

On Generating Characteristic-rich Question Sets for QA Evaluation

Yu Su¹, Huan Sun², Brian Sadler³, Mudhakar Srivatsa⁴

Izzeddin Gür¹, Zenghui Yan¹ and Xifeng Yan¹

¹University of California, Santa Barbara, Department of Computer Science

²The Ohio State University, Department of Computer Science and Engineering

³U.S. Army Research Lab, ⁴IBM Research

{ysu, izzeddingur, zyan, xyan}@cs.ucsb.edu, sun.397@osu.edu
brian.m.sadler6.civ@mail.mil, msrivats@us.ibm.com

Abstract

We **present** a semi-automated framework for constructing factoid question answering (QA) datasets, where an array of question characteristics are formalized, including **structure complexity**, function, commonness, answer cardinality, and paraphrasing. Instead of collecting questions and manually characterizing them, we employ a reverse procedure, **first** generating a kind of graph-structured logical forms from a knowledge base, and **then** converting them into questions. Our work is the first to generate questions with explicitly specified characteristics for QA evaluation. We construct a new QA dataset with over 5,000 logical form-question pairs, associated with answers from the knowledge base, and show that datasets constructed in this way enable fine-grained analyses of QA systems. The dataset can be found in <https://github.com/ysu1989/GraphQuestions>.

1 Introduction

Factoid question answering (QA) has gained great attention recently, owing to the fast growth of large knowledge bases (KBs) such as DBpedia (Lehmann et al., 2014) and Freebase (Bollacker et al., 2008), which avail QA systems of comprehensive and precise knowledge of encyclopedic scope (Yahya et al., 2012; Berant et al., 2013; Cai and Yates, 2013; Kwiatkowski et al., 2013; Berant and Liang, 2014; Fader et al., 2014; Reddy et al., 2014; Bao et al., 2014; Zou et al., 2014; Yao and Van Durme, 2014; Yih et al., 2015; Sun et al., 2015; Dong et al., 2015; Yao, 2015; Berant and Liang, 2015). With the blossoming of QA systems, evaluation is becoming an

increasingly important problem. QA datasets, consisting of a set of questions with ground-truth answers, are critical for both comparing existing systems and gaining insights to develop new systems.

Questions have rich *characteristics*, constituting dimensions along which question difficulty varies. Some questions are difficult due to their complex **semantic structure** (“Who was the coach when Michael Jordan stopped playing for the Chicago Bulls?”), while some others may be difficult because they require a precise **quantitative analysis** over the answer space (“What is the best-selling smartphone in 2015?”). Many other characteristics shall be considered too, e.g., what **topic** a question is about (questions about common topics may be easier to answer) and how many answers there are (it is harder to achieve a high recall in case of multiple answers). Worse still, due to the flexibility of natural language, different people often describe the same question in different ways, i.e., **paraphrasing**. It is important for a QA system to be robust to paraphrasing.

A QA dataset explicitly specifying such question characteristics allows for fine-grained inspection of system performance. **However, to the best of our knowledge, none of the existing QA datasets** (Voorhees and Tice, 2000; Berant et al., 2013; Cai and Yates, 2013; Lopez et al., 2013; Bordes et al., 2015; Serban et al., 2016) **provides question characteristics. In this work, we make the first attempt to generate questions with explicitly specified characteristics, and examine the impact of various question characteristics in QA.**

We present a semi-automated framework (Figure 1) to construct QA datasets with characteristic

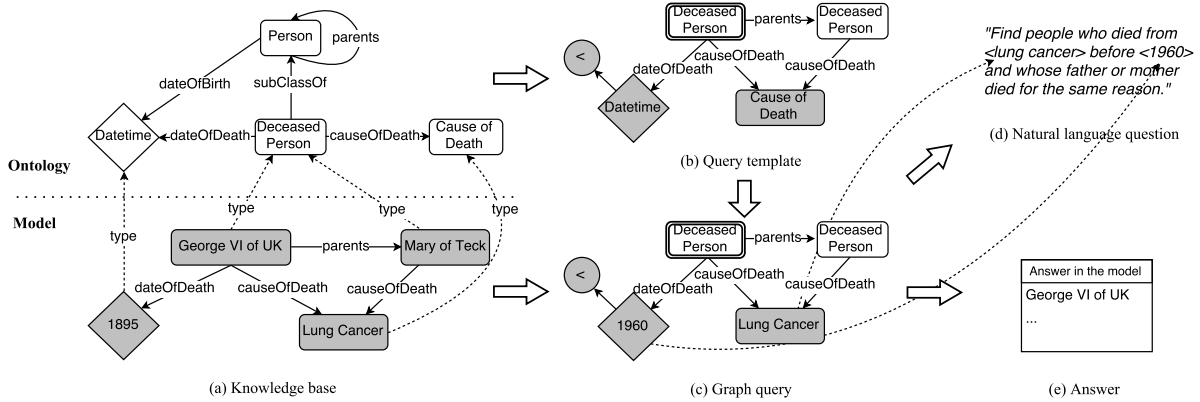


Figure 1: Running example of our framework. Graph queries are first generated from a knowledge base. After refinement (not shown), graph queries are sent to human annotators and converted into natural language questions. Answers are collected from the knowledge base.

specification from a knowledge base. The framework revolves around an intermediate **graph query** representation, which helps to formalize question characteristics and collect answers. We **first** automatically generate graph queries from a knowledge base, and **then** employ human annotators to convert graph queries into questions.

Automating graph query generation brings with it the challenge of assessing the quality of graph queries and filtering out bad ones. Our framework tackles the challenge by combining structured information in the knowledge base and statistical information from the Web. First, we identify *redundant components* in a graph query and develop techniques to remove them. Furthermore, based on the frequency of entities, classes, and relations mined from the Web, we quantify the *commonness* of a graph query and filter out too rare ones.

We employ a semi-automated approach for the conversion from graph query to natural language question, which provides two levels of paraphrasing: Common lexical forms of an entity (e.g., “*Queen Elizabeth*” and “*Her Majesty the Queen*” for `ElizabethII`) mined from the Web are used as entity paraphrases, and the remaining parts of a question are paraphrased by annotators. As a result, dozens of paraphrased questions can be produced for a single graph query.

To demonstrate the usefulness of question characteristics in QA evaluation, we construct a new dataset with over 5,000 questions based on Freebase using the proposed framework, and extensively eval-

uate several QA systems. A couple of new findings about system performance and question difficulty are discussed. For example, different from the results based on previous QA datasets (Yao et al., 2014), we **find** that semantic parsing in general works better than information extraction on our dataset. Information extraction based QA systems have trouble dealing with questions requiring aggregation or with multiple answers. A holistic understanding of the whole question is often needed for hard questions. The experiments point out an array of issues that future QA systems may need to solve.

2 Related Work

Early QA research has extensively studied problems like question taxonomy, answer type, and knowledge sources (Burger et al., 2001; Hirschman and Gaizauskas, 2001; Voorhees and Tice, 2000). This work mainly targets factoid questions with one or more answers that are guaranteed to exist in a KB.

A few KB-based QA datasets have been proposed recently. QALD (Lopez et al., 2013) and FREE917 (Cai and Yates, 2013) contain hundreds of hand-crafted questions. QALD also indicates whether a question requires aggregation. Both based on single Freebase triples, SIMPLEQUESTIONS (Bordes et al., 2015) employ human annotators to formulate questions, while Serban et al. (2016) use a recurrent neural network to automatically formulate questions. They are featured by a large size, but the questions only concern single triples, while our framework can generate ques-

tions involving multiple triples and various functions. Wang et al. (2015) generate question-answer pairs for closed domains like basketball. They also first generate logical forms (λ -DCS formulae (Liang, 2013) in their case), and then convert logical forms into questions via crowdsourcing. Logical forms are first converted into canonical questions to help crowdsourcing workers. Different from previous works, we put a particular focus on generating questions with diversified characteristics in a systematic way, and examining the impact of different question characteristics in QA.

Another attractive way for QA dataset construction is to collect questions from search engine logs (Bendersky and Croft, 2009). For example, WEBQUESTIONS (Berant et al., 2013) contains thousands of popular questions from Google search, and Yih et al. (2016) have manually annotated these questions with logical forms. However, automatic characterization of questions is hard, while manual characterization is costly and requires expertise. Moreover, users’ search behavior is shaped by search engines (Aula et al., 2010). Due to the inadequacy of current search engines to answer advanced questions, users may adapt themselves accordingly and mostly ask simple questions. Thus questions collected in this way, to some extent, may still not well reflect the true distribution of user information needs, nor does it fully exploit the potential of KB-based QA. Collecting answers is yet another challenge for this approach. Yih et al. (2016) show that only 66% of the WEBQUESTIONS answers, which were collected via crowdsourcing, are completely correct. On the other hand, although questions generated from a KB may not follow the distribution of user information needs, it has the advantage of explicit question characteristics, and enables programmatic configuration of question generation. Also, answer collecting is automated without involving human labor and errors.

3 Background

3.1 Knowledge Base

In this work, we mainly concern knowledge bases storing knowledge about entities and relations in the form of triples (simply *knowledge bases* hereafter). Suppose \mathcal{E} is a set of entities, \mathcal{L} a set of literals ($\mathcal{I} =$

$\mathcal{E} \cup \mathcal{L}$ is also called *individuals*), \mathcal{C} a set of classes, and \mathcal{R} a set of directed relations, a knowledge base \mathcal{K} consists of two parts: an *ontology* $\mathcal{O} \subseteq \mathcal{C} \times \mathcal{R} \times \mathcal{C}$ and a *model* $\mathcal{M} \subseteq \mathcal{E} \times \mathcal{R} \times (\mathcal{C} \cup \mathcal{E} \cup \mathcal{L})$. In other words, an ontology specifies classes and relations between classes, and a model consists of facts about individuals. Such knowledge bases can be naturally represented as a directed graph, e.g., Figure 1(a). Literal classes such as `Datetime` are represented as diamonds, and other classes are rounded rectangles. Individuals are shaded. We assume relations are typed, i.e., each relation is associated with a set of *domain and range classes*. Facts of a relation must be compatible with its domain and range constraints. Without loss of generality, we use Freebase (June 2013 version) in this work for compatibility with the to-be-tested QA systems. It has 24K classes, 65K relations, 41M entities, and 596M facts.

3.2 Graph Query

Motivated by the graph-structured nature of knowledge bases, we adopt a graph-centric approach. We hinge on a formal representation named *graph query* (e.g., Figure 1(c)), developed on the basis of Yih et al. (2015) and influenced by λ -DCS (Liang, 2013).

Syntax. A graph query q is a connected directed graph built on a given knowledge base \mathcal{K} . It comprises three kinds of nodes: (1) *Question node* (double rounded rectangle), a free variable. (2) *Un-grounded node* (rounded rectangle or diamond), an existentially quantified variable. (3) *Grounded node* (shaded rounded rectangle or diamond), an individual. In addition, there are *functions* (shaded circle) such as `<` and `count` applied on a node. Nodes are typed, each associated with a class. Nodes are connected by directed *edges* representing relations. Entities on the grounded nodes are called *topic entities*.

Semantics. Graph query is a strict subset of λ -calculus. For example, the graph query in Figure 1(c) can be written in λ -calculus (an existentially quantified variable is imposed by `<`):

```

 $\lambda x. \exists y. \exists z. \text{type}(x, \text{DeceasedPerson})$ 
 $\wedge \text{type}(y, \text{DeceasedPerson})$ 
 $\wedge \text{type}(z, \text{Datetime}) \wedge \text{parents}(x, y)$ 
 $\wedge \text{causeOfDeath}(x, \text{LungCancer})$ 
 $\wedge \text{causeOfDeath}(y, \text{LungCancer})$ 
 $\wedge \text{dateOfDeath}(x, z) \wedge <(z, 1960).$ 

```

The *answer* of a graph query q , denoted as $\llbracket q \rrbracket_{\mathcal{K}}$, can be easily obtained from \mathcal{K} . For example, if \mathcal{K} is stored in a RDF triplestore, then q can be automatically converted into a SPARQL query and run against \mathcal{K} to get the answer. Compared with Yih et al. (2015), graph queries are not constrained to be tree-structured, which grants us a higher expressivity. For example, linguistic phenomena like anaphora (e.g., Figure 1(d)) become easier to model.

4 Automatic Graph Query Generation

Our framework proceeds as follows: (1) Generate *query templates* from a knowledge base, ground the templates to generate graph queries, and collect answers (this section). (2) Refine graph queries to retain high-quality ones (Section 5). (3) Convert graph queries into questions via crowdsourcing (Section 6).

We now describe an algorithm to generate the query template shown in Figure 1(b) (excluding the function for now). For simplicity, we will focus on the case of a single question node. Nevertheless, the proposed framework can be extended to generate graph queries with multiple question nodes. The algorithm takes as *input an ontology* (Figure 1(a)) and *the desired number of edges*. All the operations are conducted in a random manner to avoid systematic biases in query generation. The *DeceasedPerson* class is first selected as the question node. We then iteratively grow it by adding neighboring nodes and edges in the ontology. In each iteration, an existing node is selected, and a new edge, which might introduce a new node, is appended to it. For example, the relation *causeOfDeath*, whose domain includes *DeceasedPerson*, is first appended to the question node, and then one of its range classes, *CauseOfDeath*, is added as a new node. When a node with the class *CauseOfDeath* already exists, it is possible to add an edge without introducing a new node. The same relation or class can be added multiple times, e.g., “parent of parent”.

Topic entities like *LungCancer* play an important role in a question. A query template contains some template nodes that can be grounded with different topic entities to generate different graph queries. We *randomly choose* a few nodes as template. It may cause problems. For example, ground-

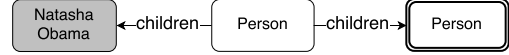


Figure 2: Mutual exclusivity example. Entities on different nodes should be different.

ing one node may make some others redundant. We conduct a formal study on this in Section 5.1.

Functions such as counting and comparatives are pervasive in real-life questions, e.g., “*how many*”, “*the most recent*”, and “*people older than*”, but are scarce in existing QA datasets. We incorporate functions as an important question characteristic, and consider *nine common functions*, grouped into three categories: counting (*count*), superlative (*max*, *min*, *argmax*, *argmin*), and comparative (*>*, *≥*, *<*, *≤*). More functions can be incorporated in the future. See Appendix A for examples. We randomly add functions to *compatible nodes* in query templates. In the running example, the *<* function imposes the constraint that only people who passed away before a certain date should be considered. Each query will have at most one function.

We then ground the template nodes with individuals to *generate graph queries*. A grounding is valid if the individuals *conform* with the class of the corresponding template nodes, and the resulted answer is not empty. For example, by grounding *CauseOfDeath* with *LungCancer* and *Datetime* with 1960, we get the graph query in Figure 1(c). A *query template can render multiple groundings*.

Finally, we convert a graph query into a SPARQL query and execute it using Virtuoso Open-Source 7 to collect answers. We further impose *mutual exclusivity* in SPARQL queries, that is, the entities on any two nodes in a graph query should be different. Consider the example in Figure 2, which is asking for the siblings of Natasha Obama. Without mutual exclusivity, however, Natasha Obama herself will also be included as an answer, which is not desired.

5 Query Refinement

Since graph queries are randomly generated, some of them may not correspond to an interesting question. Next we study two query characteristics, *redundancy* and *commonness*, based on which we provide mechanisms for automatic query refinement.

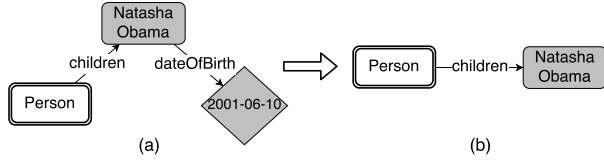


Figure 3: Query minimization example: (a) Graph query with redundant components. (b) Graph query after minimization.

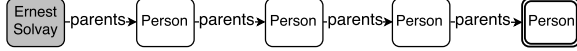


Figure 4: Uncommon query example. It is uncommon to ask for somebody’s great-great-grandparents.

5.1 Query Redundancy and Minimization

Some components (nodes and edges) in a graph query may not effectively impose any constraint on the answer. The query in Figure 3(a) is to “*find the US president whose child is Natasha Obama, and Natasha Obama was born on 2001-06-10*”. Intuitively, the bold-faced clause does not change the answer of the question. Correspondingly, the `dateOfBirth` edge and the date node are redundant. As a comparison, removing any component from the query in Figures 3(b) will change the answer. Formally, given a knowledge base \mathcal{K} , a component in a graph query q is *redundant* iff. removing it does not change the answer $\llbracket q \rrbracket_{\mathcal{K}}$.

Redundancy can be desired or not. In a question, redundant information may be inserted to reduce ambiguity. In Figure 3(a), if one uses “*Natasha*” to refer to `NatashaObama`, there comes ambiguity since it may be matched with many other entities. The additional information “*who was born on 2001-06-10*” then helps. Next we describe an algorithm to remove redundancy from queries. One can choose to either only generate queries with no redundant component, or intentionally generate redundant queries and test QA systems in presence of redundancy.

We manage to generate *minimal* queries, for which there exists no sub-query having the same answer. An important theorem, as we prove in Appendix B, is the equivalency of minimality and non-redundancy: A query is minimal iff. it has no redundant component. This renders a simple algorithm for query minimization, which directly detects and removes the redundant components in a query. We first examine every edge (in an arbitrary order), and remove an edge if it is redundant. Redundant nodes

will then become disconnected to the question node and are thus eliminated. It is easy to prove that the produced query (e.g., Figure 3(b)) is minimal, and has the same answer as the original query.

5.2 Commonness Checking

We now quantify the *commonness* of graph queries. The benefits of this study are two-fold. First, it provides a refinement mechanism to reduce too rare queries. Second, commonness is itself an important question characteristic. It is interesting to examine its impact on question difficulty. Consider the example in Figure 4, which asks for “*the great-great-grandparents of Ernest Solvay*”. It is minimal and logically plausible. Few users, however, are likely to come up with it. Ernest Solvay is famous for the Solvay Conferences, but few people outside the science community may know him. Although `Person` and `parents` are common, asking for the great-great-grandparents is quite uncommon.

A query is more common if users would more likely come up with it. We define the commonness of a query q as its *probability* $p(q)$ of being picked among all possible queries from a knowledge base. The problem then boils down to estimating $p(q)$. It is hard, if not impossible, to exhaust the whole query space. We thus make the following simplification. We break down query commonness by components, assuming mutual independence between components, and omit functions:

$$p(q) = \prod_{i \in \mathcal{I}_q} p(i) \times \prod_{c \in \mathcal{C}_q} p(c) \times \prod_{r \in \mathcal{R}_q} p(r), \quad (1)$$

where \mathcal{I}_q , \mathcal{C}_q , \mathcal{R}_q are the *multi-set* of the individuals, classes, and relations in q , respectively. Repeating components are thus accumulated (c.f. Figure 4).

We propose a data-driven method, using statistical information from the Web, to estimate $p(i)$, $p(c)$, and $p(r)$. Other methods like domain-knowledge based estimation are also applicable if available. We start with entity probability $p(e)$ (excluding literals for now). If users mention an entity more frequently, its probability of being observed in a question should be higher. We use a large entity linking dataset, FACC1 (Gabrilovich et al., 2013), which identifies around 10 billion mentions of Freebase entities in over 1 billion web documents. The estimated linking precision and recall are 80-85% and 70-85%, re-

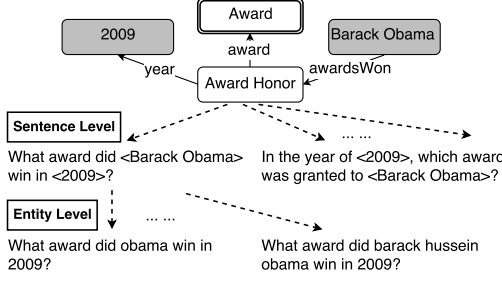


Figure 5: Question generation and paraphrasing.

spectively. Suppose entity e has $n(e)$ mentions, then $p(e) = \frac{n(e)}{\sum_{e' \in \mathcal{E}} n(e')}$. For a class c , probability $p(c)$ is higher if it has more frequently mentioned entities. If we use $e \in c$ to indicate e is an instance of c , then $p(c) = \frac{\sum_{e \in c} n(e)}{\sum_{c' \in \mathcal{C}} \sum_{e \in c'} n(e')}$. Estimating $p(r)$ requires relation extraction from texts, which is hard. We make the following simplification: If (e_1, r, e_2) is a fact in the knowledge base, we increase $n(r)$ by 1 if e_1 and e_2 co-occur in a document. This suffices to distinguish common relations from uncommon ones. We then define $p(r) = \frac{n(r)}{\sum_{r' \in \mathcal{R}} n(r')}$. Finally, we use frequency information from the knowledge base to smooth the probabilities, e.g., to avoid zero probabilities. The probability of literals are solely determined by the frequency information from the knowledge base. Refer to Appendix C for the resulted probability distributions.

6 Natural Language Conversion

In order to ensure naturalness and diversity, we employ human annotators to manually convert graph queries into natural language questions. We manage to provide two levels of paraphrasing (Figure 5). Each query is sent to multiple annotators for **sentence-level paraphrasing**. In addition, we use different lexical forms of an entity mined from FACC1 for **entity-level paraphrasing**. We provide a ranked list of common lexical forms and the corresponding frequency for each topic entity. For example, the lexical form list for `UnitedStatesOfAmerica` is “us” (108M), “united states” (44M), “usa” (22M), etc. Finally, graph queries are automatically translated into SPARQL queries to collect answers.

Natural language generation (NLG) (Wen et al., 2015; Serban et al., 2016; Dušek and Jurčiček, 2015) would be a good complement to our framework, the combination of which can lead to a fully-automated

pipeline to generate QA datasets. For example, Serban et al. (2016) automatically convert Freebase triples into questions with a neural network. More sophisticated NLG techniques able to handle graph queries involving multiple relations and various functions are an interesting future direction.

7 Experiments

We have constructed a new QA dataset, named GRAPHQUESTIONS, using the proposed framework, and tested several QA systems to show that it enables fine-grained inspection of QA systems.

7.1 Dataset Construction

We **first** randomly generated a set of minimal graph queries, and removed the ones whose commonness is below a certain threshold. The remaining graph queries were **then** screened by graduate students, and a *canonical* question was generated for each query, with each being verified by at least two students. We recruited 160 crowdsourcing workers from Amazon MTurk to generate sentence-level paraphrases of the canonical questions. Trivial paraphrases (e.g., “*which city*” vs. “*what city*”) were manually removed to retain a high diversity in paraphrasing. **At most 3 entity-level paraphrases were used for each sentence-level paraphrase.**

7.2 Dataset Analysis

GRAPHQUESTIONS contains 500 graph queries, 2,460 sentence-level paraphrases, and 5,166 questions². The dataset presents a high diversity and covers a wide range of domains including People, Astronomy, Medicine, etc. Specifically, it contains 148, 506, 596, 376 and 3,026 distinct domains, classes, relations, topic entities, and words, respectively. We evenly split GRAPHQUESTIONS into a training set and a testing set. **All the paraphrases of the same graph query are in the same set.**

While there are other question characteristics derivable from graph query, we will focus on the following ones: *structure complexity*, *function*, *commonness*, *paraphrasing*, and *answer cardinality*. We

²For each query template, we only generate one graph query, but one can also generate multiple graph queries, and easily get the corresponding questions by replacing the topic entities. This will significantly increase the total number of questions, and can be helpful in training.

	# of edges			Function				$\log_{10}(p(q))$				A	
	1	2	3	none	count	super.	comp.	$[-40, 30]$	$[-30, 20]$	$[-20, 10]$	$[-10, 0]$	1	> 1
# of graph queries	321	144	35	350	61	42	47	60	135	283	22	332	168
# of questions	3094	1648	424	3855	710	332	269	653	1477	2766	270	3487	1679

Table 1: Characteristic statistics. |A| is answer cardinality. Refer to Appendix D for paraphrase and other fine-grained distributions.

Question	Domain	Answer	# of edges	Function	$\log_{10}(p(q))$	A
Find terrorist organizations involved in September 11 attacks .	Terrorism	alQaeda	1	none	-16.67	1
The September 11 attacks were carried out with the involvement of what terrorist organizations?						
Who did nine eleven ?						
How many children of Eddard Stark were born in Winterfell ?	Fictional Universe	3	2	count	-23.34	1
Winterfell is the home of how many of Eddard Stark 's children?						
What's the number of Ned Stark 's children whose birthplace is Winterfell ?						
In which month does the average rainfall of New York City exceed 86 mm?	Travel	March, August ...	3	comp.	-37.84	7
Rainfall averages more than 86 mm in New York City during which months?						
List the calendar months when NYC averages in excess of 86 millimeters of rain?						

Table 2: Example questions and characteristics. Three sentence-level paraphrases are shown for each graph query, with the last one also involving entity-level paraphrasing. Topic entities are bold-faced. More examples can be found in Appendix D.

use the number of edges to quantify structure complexity, and limit it to **at most 3**. Commonness is limited to $\log_{10}(p(q)) \geq -40$ (c.f. Eq. 1). As shown in Section 7.4.2, such questions are already very hard for existing QA systems. Nevertheless, the proposed framework can be used to generate questions with different characteristic distributions. Some statistics are shown in Table 1 and more fine-grained statistics can be found in Appendix D.

Several example questions are shown in Table 2. **Sentence-level paraphrasing** requires to handle both commands (the first example) and “Wh” questions, light verbs (“Who did nine eleven?”), and changes of syntactic structure (“The September 11 attacks were carried out with the involvement of what terrorist organizations?”). **Entity-level paraphrasing** tests the capability of QA systems on abbreviation (“NYC” for New York City), world knowledge (“Her Majesty the Queen” for ElizabethII), or even common typos (“Shakespeare” for WilliamShakespeare). Numbers and dates are also common, e.g., “Which computer operating system was released on Sept. the 20th, 2008?”

We compare several QA datasets constructed from Freebase, shown in Table 3. Datasets focusing on single-relation questions are of a larger scale,

but are also of a significant lack in question characteristics. Overall GRAPHQUESTIONS presents the highest diversity in question characteristics.

7.3 Setup

We evaluate three QA systems whose source code is publicly available: **SEMPRE** (Berant et al., 2013), **PARASEMPRE** (Berant and Liang, 2014), and **JACANA** (Yao and Van Durme, 2014). SEMPRE and PARASEMPRE follow the semantic parsing paradigm. SEMPRE conducts a bottom-up beam-based parsing on questions to find the best logical form. PARASEMPRE, in a reverse manner, enumerates a set of logical forms, generates a canonical utterance for each logical form, and ranks logical forms according to how well the canonical utterance paraphrases the input question. In contrast, JACANA follows the information extraction paradigm, and builds a classifier to directly predict whether an individual is the answer. They all use Freebase.

The main metric for answer quality is **the average F1 score**, following Berant and Liang (2014). Because a question can have more than one answer, individual precision, recall, and F1 scores are first computed on each question and then averaged. When a system generates no response for a question,

Dataset	# of Questions	# of Multi-relation	Function (count/super./comp.)	Commonness	Paraphrase	Multi-answer
GRAPHQUESTIONS (this work)	5166	2072	710 / 332 / 269	+	+	+
WEBQUESTIONS ^{SP} (Yih et al., 2016) ¹	4737	2075	2 / 168 / 334	-	-	+
FREE917 (Cai and Yates, 2013)	917	229	185 / 0 / 0	-	-	+
Serban et al. (2016)	30M	0	0 / 0 / 0	-	-	-
SIMPLEQUESTIONS (Bordes et al., 2015)	108K	0	0 / 0 / 0	-	-	-

Table 3: Comparison of QA datasets constructed from Freebase. GRAPHQUESTIONS is the richest in question characteristics.

System	F1	Time/s
SEMPRE	10.80	56.19
PARASEMPRE	12.79	18.43
JACANA	5.08	2.01

Table 4: Overall performance on GRAPHQUESTIONS.

precision is 1, recall is 0, and F1 is 0. Average run-time is used for efficiency. Results are shown in percentage. Systems are trained on the training set using the suggested configurations (Appendix E). We use student’s t test at $p = 0.05$ for significance test.

7.4 Results

7.4.1 Overall Evaluation

Compared with the scores on WEBQUESTIONS (30%-40%), the scores on GRAPHQUESTIONS are lower (Table 4). This is because GRAPHQUESTIONS contains questions over a broader range of difficulty levels. For example, it is more diverse in topics (Appendix D); also the scores become much closer when excluding paraphrasing (Section 7.4.2).

JACANA achieves a comparable F1 score with SEMPRES and PARASEMPRES on WEBQUESTIONS (Yao et al., 2014). On GRAPHQUESTIONS, however, SEMPRES and PARASEMPRES significantly outperform JACANA (both $p < 0.0001$). The following experiments will give more insights about where the performance difference comes from. On the other hand, JACANA is much faster, showing an advantage of information extraction. The semantic parsing systems spend a lot of time on executing SPARQL queries. Bypassing SPARQL and directly working on the knowledge base may be a promising way to speed up semantic parsing on large knowledge bases (Yih et al., 2015).

²WEBQUESTIONS^{SP} is WEBQUESTIONS with manually annotated logical forms. Only those with a full logical form are included (4737 / 5810).

7.4.2 Fine-grained Evaluation

With explicitly specified question characteristics, we are able to further inspect QA systems.

Structure Complexity. We first break down system performance by structure. Answer quality is in general sensitive to the complexity of question structure: As the number of edges increases, F1 score decreases (Figure 6(a)). The tested systems often fail to take into account auxiliary constraints in a question. For example, for “How many children of Ned Stark were born in Winterfell?” SEMPRES fails to identify the constraint “born in Winterfell”, so it also considers Ned Stark’s bastard son, Jon Snow, as an answer, who was not born in Winterfell. Answering questions involving multiple relations using large knowledge bases remain an open problem. The large size of knowledge bases prohibits exhaustive search, so smarter algorithms are needed to efficiently prune the answer space. Berant and Liang (2015) point out an interesting direction, leveraging agenda-based parsing with imitation learning for efficient search in the answer space.

Function. In terms of functions, while SEMPRES and PARASEMPRES perform well on count questions, all the tested systems perform poorly on questions with superlatives or comparatives (Figure 6(b)). JACANA has trouble dealing with functions because it does not conduct quantitative analysis over the answer space. SEMPRES and PARASEMPRES do not generate logical forms with superlatives and comparatives, so they cannot answer such questions well.

Commonness. Not surprisingly, more common questions are in general easier to answer (Figure 6(c)). An interesting observation is that SEMPRES’s performance gets worse on the most common questions. The cause is likely rooted in how the QA systems construct their candidate answer sets. PARASEMPRES and JACANA exhaustively construct candidate sets, while SEMPRES employs a bottom-up beam search, making it more sensitive to the size of

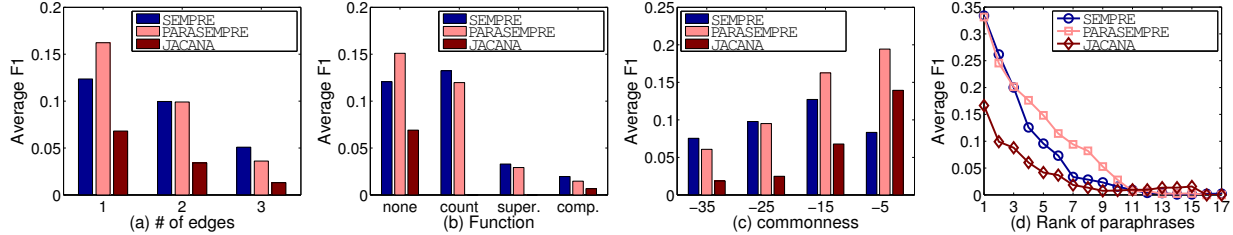


Figure 6: Performance breakdown by (a) structure complexity, (b) function, (c) commonness, and (d) paraphrase. Note that in (c) $x = -5$ indicates the commonness range $-10 \leq \log_{10}(p(q)) < 0$.

the candidate answer space. Common entities like `UnitedStatesOfAmerica` are often featured by a high degree in knowledge bases (e.g., 1 million neighboring entities), which dramatically increases the size of the candidate answer space. During SEMPRE’s iterative beam search, many correct logical forms may have fallen off beam before getting into the final candidate set. We checked the percentage of questions for which the correct logical form is in the final candidate set, and found that it decreased from 19.8% to 16.7% when commonness increased from -15 to -5, providing an evidence for the intuition.

Paraphrasing. It is **critical** for a system to tolerate the wording varieties of users. We make the first effort to evaluate QA systems on paraphrasing. For each system, **we rank, in descending order,** all the paraphrases derived from the same graph query by their F1 score achieved by the system, and then compute the average F1 score of each rank. In Figure 6(d), the decreasing rate of a curve thus describes a system’s robustness to paraphrasing; **the higher, the less robust.** All the systems achieve a reasonable score on the top-1 paraphrases, i.e., when a system can choose the paraphrase it can best answer. The F1 scores drop quickly in general. On the fourth-ranked paraphrases, the F1 score of SEMPRE, PARASEMPRE, and JACANA are respectively only 37.65%, 53.2%, and 36.2% of their score on the top-1 paraphrases. Leveraging paraphrasing in its model, **PARASEMPRE** does seem to be more robust. The results show that how to **handle paraphrased questions** is still a challenging problem.

Answer Cardinality. SEMPRE and JACANA get a significantly lower F1 score (both $p < 0.0001$) on **multi-answer questions** (Table 5), mainly coming from a decrease on recall. The decrease of PARASEMPRE is not significant ($p=0.29$). The particularly significant decrease of JACANA demon-

System	$ A $	Prec.	Rec.	F1
SEMPRE	1	59.81	16.11	12.68
	> 1	62.38	9.17	6.78
PARASEMPRE	1	17.42	17.58	13.25
	> 1	19.65	17.23	11.82
JACANA	1	14.77	6.56	6.56
	> 1	11.80	1.43	1.98

Table 5: Performance breakdown by answer cardinality $|A|$.

strates the difficulty of training a classifier that can predict all of the answers correctly; semantic parsing is more **robust** in this case. The precision of SEMPRE is high because it generates **no response for many questions**. Note that under the current definition, the average F1 score is not the harmonic mean of the average precision and recall (c.f. Section 7.3).

8 Conclusion

We proposed a framework to generate characteristic-rich questions for question answering (QA) evaluation. Using the proposed framework, we constructed a new and challenging QA dataset, and extensively evaluated several QA systems. The findings point out an array of issues that future QA research may need to solve.

9 Acknowledgements

This research was sponsored in part by the Army Research Laboratory under cooperative agreements W911NF09-2-0053, NSF IIS 0954125, and NSF IIS 1528175. The views and conclusions contained herein are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notice herein.

References

- Anne Aula, Rehan M. Khan, and Zhiwei Guan. 2010. How does search behavior change as search becomes more difficult? In *Proceedings of CHI*.
- Junwei Bao, Nan Duan, Ming Zhou, and Tiejun Zhao. 2014. Knowledge-based question answering as machine translation. In *Proceedings of ACL*.
- Michael Bendersky and W. Bruce Croft. 2009. Analysis of long queries in a large scale search log. In *Proceedings of the 2009 workshop on Web Search Click Data*.
- Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of ACL*.
- Jonathan Berant and Percy Liang. 2015. Imitation learning of agenda-based semantic parsers. *Transactions of the Association for Computational Linguistics*, 3:545–558.
- Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of EMNLP*.
- Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of SIGMOD*.
- Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*.
- John Burger, Claire Cardie, Vinay Chaudhri, Robert Gaizauskas, Sanda Harabagiu, David Israel, Christian Jacquemin, Chin-Yew Lin, Steve Maorano, George Miller, et al. 2001. Issues, tasks and program structures to roadmap research in question & answering (q&a). In *Document Understanding Conferences Roadmapping Documents*, pages 1–35.
- Qingqing Cai and Alexander Yates. 2013. Large-scale semantic parsing via schema matching and lexicon extension. In *Proceedings of ACL*.
- Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2015. Question answering over Freebase with multi-column convolutional neural networks. In *Proceedings of ACL*.
- Ondřej Dušek and Filip Jurčiček. 2015. Training a natural language generator from unaligned data. In *Proceedings of ACL*.
- Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2014. Open question answering over curated and extracted knowledge bases. In *Proceedings of KDD*.
- Evgeniy Gabrilovich, Michael Ringgaard, and Amarnag Subramanya. 2013. FACC1: Freebase annotation of ClueWeb corpora, version 1 (release date 2013-06-26, format version 1, correction level 0). <http://lemurproject.org/clueweb09/> and <http://lemurproject.org/clueweb12/>.
- Lynette Hirschman and Robert Gaizauskas. 2001. Natural language question answering: the view from here. *natural language engineering*, 7(4):275–300.
- Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of EMNLP*.
- Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, et al. 2014. DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web Journal*, 6(2):167–195.
- Percy Liang. 2013. Lambda dependency-based compositional semantics. *arXiv preprint arXiv:1309.4408*.
- Vanessa Lopez, Christina Unger, Philipp Cimiano, and Enrico Motta. 2013. Evaluating question answering over linked data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 21:3–13.
- Siva Reddy, Mirella Lapata, and Mark Steedman. 2014. Large-scale semantic parsing without question-answer pairs. *Transactions of the Association for Computational Linguistics*, 2:377–392.
- Iulian Vlad Serban, Alberto García-Durán, Caglar Gulcehre, Sungjin Ahn, Sarath Chandar, Aaron Courville, and Yoshua Bengio. 2016. Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus. In *Proceedings of ACL*.
- Huan Sun, Hao Ma, Wen-tau Yih, Chen-Tse Tsai, Jingjing Liu, and Ming-Wei Chang. 2015. Open domain question answering via semantic enrichment. In *Proceedings of WWW*.
- Ellen M Voorhees and Dawn M Tice. 2000. Building a question answering test collection. In *Proceedings of SIGIR*.
- Yushi Wang, Jonathan Berant, and Percy Liang. 2015. Building a semantic parser overnight. In *Proceedings of ACL*.
- Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. In *Proceedings of EMNLP*.
- Mohamed Yahya, Klaus Berberich, Shady Elbassuoni, Maya Ramanath, Volker Tresp, and Gerhard Weikum. 2012. Deep answers for naturally asked questions on the web of data. In *Proceedings of WWW*.
- Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with Freebase. In *Proceedings of ACL*.
- Xuchen Yao, Jonathan Berant, and Benjamin Van Durme. 2014. Freebase QA: Information extraction or semantic parsing? In *Proceedings of ACL*.
- Xuchen Yao. 2015. Lean question answering over Freebase from scratch. In *Proceedings of NAACL*.

- Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jian-feng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of ACL*.
- Wen-tau Yih, Matthew Richardson, Christopher Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of ACL*.
- Lei Zou, Ruizhe Huang, Haixun Wang, Jeffrey Xu Yu, Wenqiang He, and Dongyan Zhao. 2014. Natural language question answering over rdf: a graph data driven approach. In *Proceedings of SIGMOD*.

Appendix

A Functions

We list the functions considered in this work in Table 1. Each function can only be applied on certain kinds of nodes, i.e., its domain.

B Query Minimization

Here we give a more formal description of our query minimization algorithm. Suppose a knowledge base \mathcal{K} is given to compute the answer of graph queries, then the following concepts can be naturally defined.

Definition 1 (Redundant Component) *A component in a graph query q is redundant iff. removing it does not change the answer $\llbracket q \rrbracket_{\mathcal{K}}$.*

Definition 2 (Minimal Query) *A graph query q is minimal iff. there is no sub-query q' , i.e., a graph query resulted from removing any number of components from q , such that $\llbracket q' \rrbracket_{\mathcal{K}} = \llbracket q \rrbracket_{\mathcal{K}}$, except q itself.*

Definition 3 (Equivalent Minimal Query) *Given two graph queries q and q' , q' is an equivalent minimal query of q iff. (1) $\llbracket q' \rrbracket_{\mathcal{K}} = \llbracket q \rrbracket_{\mathcal{K}}$, and (2) q' is minimal.*

Note that we define redundancy based on a given knowledge base because it suffices for our purpose. KB-independent minimization is out of the scope of this work, but is of interest for future study.

Our query minimization algorithm is outlined in Algorithm 1. We first examine every edge (in an arbitrary order), and remove an edge if it is redundant. So the `dateOfBirth` edge in Figure 1(a) will be removed. Some nodes may become disconnected to the question node after removing redundant edges, which indicates that the node is redundant. We thus delete the redundant nodes as well, e.g., the `date` node in Figure 1(a). The query in Figure 1(b) is produced as output.

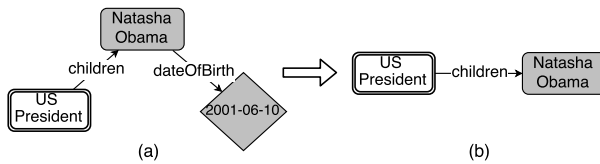


Figure 1: Query minimization: (a) a query with redundant components, (b) a corresponding equivalent minimal query.

Proof. The proof is straightforward. For convenience, we assume there is only one question node, but the proof generalizes to graph queries with multiple question nodes easily. When there is no function on the question node (`count`, `max` or `min`), the answer is a set of individuals. Since a component in a graph query imposes some constraint on the answer, removing it will not eliminate any individual from the answer, rather, the answer will only stay unchanged or get expanded. More formally, given a knowledge base \mathcal{K} , for a graph query q and a sub-query q' , we have $\llbracket q \rrbracket_{\mathcal{K}} \subseteq \llbracket q' \rrbracket_{\mathcal{K}}$. When there is a function on the question node, the answer is a single number, but the same assertion holds. For example, if there is a `count` function on the question node, it is asserted that the answer, i.e., the answer cardinality of the graph query without the function, will be non-decreasing when iteratively removing other components of the graph query. \square

Algorithm 1: Graph Query Minimization

Input: a graph query q , a knowledge base \mathcal{K}

Output: an equivalent minimal query of q

```

1 foreach edge  $e$  of  $q$  do
2    $q' \leftarrow q$  with  $e$  removed
3   if  $\llbracket q' \rrbracket_{\mathcal{K}} = \llbracket q \rrbracket_{\mathcal{K}}$  then
4      $q \leftarrow q'$ 
5   end
6 end
7 Remove nodes disconnected to the question node
8 return  $q$ 

```

To prove the generated query, denoted as q' , is an equivalent minimal query of the input query q , we need to prove (1) $\llbracket q' \rrbracket_{\mathcal{K}} = \llbracket q \rrbracket_{\mathcal{K}}$, and (2) q' is minimal. The former is guaranteed by the construction of Algorithm 1. To prove the latter, we need the following lemma:

Lemma 1. *When components (except for the question nodes) are iteratively removed from a graph query, the answer will change monotonically.*

Lemma 1 precludes such possibilities: Removing one component changes the answer, and subsequently removing another component changes the

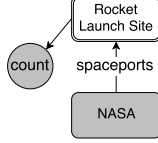
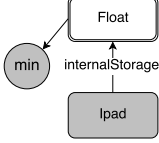
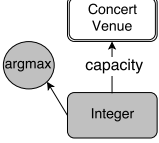
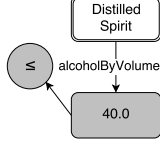
Category	Counting	Superlative		Comparative
Functions	count	max and min	argmax and argmin	$<, >, \leq$, and \geq
Domain	Question node	Question node of numeric class	Template/grounded node of numeric class	Template/grounded node of numeric class
Example				
Question	How many launch sites does nasa have?	What's the smallest internal storage of ipad?	Find the largest concert venue.	List distilled spirits with no more than 40.0% abv.

Table 1: Functions considered in this work. Domain is the type of nodes a function can be applied on.

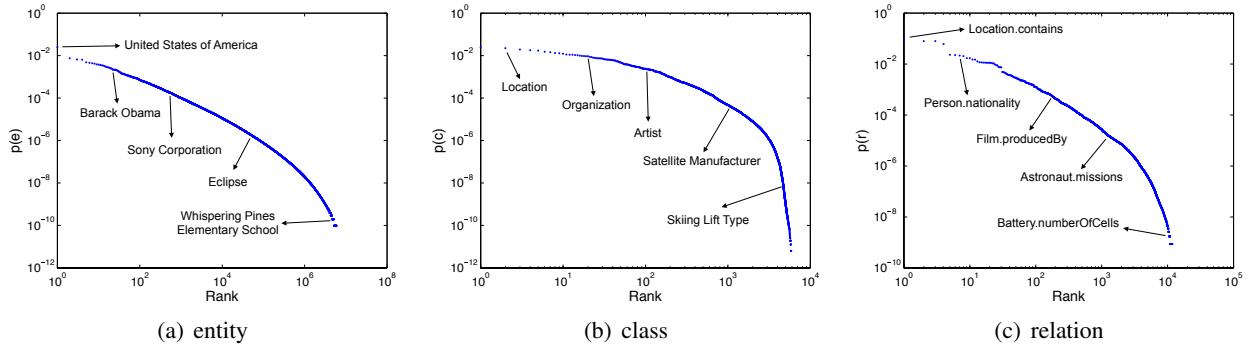


Figure 2: Component probability distributions. Components are ranked in descending order by their probability.

answer back. Therefore, it is not necessary to enumerate all the sub-queries in order to check the minimality of a graph query. Rather, we only need to examine the redundancy of each component. In other words, we have the following corollary:

Corollary 1.1. *A graph query is minimal iff. it has no redundant component.*

In addition, a single scan of the edges in an arbitrary order, as Algorithm 1 does, is sufficient, because an irredundant component will not become redundant when other components are removed.

Theorem 2. *The graph query resulted from Algorithm 1 is an equivalent minimal query of the input query.*

Proof. The answer stays unchanged by construction. After removing the redundant edges and nodes, the resulted query has no redundant component, and is therefore minimal according to Corollary 1.1. Following the definition, the resulted query is an equivalent minimal query of the input query. \square

C Probability Distribution of Freebase Components

We count the frequency n' of each component in the Freebase. For an entity e , $n'(e) = 1$; for a class c , $n'(c)$ is its number of instances; and for a relation r , $n'(r)$ is the number of facts of this relation. We then add the mention counts from FACC1 with the counts from Freebase to estimate the probability of Freebase components. The probability of literal classes are solely determined by their number of instances, and the probability of a literal instance is the same as the corresponding literal class. The distributions are shown in Figure 2. Entities with no mention in FACC1 are not shown in Figure 2(a). Classes and relations are from a filtered ontology where the User domain and the Freebase domain are removed. Relations shown in the figure are in the *class.relation* format.

D Fine-grained Statistics of GRAPHQUESTIONS

Fine-grained statistics of GRAPHQUESTIONS on different characteristics are shown in Figure (3-9).

Sub-distributions of the training set and the testing set are also shown. Note that in addition to the commonness of the whole query (Figure 6), we also give the commonness of topic entities (Figure 7). As a comparison, we also give the commonness distribution of topic entities of WEBQUESTIONS (Figure 8). GRAPHQUESTIONS contains topic entities over a broader range of commonness than WEBQUESTIONS. Overall GRAPHQUESTIONS exhibits a good diversity in all the examined characteristics. More example questions are listed in Table 2.

E Experiment Configuration Details

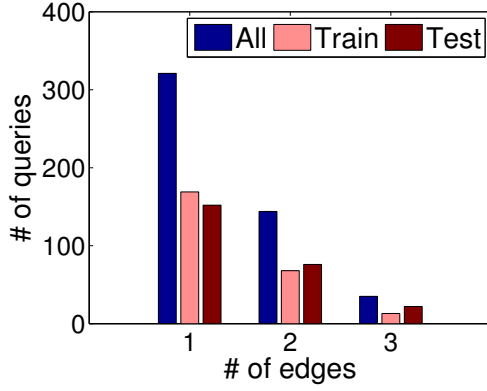
All the systems (SEMPRE, PARASEMPRE, and JACANA) are trained on the training set and tested on the testing set of GRAPHQUESTIONS, and use Freebase as the knowledge base. For SEMPRE, the grammar is the one from the original paper, the maximum training iteration is 3, the beam size is 100 for training, 200 for testing¹. The training took 5 days. For PARASEMPRE, the maximum training iteration is 3,

the number of threads is 20 for training, 1 for testing. The training took 27 hours. Both SEMPRE and PARASEMPRE cache historical SPARQL query results in order to save time. The cache from training is allowed to use during testing. For JACANA, the top-1 topic entity retrieved from the Freebase Search API is used for both training and testing². Same as the original paper, we down-sample the negative examples with a ratio of 0.2. The classifier is logistic regression with L1 regularization. The training took 1.5 hours.

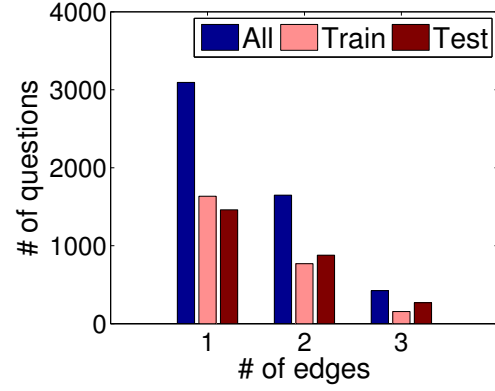
Experiments are run on a Linux server with Intel Xeon E7-8837 processors (2.67GHz) and 1T memory. A Virtuoso database is run on the same server to provide access to Freebase, so network IO cost is minimized.

¹It is too time consuming to use beam size=200 for training

²We also tried to use the top-10 retrieved topic entities for both training and testing, or use the *gold* topic entities for training and the top-10 retrieved topic entities for testing. The performance was slightly worse.

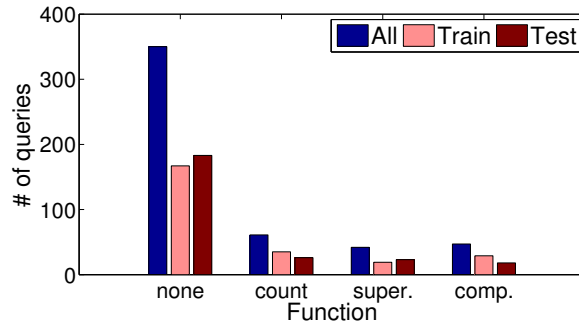


(a) Structure distribution of graph queries

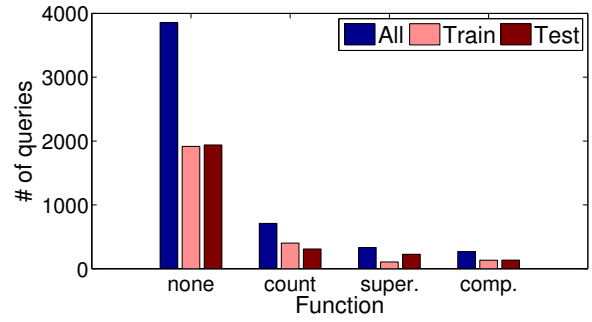


(b) Structure distribution of questions

Figure 3: Characteristics distribution of GRAPHQUESTIONS: Structure Complexity.

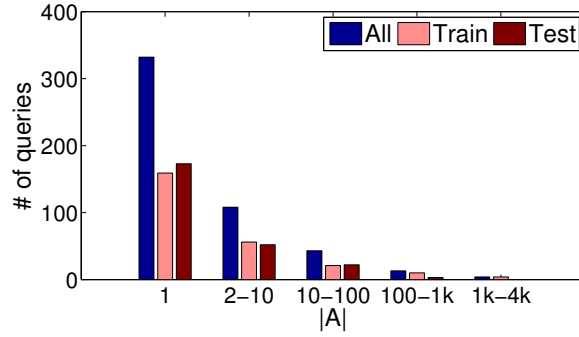


(a) Function distribution of graph queries

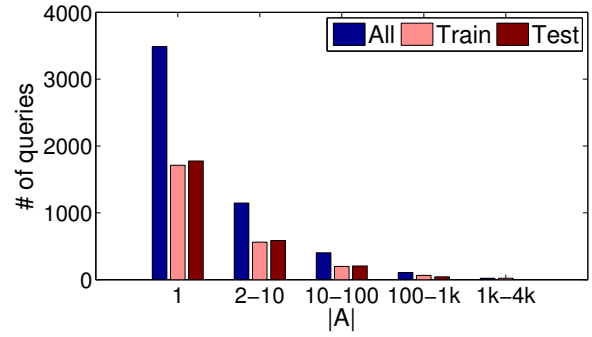


(b) Function distribution of questions

Figure 4: Characteristics distribution of GRAPHQUESTIONS: Function.

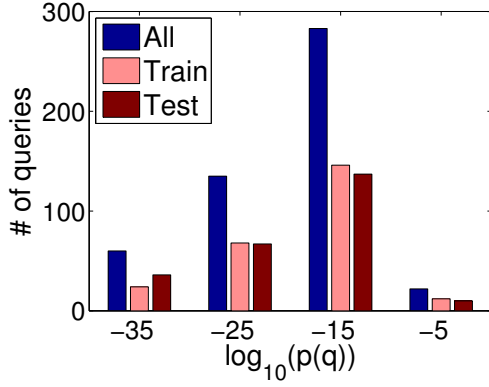


(a) Answer cardinality distribution of graph queries

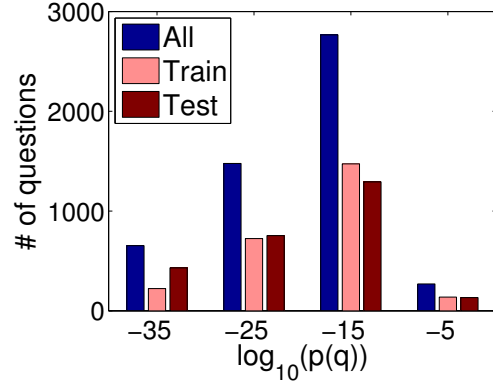


(b) Answer cardinality distribution of questions

Figure 5: Characteristics distribution of GRAPHQUESTIONS: Answer Cardinality.

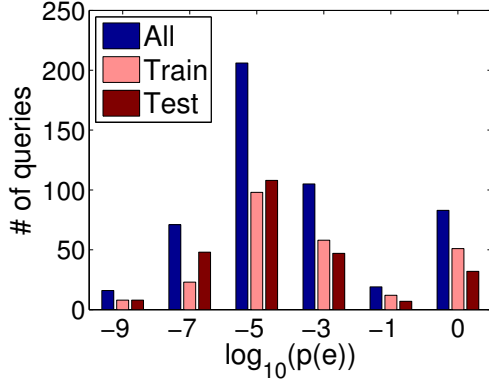


(a) Commonness distribution of graph queries

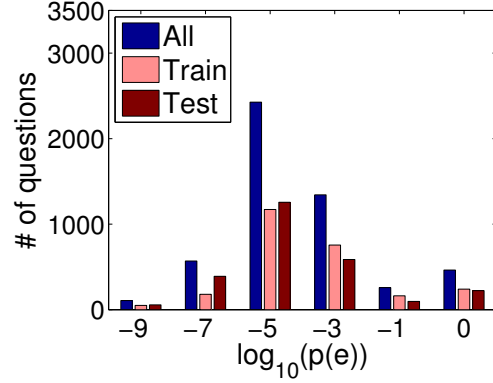


(b) Commonness distribution of questions

Figure 6: Characteristics distribution of GRAPHQUESTIONS: Commonness. Note that $x = -5$ indicates the commonness range $-10 \leq \log_{10}(p(q)) < 0$.



(a) Topic commonness distribution of graph queries



(b) Topic commonness distribution of questions

Figure 7: Characteristics distribution of GRAPHQUESTIONS: Topic Commonness. When there are multiple topic entities, the most common one is used. Note that $x = -5$ indicates the commonness range $-6 \leq \log_{10}(p(q)) < -4$. $\log_{10}(p(e)) = 0$ means there is no topic entity, e.g., “What’s the smallest Spanish autonomous city?”

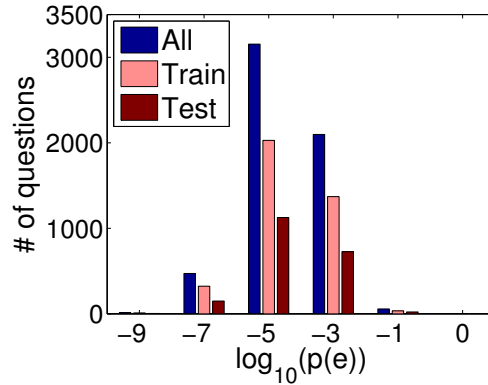
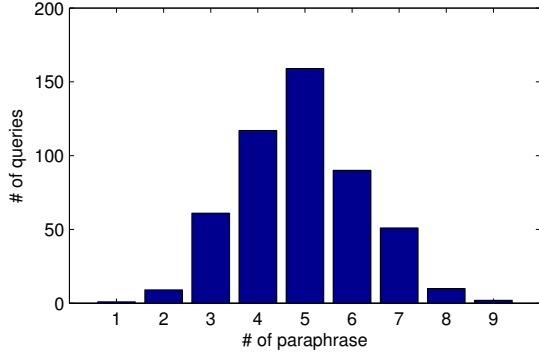
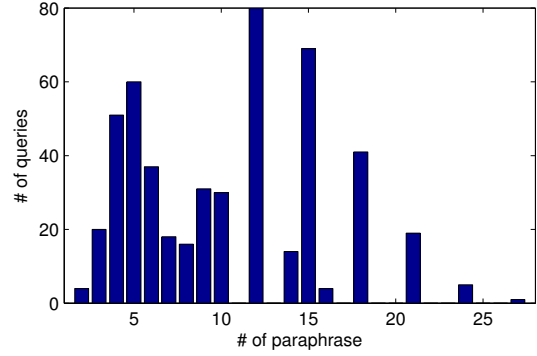


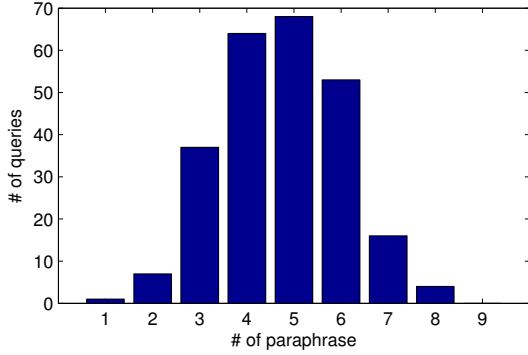
Figure 8: Topic commonness distribution of WEBQUESTIONS. Note that $x = -5$ indicates the commonness range $-6 \leq \log_{10}(p(q)) < -4$. $\log_{10}(p(e)) = 0$ means there is no topic entity, e.g., “What’s the smallest Spanish autonomous city?”



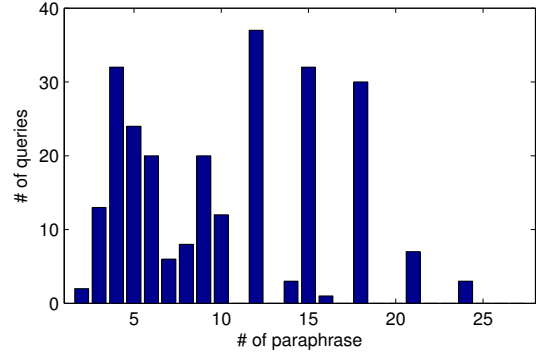
(a) Sentence paraphrase distribution of all queries



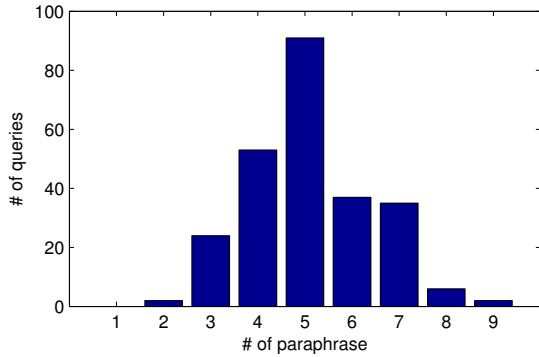
(b) Sentence+Entity paraphrase distribution of all queries



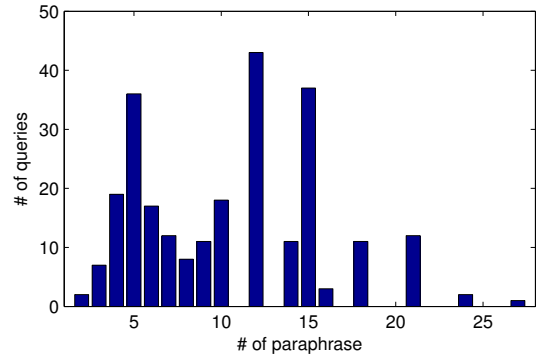
(c) Sentence paraphrase distribution of training queries



(d) Sentence+Entity paraphrase distribution of training queries



(e) Sentence paraphrase distribution of testing queries



(f) Sentence+Entity paraphrase distribution of testing queries

Figure 9: Characteristics distribution of GRAPHQUESTIONS: Paraphrase. The x -axis is the number of paraphrases, while the y -axis indicates how many graph queries have this number of paraphrases.

Question	Domain	Answer	# of edges	Function	$\log_{10}(p(q))$	A
how many scottish clans live in united kingdom ?	Scottish Clans	4	1	count	-12.11	1
give me the count of scottish clans in united kingdom .						
how many clans from scotland still exist in the uk ?						
what solid materials fuse at 435.4 joules/mole and above ?	Materials	Titanium, Beryllium, Aluminium oxide	1	comp.	-17.04	3
which materials need to take at least 435.4 joules of heat to fuse for one mole?						
name all the solid materials the fusion of which need at least 435.4 joules for a mole.						
who influenced paul the apostle ?	Influence	Jesus Christ	1	none	-8.93	1
by whom was paul the apostle influenced?						
who made a significant influence on paul ?						
find rockets made by chrysler group llc that support low earth orbit .	Spaceflight	Saturn I, Saturn IB	2	none	-25.76	2
which of chrysler group llc 's rockets are capable of low earth orbit ?						
which low earth orbit rockets are made by chrysler ?						
what is the nutritional composition of coca-cola soda?	Food	Sugar, Caffeine ...	2	none	-18.34	19
what is the supplement information for coca-cola ?						
what kind of nutrient does coke have?						
which tropical cyclone in the 2008 atlantic hurricane season caused the most fatalities?	Meteorology	Hurricane Hanna	2	super.	-29.36	1
which 2008 atlantic hurricane season 's tropical cyclone was the most deadly?						
which of the atlantic hurricane season 2008 's tropical cyclones killed the most people?						
people who are on a gluten-free diet can't eat what cereal grain that is used to make challah ?	Food	Wheat	3	none	-39.48	1
which cereal grain which can be utilized for making challah is unable to be consumed by those on a gluten-free diet ?						
what cereal grain can be used to produce challah , and people on gluten free could not eat?						
what's the theme of the casino having the most rooms under the control of caesars entertainment corporation ?	Casinos	Art Deco	3	super.	-34.11	1
what type of theme is caesars entertainment corporation 's largest casino made of?						
how is the largest casino owned by harrahs themed?						

Table 2: More example questions and characteristics. Topic entities are bold-faced. Three sentence-level paraphrases are shown for each graph query, with entity level paraphrasing applied on the third. Questions are lowercased.