http://dx.doi.org/10.1145/2806416.2806542.

constraint given in Q_2 leads to a much more complex query Y_2. Although the temporal information of the occupation that is needed to answer Q_2 has been encoded using Compound Value Types (CVTs) in Freebase, as denoted by two $\otimes$ nodes in Figure 1(a), existing curated KB-QA systems do not yet have the logic to leverage this information and correctly generate the required queries of complex questions in general. As a result, complex constraints (such as *"before he was president"*) of questions are often neglected and incorrect answers ensue.

[Incompleteness] Another problem faced by curated KB-QA systems is the often incomplete KB schema. This often results in insufficient knowledge to resolve the complex constraints posed in a question. For example, the question $Q3$: *"What was the currency of Spain before 2002?"* cannot be answered by the Freebase snippet shown in Figure 1(c) because the years of circulation of the currencies are not recorded in the KB.[2]

Recently, Fader et al. put forward open KB-QA systems as an alternative approach for KB-QA [12, 13]. An open KB [14] is a large collection of n-tuple assertions that are automatically extracted from web corpora by means of *open information extraction* (open IE) techniques [2, 11, 8, 17]. An assertion is a string n-tuple ($n \geq 3$) which contains one subject (*sbj*), one relation phrase (*rel*), and multiple arguments (*arg*) in the form:

$$\langle sbj; rel; Arg = \{arg_1, arg_2, \cdots, arg_{n-2}\}\rangle.$$

Each element in the tuple is called a *field*. Table 1 gives examples of such n-tuple assertions. The first assertion A_1, for example, is a 4-tuple: it consists of 1 subject (field), 1 relation phrase (field) and 2 arguments (fields). Since assertions in an open KB are automatically acquired from unstructured text, they are often unnormalized and noisy. Nevertheless, two unique features of open KB make it potentially more suitable for QA. First, as assertions are extracted from text, all of their fields are strings in a natural language. They can therefore naturally be used to answer NL questions by string matchings. In particular, unlike a curated KB, there is not a predefined schema and hence no complex query transformation is needed. Second, assertions in n-tuple form contain rich semantic information, which can be used to answer questions with complex semantic constraints. For example, Question Q_2, which is difficult to handle by a curated KB, can be easily answered by matching a derived *tuple query* ⟨"*James K. Polk*"; "*was*"; *?x*, "*before he was president*"⟩ against assertion A_1 in Table 1.

Although open KBs stand out in handling complex questions, the state-of-the-art open KB-QA system, OQA [12], is designed to operate on *triplet* assertions, each of which contains only one single argument (in addition to a subject and a relation phrase). In fact, the OQA system is powered by such an open KB of triples. This restricted model of assertions (triples instead of n-tuple) limits the ability of OQA in answering questions with complex semantic constraints. It is non-trivial to adapt the methodology of OQA to an n-tuple open KB for two reasons. First, questions with semantic modifiers exhibit a variety of syntactic forms. OQA, however, parses simple questions into tuple queries using a small number of hand-crafted Part-of-Speech (POS) templates. While this approach works very well with triples under a rigid subject-relation-argument format, it is difficult to extend those templates to cover questions with complex semantic constraints, especially those that require (n-tuple) assertions with multiple arguments (e.g., assertions A_1 and A_2) to derive answers. Second, the query language of OQA only supports querying triplet assertions. The query language needs to be extended to handle n-tuple open KB, whose assertions contain arbitrary numbers of arguments.

Considering the limitations of existing KB-QA systems in answering complex questions, our goal is to design and implement a novel open KB-QA system that can fully leverage the rich semantics of n-tuple assertions for answering questions with complex semantic constraints. To achieve this goal, two key components in an open KB-QA system, *question parsing* and *open KB querying*, have to be redesigned. As an overview, we propose a generic question parsing method that is based on dependency parsing. Our method is able to parse syntactically diverse questions with rich semantic constraints. Moreover, we put forward an alignment-based answer extraction approach, which is capable of efficiently answering tuple queries using assertions with an arbitrary number of arguments. Apart from these two contributions which focus on leveraging n-tuple open KBs, we also improve the general quality of KB-QA by designing a question paraphraser powered by rich *association features* [5]. Question paraphrasing has been proven important for open KB-QA in bridging the lexical/sytactic gap between user-issued questions and relevant KB assertions. Existing works employ shallow statistical features (e.g., PMI values) to evaluate each paraphrased question, which is unreliable when the dataset used for paraphrasing is noisy. To improve accuracy, we instead use rich association features to learn soft paraphrase rules.

In summary, the main contributions of our paper are:

[nOKB] Existing open KBs (such as OPEN IE [12], PROBASE [19] and NELL [7]) contain only triples. As we have explained, answering complex questions often require n-tuple assertions of multiple arguments because they are much richer in their semantic contents. Existing open KBs are thus inadequate. We fill this void by creating an n-tuple open KB (called nOKB) of 120M assertions together with an evaluation question set of 300 (and counting) complex questions with labeled reference answers, which is released to the public to facilitate further research.

[TAQA] We designed and implemented an n-Tuple-Assertion-based Question-Answering system (called TAQA) that operates on an nOKB for answering questions with complex semantic constraints. We detail the key features of all the major components of TAQA and discuss how they were designed to work with n-tuple assertions.

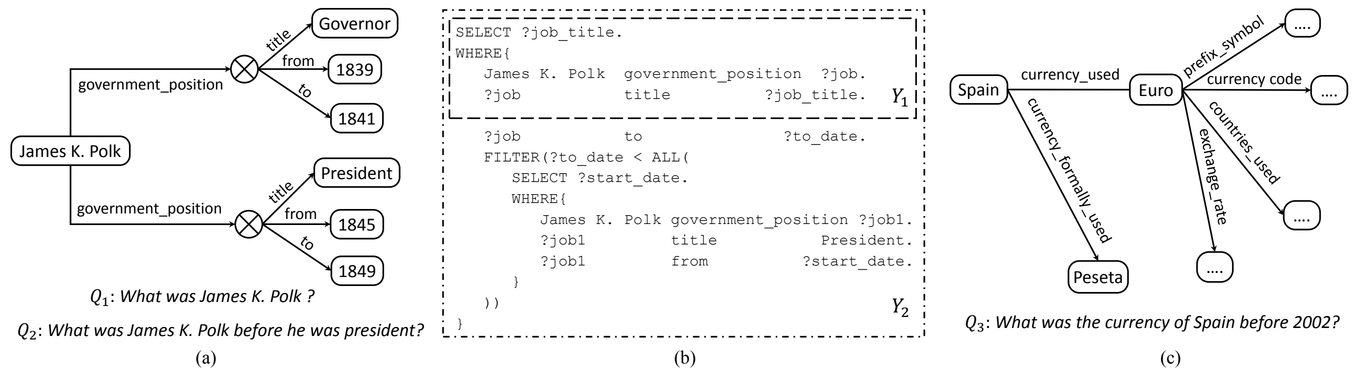
[Experiments] We conducted extensive experiments to compare TAQA against state-of-the-art open KB-QA and curated KB-QA systems. Our results show that TAQA outperforms other KB-QA systems in answering complex questions. This shows strong evidence that n-tuple assertions are indispensable in answering questions with rich semantic constraints.

The rest of the paper is organized as follows. Section 2 describes the workflow of TAQA and its major components. Section 3 shows our experimental results comparing TAQA against other state-of-the-art KB-QA systems. Section 4 further illustrates our proposed system by means of a case study. Section 5 discusses related works. Finally, Section 6 concludes the paper and discusses several future research directions.

[2]This could have been solved by introducing a CVT node.

Id	Subject	Relation Phrase	Arguments	Frequency	Confidence
A_1	James K. Polk	was	a governor, before he was president	2	0.94
A_2	the currency of Spain	was	the Peseta, before 2002	3	0.86
A_3	the Euro	is	national currency of Spain, since 2002	5	0.78
A_4	the Peseta	was	the currency of Spain	3	0.85
A_5	Spain	introduced	the Euro, as legal tender, in Jan. 2002	2	0.77
A_6	the Vatican Lira	was replaced	by Euro, as official currency, in 2002	1	0.90
A_7	the currency of France	was	the France Franc, before 2002	2	0.77
A_8	the Greek currency	was	the drachma, before 2002	4	0.94

Table 1: Example assertions in nOKB. Arguments fields are separated by commas.



```
SELECT ?job_title.
WHERE{
   James K. Polk  government_position  ?job.
   ?job           title                ?job_title.
   ?job           to                   ?to_date.
   FILTER(?to_date < ALL(
      SELECT ?start_date.
      WHERE{
         James K. Polk government_position ?job1.
         ?job1         title               President.
         ?job1         from                ?start_date.
      }
   ))
}
```

Figure 1: Freebase snippets and example SPARQL queries

2. SYSTEM

In this section we introduce our system TAQA. We first give an overview of TAQA's workflow, followed by implementation details of each major component.

2.1 Overview

Given a natural language question Q, TAQA processes the question and outputs a list of ranked candidate answers $\mathcal{A}$. Figure 2 gives an illustrative example that shows the various steps involved. There are four key components:

Question Paraphrasing (Section 2.2), which rewrites the original user-issued question into a set of paraphrased questions. Web users generally use informal and casual wordings and expressions in questions. These questions are difficult to parse and are hard to match with assertions in an open KB. Question paraphrasing reformulates original informal questions into formalized ones, which share similar grammatical structures and vocabulary with open KB assertions. Moreover, each paraphrased question derives tuple queries that are executed against the KB for answers. By rewriting the user's question into multiple paraphrased questions, multiple queries are generated. This enhances the recall of the QA system. To improve the quality of question paraphrasing, in TAQA, we employ rich *association features* to help find high-quality paraphrased questions.

Question Parsing (Section 2.3), which parses each paraphrased question into a set of *tuple queries*. A tuple query (e.g., ⟨*?x*; "*wrote*"; "*Harry Potter*"⟩) is similar to an n-tuple assertion except that one of its fields is unknown. TAQA employs a novel question parser that is able to identify rich semantic constraints expressed in the question by analyzing its dependency structure.

Open KB Querying (Section 2.4), which executes each tuple query against the open KB to obtain a list of candidate answers. In order to effectively answer tuple queries using semantically rich n-tuple assertions, we propose a novel alignment-based answer extraction approach that is capable of pinpointing answers from open KB assertions that have various numbers of arguments.

Answer Ranking (Section 2.5), which consolidates and ranks candidate answers derived from different tuple queries. Each candidate answer is associated with a set of features that are obtained through the various steps in the whole question-answering process. TAQA employs a log-linear model on the features to rank the candidate answers. Due to the large feature space (which consists of over 20,000 sparse features), we use AdaGrad [10] to facilitate fast and effective learning of the ranking model.

2.2 Question Paraphrasing

We follow the approach taken by OQA [12] of paraphrasing a question using *paraphrasing templates*. A paraphrasing template (PT) has the form

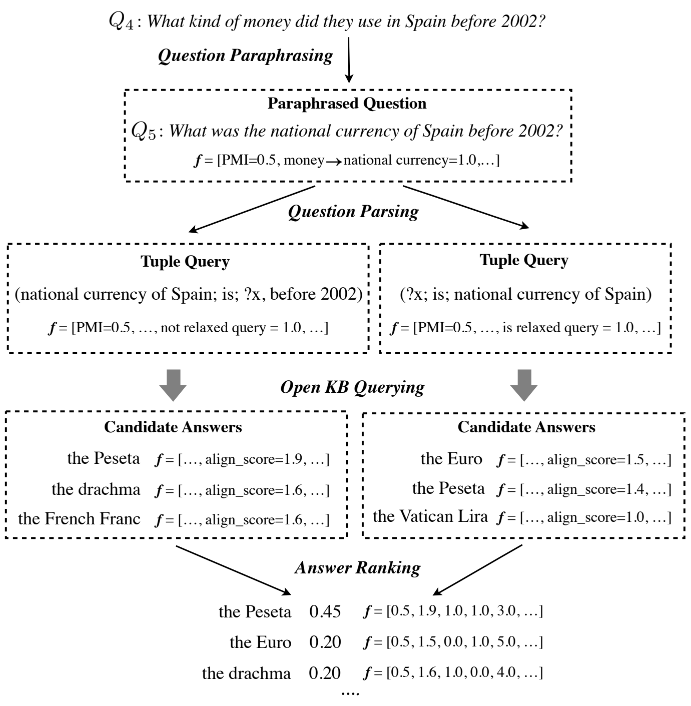
$$PT : p_{src} \mapsto p_{tgt}$$

where p_{src} and p_{tgt} are the source and target templates, respectively. Each such template is a string with wildcard variables. The purpose of a PT is to map a syntactic pattern to an alternative one to facilitate the retrieval of relevant assertions in the open KB. The following is an example PT:

PT_1: *What kind of money did they use in* $*_{NP} \mapsto$
What was the national currency of $*_{NP}$

where $*_{NP}$ is a wildcard variable that captures a noun phrase. A user-issued question Q that matches the source template of a PT is rewritten according to the target template of the PT. Complex semantic constraints expressed as prepositional/adverbial phrases in Q are left out in template matching. These phrases are retained in the rewritten questions in their original form. An example of applying PT_1 on question Q_4 is shown in Figure 2. Note that the rewritten question Q_5 allows a perfect match with assertion A_2 (Table 1), which provides the best answer to Q_4.

We use the WIKIANSWERS paraphrasing templates dataset released by [12], which contains 5 million templates created by mining related questions tagged by users on WIKIAN-



Q_4: What kind of money did they use in Spain before 2002?
Question Paraphrasing
Paraphrased Question
Q_5: What was the national currency of Spain before 2002?
f = [PMI=0.5, money→national currency=1.0,...]
Question Parsing
Tuple Query
(national currency of Spain; is; ?x, before 2002)
f = [PMI=0.5, ..., not relaxed query = 1.0, ...]
Tuple Query
(?x; is; national currency of Spain)
f = [PMI=0.5, ..., is relaxed query = 1.0, ...]
Open KB Querying
Candidate Answers
the Peseta f = [..., align_score=1.9, ...]
the drachma f = [..., align_score=1.6, ...]
the French Franc f = [..., align_score=1.6, ...]
Candidate Answers
the Euro f = [..., align_score=1.5, ...]
the Peseta f = [..., align_score=1.4, ...]
the Vatican Lira f = [..., align_score=1.0, ...]
Answer Ranking
the Peseta 0.45 f = [0.5, 1.9, 1.0, 1.0, 3.0, ...]
the Euro 0.20 f = [0.5, 1.5, 0.0, 1.0, 5.0, ...]
the drachma 0.20 f = [0.5, 1.6, 1.0, 0.0, 4.0, ...]
....

Figure 2: TAQA's workflow

Algorithm 1: question parsing algorithm

```
   input  : natural language question $Q$
   output: a set of tuple queries $\mathcal{T}_Q$
1  $\mathcal{T}_Q = \emptyset$, $AnsType = null$
2  $DT_Q = dependency\_parsing(Q)$
3  $rel = root = root\_word(DT_Q)$
4  $\mathcal{S} = \{w \in Q | w \xleftarrow{\texttt{nsubj,nsubjpass}} root\}$
5  foreach word $s \in \mathcal{S}$ do
6      if $qword \in descendants(s)$ then $sbj = ?x$
7      else $sbj =$ all words in $descendants(s)$
8      $Arg = \emptyset$
9      foreach $w \in children(root)$, $w \notin rel$, $w \notin sbj$ do
10         if $qword \in descendants(w)$ then
11             if $w$ is a preposition then $arg = w\ ?x$
12             else if $qword =$ "where" or "when" then
13                 $arg = in/on\ ?x$
14             else $arg = ?x$
15         else $arg =$ all words in $descendants(w)$
16         $Arg = Arg \bigcup \{arg\}$
17     $TQ = \langle sbj; rel; Arg\rangle$
18     $\mathcal{T}_Q = \mathcal{T}_Q \bigcup \{TQ\}$
19 $AnsType = detect\_answer\_type(Q)$
20 if $AnsType! = null$ then
21     foreach query $TQ_i \in \mathcal{T}_Q$ do
22         $TQ_i = [TQ_i \wedge \langle ?x;$ "is-a"$; AnsType\rangle]$
23 return $\mathcal{T}_Q$
```

SWERS. The dataset is large but noisy, containing many invalid *PTs*. Consider the following *PT*:
PT_2: What kind of money did they use in $*_{NP} \mapsto$
What is the economic system of $*_{NP}$
In [12], this problem is addressed by scoring each paraphrasing template using shallow features like its PMI value and Part-of-Speech (POS) tags. However, we found that using those shallow features alone is often inadequate. For example, PT_1 has identical POS tags as those of PT_2.

In this paper we propose using rich *association features* to identify high-quality question paraphrasing templates from noisy data. The intuition is that strong associations between elements in p_{src} and p_{tgt} often indicate good paraphrasing templates. For example, the association between *"money"* and *"national currency"* (PT_1, a good *PT*) is much stronger than that between *"money"* and *"economic system"* (PT_2, a bad *PT*). Although the idea of using association features has been explored in curated KB-QA systems [5] to rank generated queries, our work first applies it in the context of open KB-QA for measuring the quality of question paraphrasing.

Specifically, for each *PT*: $p_{src} \mapsto p_{tgt}$, we iterate through spans of text $s \in p_{src}$ and $t \in p_{tgt}$ to identify a set of *association phrases*, $\langle s, t\rangle$. For each $\langle s, t\rangle$, we generate a set of indicative association features using the feature templates listed in Table 3. These features are used in the answer ranker (Section 2.5) to measure the quality of the *PT*s used in deriving the answers. Following [5], we identify association phrases by looking-up a table that contains 1.3 million phrase pairs. For example, an indicator feature, $\mathbb{I}\{$*"money"* $\in p_{src} \wedge$ *"national currency"* $\in p_{tgt}\}$ is generated for the association phrases $\langle$*"money"*, *"national currency"*$\rangle$. During training, the answer ranker will learn each feature's weight based on its predictive power. Also, answers derived from high quality *PT*s will be given more credits. With our training data, the question paraphraser uses over 20,000 association features.

Besides the association features introduced above, we also use a set of statistical features such as the PMI value of the *PT*s, the log normalized frequency of p_{src} and p_{tgt}, etc. The full set of features used in paraphrasing is listed in Table 3.

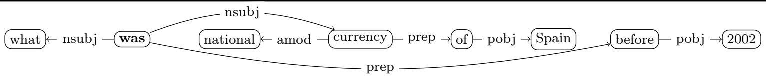
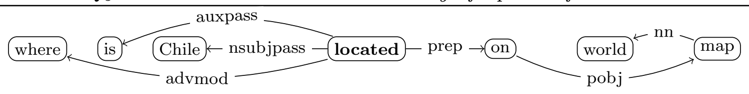
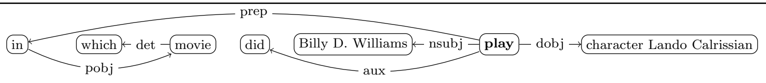
2.3 Question Parsing

Each paraphrased question Q is parsed into a set of structured tuple queries (TQs): $\mathcal{T}_Q = \{TQ_1, TQ_2, \ldots, TQ_K\}$. Each tuple query TQ_i has the form: $TQ_i = \langle sbj; rel; Arg\rangle$, which is similar to that of an n-tuple assertion (see Section 1). For a TQ, either the subject field *sbj* or one of the arguments in *Arg* is a wildcard (denoted by $?x$), which indicates the hidden answer the question Q seeks for. We call that wildcard, $?x$, the *query focus* of Q, denoted by F_Q.

Existing open KB-QA systems employ a small number of hand-crafted POS templates to parse questions into tuple queries [12], which are limited in coverage and can only handle questions *without* complex semantic constraints. In this paper we propose a novel approach of parsing questions by *dependency analysis*. This approach is partially inspired by recent advances in open IE [17, 8] research, where dependency parsing has been used to extract open IE assertions, but never to parse questions into tuple queries. Algorithm 1 shows our question parsing algorithm.

Given a question Q, TAQA first applies the Stanford parser to construct a *dependency tree*, DT_Q (Line 2). The nodes in DT_Q are made up of words in Q. The root of DT_Q is usually the main verb of Q and each edge in DT_Q denotes a pairwise dependency relation between two words in Q. For example, in the sentence: "*Bell makes electronic products*", the word *"Bell"* is a nominal subject (`nsubj`) of *"makes"*, and hence the dependency *"Bell"*$\xleftarrow{\texttt{nsubj}}$*"makes"* is derived. Table 2 shows three example questions (Q_5, Q_6, Q_7) and their respective dependency trees. Given a dependency tree, TAQA analyzes the tree to identify the various fields of a tuple query (i.e., *sbj*, *rel*, and the arguments in *Arg*) as follows:

Table 2: Examples of questions, dependency trees and generated tuple queries

question and dependency tree	tuple queries
 what nsubj was nsubj national amod currency prep of pobj Spain before pobj 2002 prep Q_5: *What was the national currency of Spain before 2002?*	TQ_1: ⟨*national currency of Spain; is; ?x, before 2002*⟩ TQ_2: ⟨*?x; is; national currency of Spain, before 2002*⟩ TQ_3(relaxed): ⟨*national currency of Spain; is; ?x*⟩ TQ_4(relaxed): ⟨*?x; is; national currency of Spain*⟩
 auxpass where is Chile nsubjpass located prep on world nn map advmod pobj Q_6: *Where is Chile located on world map?*	TQ_5: ⟨*Chile; is located; in/on ?x, on world map*⟩ TQ_6(relaxed): ⟨*Chile; is located; in/on ?x*⟩
 prep in which det movie did Billy D. Williams nsubj play dobj character Lando Calrissian pobj aux Q_7: *In which movie did Billy D. Williams play character Lando Calrissian?*	TQ_7: ⟨*Billy D. Williams; play; character Lando Calrissian, in ?x*⟩ ∧ ⟨*?x; is-a; movie*⟩ TQ_8(relaxed): ⟨*Billy D. Williams; play; in ?x*⟩ ∧⟨*?x; is-a; movie*⟩

[**Identify Relation Phrase**] (Line 3) The root word *root* of DT_Q is included in *rel* together with auxiliary words that are children of *root* and that satisfy certain dependency relations with *root*. These dependency relations include *copula* (`cop`), *auxiliary* (`aux`), *passive auxiliary* (`auxpass`), *phrasal verb particle* (`prt`), etc. For example, the relation phrase of question Q_6 in Table 2 is *"is located"* because *"located"* is the root verb and *"is"* is related to *"located"* by the dependency *"is"*$\xleftarrow{\texttt{auxpass}}$*"located"*.

[**Identify Subject**] (Lines 6-7) First, all nodes that are nominal subjects (`nsubj`) or passive nominal subjects (`nsubjpass`) of *root* are collected in a set $\mathcal{S}$. Then, for each node $s \in \mathcal{S}$, if the question word[3], *qword*, is a descendant of s, the subject *sbj* is the query focus *?x*. Otherwise, *sbj* includes all the descendants of s. Note that there could be multiple nominal subjects in DT_Q. In this case, we generate one tuple query for each `nsubj`. For example, question Q_5 in Table 2 derives 2 tuple queries TQ_1 and TQ_2.

[**Identify Arguments**] (Lines 9-16) Any child node w of *root* that is not identified as a subject (*sbj*) or does not belong to the relation phrase (*rel*) in the previous step derives an argument in *Arg*. Specifically, if the question word is a descendant of w, we create an argument that includes a query focus *?x*; otherwise, the argument includes all the descendants of w.

[**Compound Queries**] (Lines 19-22) For each tuple query TQ_i obtained from the previous steps, we augment it by deducing an *answer type*, *AnsType*, forming a compound conjunctive query $[TQ_i \wedge \langle ?x; \text{"}\textit{is-a}\text{"}; \textit{AnsType}\rangle]$. The purpose is to prune the search space for the answer (when the query is later executed against a KB) by restricting the scope of the query focus. For example, Question Q_7 in Table 2 derives the tuple query TQ_7. Note that the focus for this query can be restricted to the type *"movie"*, which is expressed by the augmenting tuple query ⟨*?x*; "*is-a*"; "*movie*"⟩. TAQA identifies the noun phrase between *qword* and the auxiliary[4] or root verb as *AnsType*.

[**Generate Relaxed Queries**] Finally, if the question Q is a complex question (with prepositional/adverbial constraints), TAQA will remove the complex constraints and additional objects of *root* (e.g., *"character Lando Calrissian"* in Q_7) and derives a *relaxed question* Q'. Algorithm 1 is then applied onto Q' to generate (relaxed) tuples queries. The reasons are twofold. First, some complex constraints are redundant and uninformative. For example, consider question Q_6 in Table 2. The constraint *"on world map"* is uninformative; Q_6 can be answered as well in its relaxed form. Second, using relaxed tuple queries can improve coverage, since there are likely assertions in the KB that do not mention the constraints given in the original question Q. TQ_6 in Table 2 is a relaxed tuple query of question Q_6. The use of relaxed tuple queries in answering questions is also illustrated in Figure 2 (see the right branch of the workflow).

2.4 Open KB Querying

After obtaining the set of tuple queries $\mathcal{T}_Q$, the next task is to execute each query $TQ \in \mathcal{T}_Q$ against the open KB to obtain a set of candidate answers $\mathcal{A}_{TQ}$. This is carried out in two steps. First, a set of assertions, $\mathcal{R}_{TQ}$, in the open KB that are relevant to the query TQ is retrieved. In TAQA, assertions are indexed by Apache Solr[5], which retrieves a ranked list of assertions based on the terms mentioned in the assertions and those in the tuple query. We refine $\mathcal{R}_{TQ}$ by removing assertions that have fewer fields than TQ, and then retaining the top-50 of the remaining ones.

Next, we need to align the fields of TQ against those of each assertion $r \in \mathcal{R}_{TQ}$. This is done to identify the field in r that matches the query focus F_Q, which is then taken as the answer of TQ. Unlike OQA, for which queries and assertions are both triples, TAQA deals with n-tuple queries and assertions with an arbitrary number of arguments. The multiplicity of *Arg* fields involves a complex alignment problem. We put forward a novel answer extraction approach which models the alignment problem as a matching problem on a weighted bipartite graph. Figure 3 illustrates the alignment between a tuple query TQ_7 (mentioned in Table 2) against the assertion r : ⟨"*Billy Dee Williams*"; ...⟩ (shown in the figure). Specifically, in the bipartite graph G, fields in TQ form one type of nodes while fields in r form another. Let $TQ.i$ and $r.j$ denote the i-th field of TQ and the j-th field of r, respectively. The weight of the edge connecting $TQ.i$ and $r.j$ in G is given by the following similarity measure:

$$\begin{aligned}\textit{similarity}(TQ.i, r.j) &= \alpha \cdot \textit{text\_similarity}(TQ.i, r.j)\\ &\quad + (1-\alpha)\cdot \textit{pattern\_similarity}(TQ.i, r.j)\end{aligned}$$

where α is a tuning weight ($\alpha = 0.5$ in the experiment). *text_similarity* is the TF-IDF score of two lemmatized strings. *pattern_similarity* is the sum of three indicator functions that utilize POS/prefix/question patterns:

- $\mathbb{I}_1 = 1$ iff $(TQ.i \text{ is } ?x) \wedge (r.j \text{ is a noun phrase})$.

[3]Question words are "*what*", "*who*", "*where*", "*when*" and "*which*".
[4]Auxiliary verbs include "*do*", "*did*", "*does*", "*is*", "*was*", "*are*", etc.

[5]`http://lucene.apache.org/solr/`

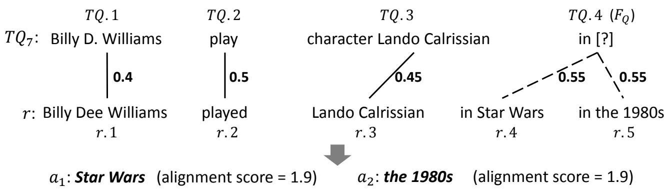


Figure 3: n-tuple alignment

- $\mathbb{I}_2 = 1$ iff ($qword$ is *"when"*) $\wedge$ ($TQ.i$ is *"in/on" ?x*) $\wedge$ ($r.j$ is of the pattern *"in/on date_time_string"*).
- $\mathbb{I}_3 = 1$ iff ($TQ.i$ is F_Q) $\wedge$ (F_Q and $r.j$ start with the same preposition).

We then seek all optimal matchings for the following constrained optimization:

$$\max : \sum_i \sum_j x_{ij} \cdot \mathit{similarity}(TQ.i, r.j)$$

$$\text{subject to: } (x_{11} = 1),\ \ (x_{22} = 1),\ \ (x_{ij} \in \{0,1\})$$
$$(0 \le \sum_i x_{ij} \le 1),\quad (\sum_j x_{ij} = 1). \qquad (1)$$

where $x_{ij} = 1$ iff $TQ.i$ is aligned to $r.j$. The constraints $x_{11} = 1$ forces the alignment between two *sbj* fields. Likewise with $x_{22} = 1$ for the *rel* fields. The field of r that is aligned to the query focus field of TQ in each optimal matching is added to $\mathcal{A}_{TQ}$ as an answer. As an example, in Figure 3, there are two optimal matchings (the query focus field has equal weights with "*in Star Wars*" and "*in the 1980s*"), deriving two answers (a_1 and a_2).

The tuple query used in the illustration (Figure 3) is actually part of a compound query (TQ_7 in Table 2). For such cases, TAQA verifies the answer by querying the KB using the other part of the query. For example, TAQA evaluates the tuples ⟨"*Star Wars*"; "*is-a*"; "*movie*"⟩ and ⟨"*the 1980s*"; "*is-a*"; "*movie*"⟩ against the KB. Answer a_2 is eliminated because it is not supported by the assertions in the KB.

2.5 Answer Ranking

Generally, a question Q is paraphrased into multiple questions, each such question is parsed into multiple tuple queries and each tuple query can derive multiple answers. TAQA collects all these candidate answers into an answer set $\mathcal{A}$. The final stage of the QA process is to determine the best answer in $\mathcal{A}$. An answer $a \in \mathcal{A}$ is associated with many features as it is derived along the paraphrasing-parsing-querying process. We use $\boldsymbol{f}(a) = [f_1(a), f_2(a), \ldots, f_M(a)]$ to denote the feature vector of a. TAQA uses these features to rank the candidate answers. Table 3 lists the features used in TAQA.

The answer set $\mathcal{A}$ is first *consolidated.* In many cases, the same answers are derived via different paraphrasing-parsing-querying outputs. These answers have the same string value but different feature vectors, which should be combined. Specifically, if a_i, $a_j \in \mathcal{A}$ are same answers, then their feature vectors are combined by taking the *best* value for each individual feature. For example, if f_k represents the alignment score (Equation 1) of the assertion that derives an answer, then the combined feature value is $\max(f_k(a_i), f_k(a_j))$ because a higher alignment score indicates better answer quality. This simple approach of consolidating features has been proven useful in [15]. Furthermore, we add statistical features that measure the "popularity" of answers. These features include the occurrence frequency of an answer in $\mathcal{A}$, and the answer's n-gram correlation defined in [9].

KB	# assertions	# relation phrases
OPEN IE	458M	6M
PROBASE	170M	1
NELL	2M	300
nOKB	120M	4.6M

Table 4: Statistics of nOKB+ used by TAQA

WebQuestions	
▷ who played jacob black in twilight?	Taylor Lautner
▷ what timezone is nashville tn?	Central Time Zone
▷ who did roger federer married?	Mirka Federer
ComplexQuestions	
▷ what team did shaq play for before the lakers?	Orlando Magic
▷ what country gained its independence from britain in 1960?	Cyprus
▷ what did france lose to the british in the treaty of paris in 1763?	Dominica

Table 5: Sample questions from WEBQ and COMPQ

After consolidation, we apply a standard log-linear model to estimate the probability $p(a|Q)$ for each $a \in \mathcal{A}$:

$$p(a|Q) = \frac{exp\{\sum_{k=1}^{M} \lambda_k \cdot f_k(a)\}}{\sum_{a' \in \mathcal{A}} exp\{\sum_{k=1}^{M} \lambda_k \cdot f_k(a')\}}$$

where λ_k denotes the k-th feature's weight, and rank candidate answers accordingly. We limit the size of $\mathcal{A}$ to the top 200 candidates when calculating $p(a|Q)$.

We tune the feature weights $\boldsymbol{\lambda} = \{\lambda_k\}$ using a set of N question and gold-standard answer pairs $\mathcal{D} = \{(Q_t, a_t)\}$, by maximizing the log-likelihood of the training data:

$$\mathcal{L}(\mathcal{D}; \boldsymbol{\lambda}) = \sum_{t=1}^{N} \log p(a_t|Q_t; \boldsymbol{\lambda})$$

TAQA employs AdaGrad [10] to effectively learn the weights of over 20,000 sparse features.

3. EXPERIMENT

In this section we evaluate TAQA. We primarily address the following two questions: (1) How does TAQA's performance compare with the state-of-the-art open KB-QA and curated KB-QA systems, especially in answering complex questions? (2) How do the various components of TAQA impact the system's performance? We will first describe the open KB TAQA uses, the evaluating question sets, the performance metrics, and two other KB-QA systems, namely, OQA and PARASEMPRE, before presenting the results. We also present insightful case study in Section 4.

[Open KBs] There are three well known open KBs, namely, OPEN IE [12], PROBASE [19] and NELL [7]. OPEN IE is a large open KB built by performing open IE on CLUEWEB. PROBASE is an open KB of `is-a` assertions extracted from 1.68B web pages. NELL is a relatively small open KB which contains highly precise assertions. Assertions in the above three KBs are all triples. As we have explained, n-tuple assertions are richer in semantics, and are more suitable for answering complex questions. Hence, we build a new open KB of n-tuple assertions (called nOKB) using the latest open IE technique. Here, we briefly discuss how nOKB was built.

First, we need to collect a set of documents $\mathcal{D}$ from which assertions are extracted. We collect all English Wikipedia

Table 3: List of features used in TAQA

Question Paraphrasing, association feature templates ($\langle s, t\rangle$: association phrases)		
▷ $\mathbb{I}\{lemma(s) \in p_{src} \wedge lemma(t) \in p_{tgt}\}$, $lemma(\cdot)$ is the lemmatised text	▷ $\mathbb{I}\{pos(s) \in p_{src} \wedge pos(t) \in p_{tgt}\}$, $pos(\cdot)$ is POS tags	
▷ $\mathbb{I}\{lemma(s)$ and $lemma(t)$ are synonyms$\}$	▷ $\mathbb{I}\{lemma(s)$ and $lemma(t)$ are word-net derivations$\}$ [5]	
Question Paraphrasing, other features (Q_i: paraphrased question, p_{src}: source template, p_{tgt}: target template)		
▷ $\mathbb{I}\{Q_i$ is the original question$\}$	▷ $PMI(p_{src} \mapsto p_{tgt})$, PMI value of the paraphrase template	
▷ $\lvert$words captured by $*_{NP}\rvert / \lvert p_{src}\rvert$	▷ log normalized frequency of p_{src} and p_{tgt}	
Question Parsing (TQ: tuple query)		
▷ $\mathbb{I}\{TQ$ is relaxed query$\}$	▷ $\mathbb{I}\{TQ$ is not relaxed query$\}$	
Open KB Querying (r: assertion, a: candidate answer)		
▷ alignment score of TQ and r	▷ word count of the answer a	▷ ratio of capital words in a
▷ the average IDF value of words in a	▷ $\mathbb{I}\{a$ in a predefined entity list$\}$	▷ $\mathbb{I}\{a$ is a date time $\wedge$ $qword$ = *"when"*$\}$
▷ cosine similarity of fields in r and TQ	▷ extraction frequency and confidence of r	▷ the rank of $r \in \mathcal{R}_{TQ}$
▷ the source of r (cf. Table 4)	▷ $\mathbb{I}\{TQ$ is join query$\}$	▷ cosine similarity of join keys
Answer Ranking ($\mathcal{A}$: candidate answers set)		
▷ n-gram correlation score of a [9]	▷ occurrence frequency of a in $\mathcal{A}$	

documents into a set $\mathcal{D}_{wiki}$. Also, for each question Q in an evaluation question set $\mathcal{QS}$ (we will explain shortly how $\mathcal{QS}$ is obtained), we submit Q to a commercial search engine and retrieve top-200 documents. The retrieved documents for all the questions form the set $\mathcal{D}_{\mathcal{QS}}$. We get $\mathcal{D} = \mathcal{D}_{wiki} \cup \mathcal{D}_{\mathcal{QS}}$, which contains 5.7M web pages. We use OpenNLP[6] to extract sentences from documents, and filter out non-English sentences and those with more than 100 words. After that, we apply Open IE 4.1[7] to the sentences to obtain n-tuple assertions. We remove assertions whose subject or argument fields are either stop words or more than 10 words in length. The process results in 163M sentences and 120M n-tuple assertions. These 120M assertions form nOKB. TAQA operates on assertions provided by all 4 open KBs. We denote the combined KB, nOKB+. Table 4 summarizes these KBs.

[Question Sets] The experiments were conducted on two sets of questions, WebQuestions and ComplexQuestions. Each question set is a collection of (question, gold-standard-answer) pairs. Table 5 gives example pairs taken from the two sets.

WebQuestions [4] (abbrev. WEBQ) is the de facto question set used to evaluate curated KB-QA systems. It consists of 5,810 question-answer pairs. Questions are collected by the Google Suggest API and manually answered by AMT workers using Freebase. WEBQ is moderated for curated KB-QA systems in that all questions are answerable with Freebase (a curated KB). We observe that most of the questions in WEBQ are simple questions; only 80 (4%) of the questions in the test set of WEBQ (2,032 questions) come with complex semantic constraints.

ComplexQuestions (abbrev. COMPQ) is a new dataset created in this work, which focuses on open domain complex questions with prepositional/adverbial constraints. COMPQ consists of 300 complex questions obtained as follows: First, all 80 complex questions in WEBQ are added to COMPQ. Next, we follow the approach given in [4] to crawl questions using the Google Suggest API. We start with a seed question Q^*: *"Who did Tom Hanks play in Toy Story?"*. We google Q^* to obtain at most 10 suggested questions. We retain those suggested questions that begin with a *wh*-word and which contain at least one preposition. For each such question, we remove the entities mentioned in the question and submit the resulting string to google to obtain more suggested questions. The process is repeated in a breadth-first manner. Due to space limitation, readers are referred to [4] for the details of the procedure. We collected a total of 100,000 questions, and then randomly picked 220 complex questions out of the pool. Of all the questions in COMPQ, 20 of them have either 2 or 3 complex constraints, the rest have 1 complex constraint each.

[Metrics] WEBQ comes with a set of answers that are labeled as correct or incorrect [12]. For COMPQ, we manually label the answers returned by the QA systems we tested. Given a question, each QA system computes a ranked list of answers and returns the top-ranked one as the final answer. We measure the performance of a system by its accuracy:

$$acc = \frac{\text{number of correctly answered questions}}{\text{total number of questions}}.$$

[QA systems] We compare TAQA against the current state-of-the-art KB-QA systems: OQA [12], which is the latest open KB-QA system based on triple-form assertions, and PARASEMPRE [5], which is an advanced curated KB-QA system powered by Freebase. We use nOKB+ as the open KB for both OQA and TAQA. For OQA and PARASEMPRE, we use the trained models provided by their authors.[8] For TAQA, we train its feature weights $\boldsymbol{\lambda}$ on nOKB+ using the standard training set of WEBQ (3,778 questions).

3.1 Results

Table 6 shows the accuracy (acc) of the the three KB-QA systems when presented questions from the two question sets. Readers are reminded that PARASEMPRE uses a different KB (Freebase) from that of OQA and TAQA (nOKB+). The performance numbers of PARASEMPRE should not be compared directly with those of the other two systems. Nonetheless, these numbers provide interesting references.

We first compare OQA and TAQA in answering complex questions (COMPQ). From Table 6, we see that OQA had a hard time handling complex questions with an accuracy of only 2%. We studied how OQA handled each of the 300 questions in COMPQ. We make the following observations about its poor performance. First, OQA relies on a limited number of hand-crafted POS templates for question parsing. These templates are inadequate in parsing questions with complex semantic constraints (see Section 2.3). In fact, only about 10% of the questions in COMPQ are successfully parsed by OQA into tuple queries; the other 90% do not result in any tuple queries and thus are not answered by OQA. In sharp contrast, TAQA employs dependency analysis in ques-

[6]http://opennlp.apache.org/

[7]http://knowitall.github.io/openie/

[8]We also train OQA on nOKB+, but the provided model gives better results.

Systems	ComplexQuestions	WebQuestions
OQA	2.0%	22.9%
TAQA	**39.3%**	35.6%
PARASEMPRE	9.7%	**45.8%**

Table 6: Accuracies (*acc*) of systems

tion parsing, which allows it to successfully parse 98% of the questions. Second, OQA was designed to handle triplets assertions. In particular, only the first three fields (*sbj*, *rel*, and the first argument in *Arg*) of assertions in nOKB+ are used. OQA is therefore unable to fully utilize the rich semantic information that is often displayed in the additional argument fields of n-tuple assertions. TAQA, on the other hand, employs a matching approach to align complex tuple queries to n-tuple assertions. This allows TAQA to make full use of the assertions' argument fields, resulting in a very respectable accuracy of 39.3%.

We also used PARASEMPRE to answer COMPQ questions and registered a 9.7% accuracy. Given a question Q, PARASEMPRE first parses Q into a number of SPARQL queries. It then evaluates the queries and executes the top-ranked one against Freebase. We found that due to the complexity of questions in COMPQ, PARASEMPRE generates an average of 2,090 candidate queries for each question in COMPQ, in an attempt to cover all potential answers. It turns out that most of these queries are *false queries*, which lead to incorrect answers. The big pool of queries makes it very difficult for PARASEMPRE to identify the best query to obtain the correct answers with any less-than-perfect ranking method. In contrast, TAQA generates, on average, 25 tuple queries for each question in COMPQ. The small query pool makes it easier to identify (rank) the best query for a correct answer. Moreover, many of the SPARQL queries generated by PARASEMPRE fail in capturing the questions' complex constraints. For example, the query PARASEMPRE executed for the question: *"What character did Anna Kendrick play in Twilight?"* is without the underlined constraint. In contrast, for the same question, TAQA executes the tuple query ⟨"*Anna Kendrick*"; "*play*"; *?x*, "*in Twilight*"⟩∧⟨*?x*; "*is-a*"; "*character*"⟩ and extracts the correct answer from assertion ⟨"*Anna Kendrick*"; "*plays*"; "*Jessica Stanley*", "*in Twilight*"⟩.

Another reason why PARASEMPRE gives a relatively low accuracy for COMPQ questions is the restricted coverage of Freebase (a curated KB). For example, PARASEMPRE fails to answer the question *"When did Canada gain independence from England?"* because Freebase does not contain the factual knowledge about the independence of countries.

Next, we consider questions from WEBQ. Recall that questions in WEBQ are predominately simple questions and that all are answerable with information on Freebase. PARASEMPRE is very effective in answering them, which is reflected by the very high (45.8%) accuracy.

Because of the simplicity of WEBQ questions, OQA does not encounter much problem in parsing questions or executing queries in triples form. Its gives a decent accuracy of 22.9% with WEBQ. Despite the fact that simple questions do not necessarily require the rich information provided by the additional argument fields in n-tuple assertions (which TAQA excels in manipulating), TAQA still outperforms OQA significantly (with an accuracy of 35.6%). Besides the sophisticated techniques TAQA employs in the paraphrasing, parsing, querying, and ranking, we found that another factor that contributes to the gap is the difference in the style of the queries generated by the two systems. The POS templates used in OQA for question parsing is designed for generating REVERB style queries [11] (e.g., ⟨*?x*; "*is national currency of*"; "*Spain*"⟩) instead of OPENIE style queries (e.g., ⟨*?x*; "*is*"; "*national currency of Spain*"⟩), which are generated by the tool we use to build nOKB. In other words, assertions in nOKB, which are used in our experiment, match the tuple queries generated by TAQA better. This facilitates query execution and hence gives a higher accuracy. This observation leads to an interesting question of how open IE assertion styles impact QA systems, which we leave as a future work.

Systems	ComplexQuestions	WebQuestions
TAQA (Full model)	39.3%	35.6%
No complex constraints	29.3% (-10.0%)	34.0% (-1.6%)
No relaxed queries	27.7% (-11.6%)	32.3% (-3.3%)
No association features	35.3% (-4.0%)	30.3% (-5.3%)
No answer consolidation	32.3% (-7.0%)	26.3% (-9.3%)

Table 7: Component ablation test for TAQA (*acc*)

TAQA's performance on WEBQ is slightly lower than that on COMPQ. We note that this could be attributed to the fact that nOKB+ does not have adequate knowledge to support answering some questions in WEBQ that are tailored for Freebase. For example, TAQA makes a constant error in answering questions about tourist attractions like *"What to see in Paris?"*, a category of question that is not well covered by nOKB+. In contrast, PARASEMPRE can easily answer these questions using `tourist_attractions` relation in Freebase.

TAQA employs a number of techniques along the question-answering process. Our next set of experiments is to evaluate the effectiveness of each component. This is done by removing these components one at a time and observing the accuracy of the resulting system. Table 7 shows the results.

In the first test (labeled "No complex constraints"), we stripped all complex constraints (e.g., *"before 2002"*) from the tuple queries generated by the question parser. Each *TQ* is thus in the basic triple form. For COMPQ questions, we see that the accuracy drops by 10% points. This shows that the ability of TAQA's parser in composing queries in the n-tuple form has a significant impact in answering complex questions. One might expect *acc* to drop even lower than the 29.3% shown, considering that stripping the complex constraints would "destroy" the questions in COMPQ, which are all complex questions. The answer to this is twofold. First, sometimes complex constraints in questions are redundant and the questions can be answered by simpler triple queries. An example is *"Who plays Bilbo Baggins in The Hobbit?"*, which can be answered by the triple query ⟨*?x*; "*plays*"; "*bilbo baggins*"⟩. Second, for cases where dropping the constraints causes ambiguity in answers, sometimes, the correct answers can still be got simply because they happen to be the most popular answers among the assertions in the KB. For questions in WEBQ, stripping complex constraints had little effects, since questions are predominately simple questions.

In question parsing, relaxed queries (those with complex constraints removed) are generated in addition to the regular ones. The objective is to improve the coverage of queries (Section 2.3). In our next test, we suppressed relaxed query generation (labeled "No relaxed queries"). For COMPQ, we see an accuracy drop of 11.6% points, which is even more than that in the first test. This shows that considering all the constraints in questions (no relaxed queries) is no bet-

Rank:	1	2-5	6-10	11-20	>20
ComplexQuestions	59.9%	19.8%	8.1%	4.1%	8.1%
WebQuestions	62.7%	15.8%	6.4%	4.9%	10.2%

Table 8: Rank distribution of correct answers

ter than ignoring all of them (no constraints); the coverage of queries is indeed an important factor. TAQA (full model) is able to combine the scattered evidence of answers extracted from both relaxed and "complete" queries via answer consolidation (Section 2.5), which boosts the rank of the answers that are derived from both forms of queries. An example question is *"What book did Charles Darwin wrote in 1859?"*, which has a relevant assertion A: ⟨"*Charles Darwin*"; "*wrote*"; "*On the Origin of Species*", "*in 1859*"⟩. Without relaxed queries, the correct answer was ranked only second because A has a low frequency in the KB. The full model correctly ranked the answer top-1 because it got more evidence for the answer from the relaxed query ⟨"*Charles Darwin*"; "*wrote*"; *?x*⟩ which is supported by a frequent assertion ⟨"*Darwin*"; "*wrote the book*"; "*On the Origin of Species*"⟩.

While OQA uses shallow features in question paraphrasing, TAQA uses 20,000 association features in addition to shallow statistical features (Section 2.2). Our next test removed all association features (labeled "No association features"). We observed an accuracy drop of 4.0% and 5.3% for COMPQ and WEBQ, respectively. This shows that association features are very useful in paraphrasing. Finally, TAQA performs answer consolidation in the answer ranking step (Section 2.5) — the same answers (but with different feature vectors) are consolidated into one by combining their feature vectors. The last test turned off answer consolidation (labeled "No answer consolidation"). In this case, we see significant drops in accuracy of 7.0% and 9.3% for COMPQ and WEBQ, respectively. Answer consolidation, which allows TAQA to ensemble scattered evidence in the KB for more confidence in answers, is seen as an important technique.

Our last experiment investigates the effectiveness of TAQA's answer ranker (Section 2.5). We say that a question Q is *answerable* by TAQA if it's correct answer appears in the ranked list of candidate answers generated for Q. Intuitively, for an answerable question, TAQA is capable of identifying the correct answer as a candidate. An effective ranker would rank the correct answer first in the list so that it is returned as the final answer. In our experiment, 65.7% of COMPQ questions and 56.8% of WEBQ questions are answerable. For those answerable questions, Table 8 shows the distribution of their ranks. From the table, we see that TAQA's ranker is very effective; the distribution is very skewed towards the top ranks. In particular, for both COMPQ and WEBQ questions, around 60% of correct answers were ranked top-1 and around 80% of them were ranked top-5. The distributions shown in Table 8 is indeed insightful. For example, for COMPQ, 19.8% of correct answers were ranked top-2-to-5, which are about 1/3 of those top-1 ones (59.9%). These correct answers were so close to be picked as the final answers. If we can further improve TAQA's ranker such that all these correct answers are promoted to top-1, we would have effectively improve the accuracy of TAQA by 1/3. This discussion indicates that more effort should be spent on further improving the ranker.

4. CASE STUDY

In this section we present a few representative questions that illustrate successful and failed cases of TAQA (Table 9). Through these cases, we obtain insights into the key components of a good KB-QA system.

	Example 1
Question:	Q_9: What movies did Morgan Freeman star in?
Paraphrased:	Q'_9: What movies did Morgan Freeman play in?
	Q''_9: What movies did Morgan Freeman star in?
Queries:	TQ'_9: ⟨*Morgan Freeman; play; in ?x*⟩∧⟨*?x; is-a; movie*⟩
	TQ''_9: ⟨*Morgan Freeman; star; in ?x*⟩∧⟨*?x; is-a; movie*⟩
Assertions:	A_9: ⟨*Morgan Freeman; played; god, in Bruce Almighty*⟩
	A_{10}: ⟨*Morgan Freeman; starred; in Bruce Almighty*⟩
Answer:	**Bruce Almighty**
	Example 2
Question:	Q_{10}: Where did Barack Obama go to college in 1991?
Paraphrased:	Q'_{10}: Where did Barack Obama graduate from in 1991?
	Q''_{10}: Which university did Barack Obama attend in 1991?
Queries:	TQ'_{10}: ⟨*Barack Obama; graduate; from ?x, in 1991*⟩
	TQ''_{10}: ⟨*Barack Obama; attend; ?x*⟩∧⟨*?x; is-a; university*⟩
Assertions:	A_{11}: ⟨Barack Obama; graduated; *from Harvard Law School, in 1979 and 1991*⟩
	A_{12}: ⟨*Barack Obama; graduated; magna cum laude, from Harvard Law School, in 1991*⟩
	A_{13}: ⟨*Barack Obama; attended; Harvard Law School*⟩
Answer:	**Harvard Law School**
	Example 3
Question:	Q_{11}: Where Turkish people originate?
Paraphrased:	Q'_{11}: Where is Turkish people origin?
Queries:	TQ'_{11}: ⟨*Turkish people; is; origin, in ?x*⟩
Assertions:	A_{14}: ⟨*many Turkish Alevi; are; of Arab origin, in Syria*⟩
Answer:	**Syria**
	Example 4
Question:	Q_{12}: Who was the father of King George VI?
Queries:	TQ_{12}: ⟨*?x; was; father of King George VI*⟩
Assertions:	A_{15}: ⟨*Queen Elizabeth; was; wife of King George VI*⟩ (extracted from 10 relevant assertions)
	A_{16}: ⟨*George V; was; father of King George VI*⟩ (extracted from 1 relevant assertion)
Answer:	**Queen Elizabeth**

Table 9: Examples of successful and failed cases

Open KBs are noisy and unnormalized. A challenging problem is to bridge the lexical/syntactic gap between input questions and relevant assertions. When this gap is small, TAQA is very successful in finding the correct answer. Example 1 shows such a case. Question Q_9 is paraphrased into Q'_9 and Q''_9, which match perfectly the assertions A_9 and A_{10} in the KB, respectively. This results in a correct answer. When the lexical gap is large, paraphrasing is our hope in narrowing the gap. Example 2 shows such a case. Question Q_{10} uses informal expressions, which shares little lexical overlap with the relevant assertions (A_{11}, A_{12}, A_{13}). In this example, TAQA successfully paraphrases Q_{10} into formalized ones (Q'_{10}, Q''_{10}), which are parsed into tuple queries (TQ'_{10}, TQ''_{10}) that perfectly match the assertions.

While good paraphrasing can help bridge the lexical gaps, poor paraphrasing can adversely affect question parsing. TAQA's parser employs dependency analysis, which requires questions to follow correct grammatical rules. If the paraphrased questions are grammatically incorrect, the parsed tuple queries will be erroneous, resulting in incorrect answers. Example 3 shows such a case. Question Q_{11} is paraphrased as a grammatically ill-formed question (Q'_{11}), which is parsed into a malformed dependency tree, resulting in a meaningless tuple query (TQ'_{11}) and an incorrect answer.

Similarity measures (see Section 2.4), which are used to evaluate the relatedness between tuple queries and assertions play an important role in filtering incorrect candidate answers. In the current implementation, TAQA primarily employs cosine similarity on text, which is insufficient to measure the semantic relatedness between queries and assertions. This is illustrated by Example 4. For Question Q_{12}, TAQA returns the incorrect answer *"Queen Elizabeth"* instead of the correct one *"King George V"*. The reason is that although assertion A_{16} gives a larger cosine similarity with the tuple query TQ_{12} than does A_{15}, the difference in the similarity scores is marginal. The higher popularity of A_{15} in the KB over-offsets the similarity difference, giving *"Queen Elizabeth"* a higher rank. To improve, one needs to look into similarity measures that can discover deep semantic relatedness between short-texts, which has been recognized as a fundamental task in AI.

5. RELATED WORKS

There are two lines of research in KB-QA systems. One focuses on curated KBs and the key challenge is to transform the lexicon of NL questions to structured KB queries. SEMPRE [4] utilizes a set of REVERB extractions to map NL phrases to KB relations. Kwiatkowski et al. [16] proposed to learn intermediate semantic representations directly from NL questions, with which KB queries are derived on the fly using WIKTIONARY as synonymy features. Xu et al. [20] proposed to learn an alignment model between NL phrases and Freebase relations from large web corpora. They approached the task of finding corresponding Freebase relations given NL phrases via classification. Recently, Berant et al. [5] proposed PARASEMPRE, which applies an "overgenerate and re-rank" strategy. The strategy is to enumerate possible KB queries and transform them to synthetic NL questions. They also applied a paraphrase model with rich association and embedding features to rank candidate queries based on the similarity between the input and the synthesized questions.

Another line of research focuses on open KB-QA. PARALEX [13] is the first open KB-QA system, which is based on learned lexicons from a corpus of question paraphrases. More recently, Fader et al. proposed OQA [12], a unified framework for open domain question answering on curated and open KBs. OQA processes questions using a cascaded pipeline of paraphrasing, question parsing, query rewriting and execution. Our work is similar with OQA in decomposing QA into different sub-components. However, we put forward a number of techniques which are essential in processing complex questions.

6. CONCLUSION AND FUTURE WORK

In this paper we presented a novel open KB-QA system, TAQA, that is able to answer questions with complex semantic constraints using n-tuple open KB, nOKB. We proposed new question parsing and answer extraction methods that are adapted to n-tuple assertions. Empirical evaluation showed that our system significantly outperformed state-of-the-art curated KB-QA and open KB-QA systems in answering questions with rich semantic constraints.

Through discussions, we have identified a number of key issues that have strong implications to the performance of TAQA. These include (1) question parsing that is robust to grammatical errors in questions, (2) similarity measures that can capture the deep semantic relatedness between queries and assertions, (3) answer ranking that can leverage more powerful features. In addition, efficient methods for cleaning and canonicalizing n-tuple assertions would greatly improve the quality of the open KB, and thus the QA accuracy.

Finally, we remark that TAQA and PARASEMPRE excel in handling complex questions and simple questions, respectively. In practice, user-issued questions can be of either kind. A natural extension to our work for handling a mixed question workload is ensembling. As an example, we trained an ensemble model using RankBoost with simple statistical features to re-rank the combined top-10 outputs from TAQA and PARASEMPRE. On a mixed question set of 30% complex questions and 70% simple questions, the ensemble model achieves an accuracy of 46.6%, which is better than running TAQA on COMPQ (39.3%) or PARASEMPRE on WEBQ (45.8%). This shows that ensembling is a promising approach for further explore.

Acknowledgments

This research is supported by Hong Kong Research Grants Council GRF grant HKU712712E. We would like to thank the anonymous reviewers for their insightful comments.

7. REFERENCES

[1] S. Auer et al. Dbpedia: A nucleus for a web of open data. In *ISWC/ASWC*, 2007.
[2] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. Open information extraction from the web. In *IJCAI*, 2007.
[3] J. Bao et al. Knowledge-based question answering as machine translation. In *ACL (1)*, 2014.
[4] J. Berant, A. Chou, R. Frostig, and P. Liang. Semantic parsing on freebase from question-answer pairs. In *EMNLP*, 2013.
[5] J. Berant and P. Liang. Semantic parsing via paraphrasing. In *ACL (1)*, 2014.
[6] K. D. Bollacker and C. Evans et al. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*, 2008.
[7] A. Carlson et al. Toward an architecture for never-ending language learning. In *AAAI*, 2010.
[8] L. D. Corro et al. Clausie: clause-based open information extraction. In *WWW*, 2013.
[9] N. Duan. Minimum bayes risk based answer re-ranking for question answering. In *ACL (2)*, 2013.
[10] J. C. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. In *JMLR*, 2011.
[11] A. Fader, S. Soderland, and O. Etzioni. Identifying relations for open information extraction. In *EMNLP*, 2011.
[12] A. Fader, L. Zettlemoyer, and O. Etzioni. Open Question Answering Over Curated and Extracted Knowledge Bases. In *KDD*, 2014.
[13] A. Fader, L. S. Zettlemoyer, and O. Etzioni. Paraphrase-driven learning for open question answering. In *ACL (1)*, 2013.
[14] L. Galarraga, G. Heitz, K. Murphy, and F. M. Suchanek. Canonicalizing open knowledge bases. In *CIKM*, 2014.
[15] D. Gondek and A. Lally et al. A framework for merging and ranking of answers in deepqa. *IBM Journal of Research and Development*, 2012.
[16] T. Kwiatkowski et al. Scaling semantic parsers with on-the-fly ontology matching. In *EMNLP*, 2013.
[17] Mausam, M. Schmitz, S. Soderland, R. Bart, and O. Etzioni. Open language learning for information extraction. In *EMNLP-CoNLL*, 2012.
[18] Y. Mohamed. Natural language questions for the web of data. In *EMNLP-CoNLL*, 2012.
[19] W. Wu et al. Probase: a probabilistic taxonomy for text understanding. In *SIGMOD*, 2012.
[20] X. Yao and B. V. Durme. Information extraction over structured data: Question answering with freebase. In *ACL (1)*, 2014.