Open-Vocabulary Semantic Parsing with both Distributional Statistics and Formal Knowledge

Matt Gardner and Jayant Krishnamurthy
Allen Institute for Artificial Intelligence
Seattle, Washington, USA
{mattg,jayantk}@allenai.org

Abstract

Traditional semantic parsers map language onto compositional, executable queries in a fixed schema. This mapping allows them to effectively leverage the information contained in large, formal knowledge bases (KBs, e.g., Freebase) to answer questions, but it is also fundamentally limiting—these semantic parsers can only assign meaning to language that falls within the KB's manually-produced schema. Recently proposed methods for open vocabulary semantic parsing overcome this limitation by learning execution models for arbitrary language, essentially using a text corpus as a kind of knowledge base. However, all prior approaches to open vocabulary semantic parsing *replace* a formal KB with textual information, making no use of the KB in their models. We show how to combine the disparate representations used by these two approaches, presenting for the first time a semantic parser that (1) produces compositional, executable representations of language, (2) can successfully leverage the information contained in both a formal KB and a large corpus, and (3) is not limited to the schema of the underlying KB. We demonstrate significantly improved performance over state-of-the-art baselines on an open-domain natural language question answering task.

1 Introduction

Semantic parsing is the task of mapping a phrase in natural language onto a formal query in some fixed schema, which can then be executed against a knowledge base (KB) (**?**; **?**). For example, the phrase "Who is the president of the United States?" might be mapped onto the query $\lambda(x)$./GOVERNMENT/PRESIDENT_OF(x, USA), which, when executed against Freebase (**?**), returns BARACK OBAMA. By mapping phrases to executable statements, semantic parsers can leverage large, curated sources of knowledge to answer questions (**?**).

This benefit comes with an inherent limitation, however—semantic parsers can only produce executable statements within their manually-produced schema. There is no query against Freebase that can answer questions like "Who are the Democratic front-runners in the US election?", as Freebase does not encode information about front-runners. Semantic parsers trained for Freebase fail on these kinds of questions.

To overcome this limitation, recent work has proposed methods for *open vocabulary semantic parsing*, which replace a formal KB with a probabilistic database learned from a text corpus. In these methods, language is mapped onto queries with predicates derived directly from the text itself (**?**; **?**). For instance, the question above might be mapped to $\lambda(x)$.*president_of*(x, USA). This query is not executable against any KB, however, and so open vocabulary semantic parsers must *learn* execution models for the predicates found in the text. They do this with a distributional approach similar to word embedding methods, giving them broad coverage, but lacking access to the large, curated KBs available to traditional semantic parsers.

Prior work in semantic parsing, then, has either had direct access to the information in a knowledge base, or broad coverage over all of natural language using the information in a large corpus, but not both.

In this work, we show how to combine these two approaches by incorporating KB information into open vocabulary semantic parsing models. Our key insight is that formal KB queries can be converted into *features* that can be added to the learned execution models of open vocabulary semantic parsers. This conversion allows open vocabulary models to use the KB fact /GOVERNMENT/PRESIDENT_OF(BARACKOBAMA, USA) when scoring *president_of*(BARACKOBAMA, USA), without requiring the model to map the language onto a single formal statement. Crucially, this featurization also allows the model to use these KB facts even when they only provide *partial* information about the language being modeled. For example, knowing that an entity is a POLITICIAN is very helpful information for deciding whether that entity is a front-runner. Our approach, outlined in Figure 1, effectively learns the meaning of a word as a distributional vector plus a *weighted combination of Freebase queries*, a considerably more expressive representation than those used by prior work.

While this combination is the main contribution of our work, we also present some small improvements that allow open vocabulary semantic parsing models to make better use of KB information when it is available: improving the logical forms generated by the semantic parser, and employing a simple technique from related work for generating candidate entities from the KB.

We demonstrate our approach on the task of answering

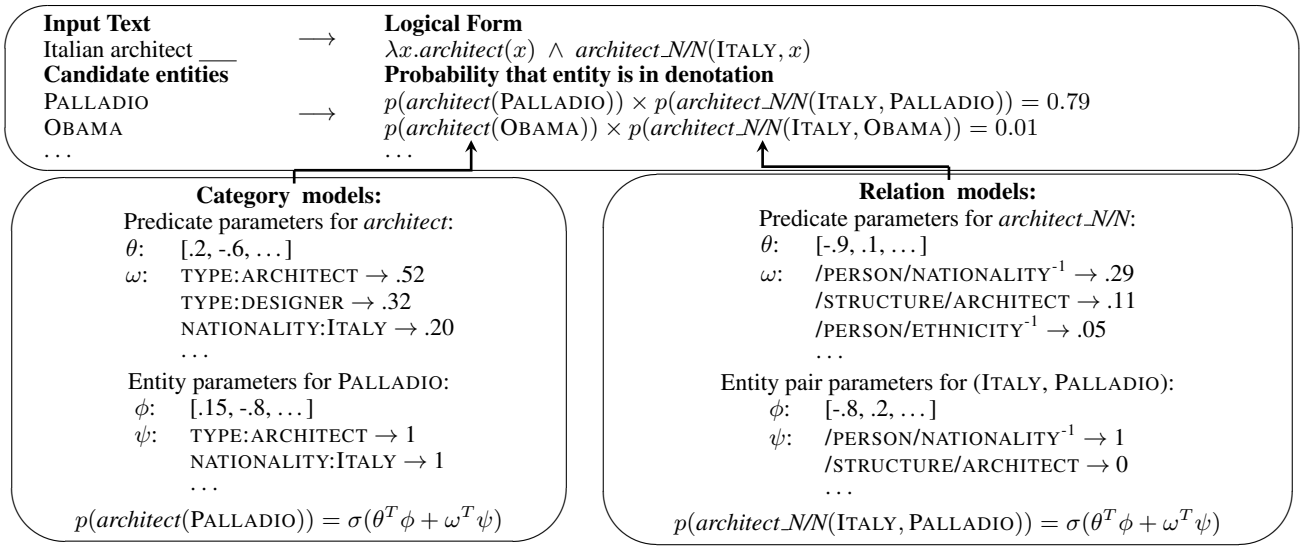


Figure 1: Overview of the components of our model. Given an input text, we use a CCG parser and an entity linker to produce a logical form with predicates derived from the text (shown in italics). For each predicate, we learn a distributional vector θ, as well as weights ω associated with a set of selected Freebase queries. For each entity and entity pair, we learn a distributional vector ϕ, and we extract a binary feature vector ψ from Freebase, indicating whether each entity or entity pair is in the set returned by the selected Freebase queries. These models are combined to assign probabilities to candidate entities.

open-domain fill-in-the-blank natural language questions. By giving open vocabulary semantic parsers direct access to KB information, we improve mean average precision on this task by over 120%.

2 Open vocabulary semantic parsing

In this section, we briefly describe the current state-of-the-art model for open vocabulary semantic parsing, introduced by Krishnamurthy and Mitchell (**?**). Instead of mapping text to Freebase queries, as done by a traditional semantic parser, their method parses text to a *surface logical form* with predicates derived directly from the words in the text (see Figure 1). Next, a distribution over denotations for each predicate is learned using a matrix factorization approach similar to that of Riedel et al. (**?**). This distribution is concisely represented using a probabilistic database, which also enables efficient probabilistic execution of logical form queries.

The matrix factorization has two sets of parameters: each category or relation has a learned k-dimensional embedding θ, and each entity or entity pair has a learned k-dimensional embedding ϕ. The probability assigned to a category instance $c(e)$ or relation instance $r(e_1, e_2)$ is given by:

$$p(c(e)) = \sigma(\theta_c^T \phi_e)$$
$$p(r(e_1, e_2)) = \sigma(\theta_r^T \phi_{(e_1,e_2)})$$

The probability of a predicate instance is the sigmoided inner product of the corresponding predicate and entity embeddings. Predicates with nearby embeddings will have similar distributions over the entities in their denotation. The parameters θ and ϕ are learned using a query ranking objective that optimizes them to rank entities observed in the denotation of a logical form above unobserved entities. Given the trained predicate and entity parameters, the system is capable of efficiently computing the marginal probability that an entity is an element of a logical form's denotation using approximate inference algorithms for probabilistic databases.

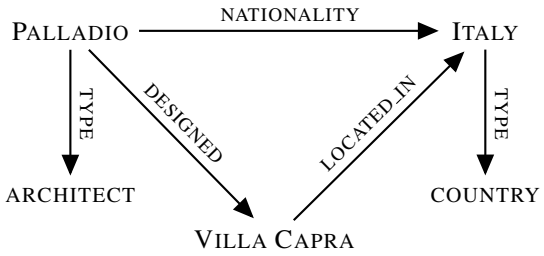
The model presented in this section is purely *distributional*, with predicate and entity models that draw only on co-occurrence information found in a corpus. In the following sections, we show how to augment this model with information contained in large, curated KBs such as Freebase.

3 Converting Freebase queries to features

Our key insight is that the executable queries used by traditional semantic parsers can be converted into features that provide KB information to the execution models of open vocabulary semantic parsers. Here we show how this is done.

Traditional semantic parsers map words onto distributions over executable queries, select one to execute, and return sets of entities or entity pairs from a KB as a result. Instead of executing a single query, we can simply execute *all* possible queries and use an entity's (or entity pair's) membership in each set as a feature in our predicate models.

There are two problems with this approach: (1) the set of all possible queries is intractably large, so we need a mechanism similar to a semantic parser's lexicon to select a small set of queries for each word; and (2) executing hundreds or thousands of queries at runtime for each predicate and entity is not computationally tractable. To solve these problems, we use a graph-based technique called subgraph feature extraction (SFE) (**?**).



Features between PALLADIO and ITALY:
⟨NATIONALITY⟩
⟨DESIGNED→LOCATED_IN⟩
Features for PALLADIO:
⟨NATIONALITY⟩
⟨NATIONALITY⟩:ITALY
⟨TYPE⟩:ARCHITECT
⟨DESIGNED→LOCATED_IN⟩
⟨DESIGNED→LOCATED_IN⟩:ITALY

Figure 2: A subset of the Freebase graph, and some example extracted features. The actual Freebase relations and entity identifiers used are modified here to aid readability.

3.1 Subgraph feature extraction

SFE is a technique for generating feature matrices over node pairs in graphs with labeled edges. When the graph corresponds to a formal KB such as Freebase, the features generated by SFE are isomorphic to statements in the KB schema (**?**). This means that we can use SFE to generate a feature vector for each entity (or entity pair) which succinctly captures the set of all statements[1] in whose denotations the entity (or entity pair) appears. Using this feature vector as part of the semantic parser's entity models solves problem (2) above, and performing feature selection for each predicate solves problem (1).

Some example features extracted by SFE are shown in Figure 2. For entity pairs, these features include the sequence of edges (or *paths*) connecting the nodes corresponding to the entity pair. For entities, these features include the set of paths connected to the node, optionally including the node at the end of the path. Note the correspondence between these features and Freebase queries: the path ⟨DESIGNED→LOCATED_IN⟩ can be executed as a query against Freebase, returning a set of (architect, location) entity pairs, where the architect designed a structure in the location. (PALLADIO, ITALY) is one such entity pair, so this pair has a feature value of 1 for this query.

3.2 Feature selection

The feature vectors produced by SFE contain tens of millions of possible formal statements. Out of these tens of millions of formal statements, only a handful represent relevant Freebase queries for any particular predicate. We therefore select a small number of statements to consider for each learned predicate in the open vocabulary semantic parser.

[1]In a restricted class, which contains Horn clauses and a few other things; see Gardner (**?**) for more details.

We select features by first summing the entity and entity pair feature vectors seen with each predicate in the training data. For example, the phrase "Italian architect Andrea Palladio" is considered a positive training example for the predicate instances *architect*(PALLADIO) and *architect_N/N*(ITALY, PALLADIO). We add the feature vectors for PALLADIO and (ITALY, PALLADIO) to the feature counts for the predicates *architect* and *architect_N/N*, respectively. This gives a set of counts $\text{COUNT}(\pi)$, $\text{COUNT}(f)$, and $\text{COUNT}(\pi \wedge f)$, for each predicate π and feature f. The features are then ranked by PMI for each predicate by computing $\frac{\text{COUNT}(\pi \wedge f)}{\text{COUNT}(\pi)\text{COUNT}(f)}$. After removing low-frequency features, we pick the $k = 100$ features with the highest PMI values for each predicate to use in our model.

4 Combined predicate models

Here we present our approach to incorporating KB information into open vocabulary semantic parsers. Having described how we use SFE to generate features corresponding to statements in a formal schema, adding these features to the models described in Section 2 is straightforward.

We saw in Section 2 that open vocabulary semantic parsers learn distributional vectors for each category, relation, entity and entity pair. We augment these vectors with the feature vectors described in Section 3. Each category and relation receives a weight ω for each selected Freebase query, and each entity and entity pair has an associated feature vector ψ. The truth probability of a category instance $c(e)$ or relation instance $r(e_1, e_2)$ is thus given by:

$$p(c(e)) = \sigma(\theta_c^T \phi_e + \omega_c^T \psi_c(e))$$
$$p(r(e_1, e_2)) = \sigma(\theta_r^T \phi_{(e_1,e_2)} + \omega_r^T \psi_r(e_1, e_2))$$

In these equations, θ and ϕ are learned predicate and entity embeddings, as described in Section 2. The second term in the sum represents our new features and their learned weights. $\psi_c(e)$ and $\psi_r(e_1, e_2)$ are SFE feature vectors for each entity and entity pair; a different set of features is chosen for each predicate c and r, as described in Section 3.2. ω_c and ω_r are learned weights for these features.

In our model, there are now three sets of parameters to be learned: (1) θ, low-dimensional distributional vectors trained for each predicate; (2) ϕ, low-dimensional distributional vectors trained for each entity and entity pair; and (3) ω, weights associated with the selected formal SFE features for each predicate. All of these parameters are optimized jointly, using the same method described in Section 2.

Note here that each SFE feature corresponds to a query over the formal schema, defining a set of entities (or entity pairs). The associated feature weight measures the likelihood that an entity in this set is also in the denotation of the surface predicate. Our models include *many* such features for each surface predicate, effectively mapping each surface predicate onto a *weighted combination of Freebase queries*.

5 Making full use of KB information

In addition to improving predicate models, as just described, adding KB information to open vocabulary semantic parsers

suggests two other simple improvements: (1) using more specific logical forms, and (2) generating candidate entities from the KB.

5.1 Logical form generation

Krishnamurthy and Mitchell (**?**) generate logical forms from natural language statements by computing a syntactic CCG parse, then applying a collection of rules to produce logical forms. However, their logical form analyses do not model noun-mediated relations well. For example, given the phrase "Italian architect Andrea Palladio," their system's logical form would include the relation *N/N*(ITALY, PALLADIO). Here, the *N/N* predicate represents a generic noun modifier relation; however, this relation is too vague for the predicate model to accurately learn its denotation. A similar problem occurs with prepositions and possessives, e.g., it is similarly hard to learn the denotation of the predicate *of*.

Our system improves the analysis of noun-mediated relations by simply including the noun in the predicate name. In the architect example above, our system produces the relation *architect_N/N*. It does this by concatenating all intervening noun modifiers between two entity mentions and including them in the predicate name; for example, "Illinois attorney general Lisa Madigan" produces the predicate *attorney_general_N/N*. We similarly improve the analyses of prepositions and possessives to include the head noun. For example, "Barack Obama, president of the U.S." produces the predicate instance *president_of*(BARACK OBAMA, U.S.), and "Rome, Italy's capital" produces the predicate *'s_capital*. This process generates more specific predicates that more closely align with the KB facts that we make available to the predicate models.

5.2 Candidate entity generation

A key benefit of our predicate models is that they are able to assign scores to entity pairs that were never seen in the training data. Distributional models have no learned vectors for these entity pairs and therefore assume $p(r(e_1, e_2)) = 0$ for unseen entity pairs (e_1, e_2). This limits the recall of these models when applied to question answering, as entity pairs will not have been observed for many correct, but rare entity answers. In contrast, because our models have access to a large KB, the formal component of the model can always give a score to any entity pair in the KB. This allows our model to considerably improve question answering performance on rare entities.

It would be computationally intractable to consider *all* Freebase entities as answers to queries, and so we use a simple candidate entity generation technique to consider only a small set of likely entities for a given query. We first find all entities in the query, and consider as candidates any entity that has either been seen at training time with a query entity or is directly connected to a query entity in Freebase.[2] This candidate entity generation is common practice for recent question answering models over Freebase (**?**), though, for the reasons stated above, it has not been used previously in open vocabulary semantic parsing models.

[2]Or connected by a mediator node, which is how Freebase represents relations with more than two arguments.

6 Evaluation

We evaluate our open-vocabulary semantic parser on a fill-in-the-blank natural language query task. Each test example is a natural language phrase containing at least two Freebase entities, one of which is held out. The system must propose a ranked list of Freebase entities to fill in the blank left by the held out entity, and the predicted entities are then judged manually for correctness. We compare our proposed models, which combine distributional and formal elements, with a purely distributional baseline from prior work. All of the data and code used in these experiments is available at http://github.com/allenai/open_vocab_semparse.

6.1 Data

Much recent work on semantic parsing has been evaluated using the WebQuestions dataset (**?**). This dataset is not suitable for evaluating our model because it was filtered to only questions that are mappable to Freebase queries. In contrast, our focus is on language that is *not* directly mappable to Freebase. We thus use the dataset introduced by Krishnamurthy and Mitchell (**?**), which consists of the ClueWeb09 web corpus[3] along with Google's FACC entity linking of that corpus to Freebase (**?**). For training data, 3 million webpages from this corpus were processed with a CCG parser to produce logical forms (**?**). This produced 2.1m predicate instances involving 142k entity pairs and 184k entities. After removing infrequently-seen predicates (seen fewer than 6 times), there were 25k categories and 4.2k relations.

We also used the test set created by Krishnamurthy and Mitchell, which contains 220 queries generated in the same fashion as the training data from a separate section of ClueWeb. However, as they did not release a development set with their data, we used this set as a development set. For a final evaluation, we generated another, similar test set from a different held out section of ClueWeb, in the same fashion as done by Krishnamurthy and Mitchell. This final test set contains 307 queries.

6.2 Models

We compare three models in our experiments: (1) the distributional model of Krishnamurthy and Mitchell, described in Section 2, which is the current state-of-the-art method for open vocabulary semantic parsing; (2) a formal model (new to this work), where the distributional parameters θ and ϕ in Section 4 are fixed at zero; and (3) the combined model described in Section 4 (also new to this work). In each of these models, we used vectors of size 300 for all embeddings. Except where noted, all experiments use our modified logical forms (Section 5.1) and our entity proposal mechanism (Section 5.2). We do not compare against any traditional semantic parsers, as more than half of the questions in our dataset are not answerable by Freebase queries, and so are out of scope for those parsers (**?**).

[3]http://www.lemuproject.org/clueweb09.php

Model	K&M's LFs	Our LFs	Delta
Distributional (K&M 2015)	.269	**.284**	+.015
Formal	.231	**.276**	+.045
Combined	.313	**.335**	+.022

Table 1: Improvement in mean average precision when using our logical forms on the development set.

Model	MAP	W-MAP	MRR
Distributional (K&M 2015)	.163	.163	.288
With freebase candidates	**.229**	**.275**	**.312**
Relative improvement	40%	69%	8%

Table 2: Improvement to the distributional model when using our candidate entity generation.

6.3 Methodology

Given a fill-in-the-blank query such as "Italian architect ___", each system produces a ranked list of 100 candidate entities. To compare the output of the systems, we follow a pooled evaluation protocol commonly used in relation extraction and information retrieval (**?**; **?**). We take the top 30 predictions from each system and manually annotate whether they are correct, and use those annotations to compute the average precision (AP) and reciprocal rank (RR) of each system on the query. Average precision is defined as $\frac{1}{m}\sum_{k=1}^{m}\mathrm{Prec}(k)\times\mathrm{Correct}(k)$, where $\mathrm{Prec}(k)$ is the precision at rank k, $\mathrm{Correct}(k)$ is an indicator function for whether the kth answer is correct, and m is number of returned answers (up to 100 in this evaluation). AP is equivalent to calculating the area under a precision-recall curve. Reciprocal rank is computed by first finding the rank r of the first correct prediction made by a system. Reciprocal rank is then $\frac{1}{r}$, ranging from 1 (if the first prediction is correct) to 0 (if there is no correct answer returned). In the tables below we report *mean* average precision (MAP) and *mean* reciprocal rank (MRR), averaged over all of the queries in the test set. We also report a weighted version of MAP, where the AP of each query is scaled by the number of annotated correct answers to the query (shown as W-MAP in the tables for space considerations).

6.4 Results

We first show the effect of the new logical forms introduced in Section 5.1. As can be seen in Table 1, with our improved logical forms, all models are better able to capture the semantics of language. This improvement is most pronounced in the formal models, which have more capacity to get specific features from Freebase with the new logical forms. As our logical forms give all models better performance, the remaining experiments we present all use these logical forms.

We next show the improvement gained by using the simple candidate entity generation outlined in Section 5.2. By simply appending the list of connected entities in Freebase to the end of the rankings returned by the distributional model, MAP improves by 40% (see Table 2). The connectedness of an entity pair in Freebase is very informative, especially for rare entities that are not seen together during training.

Model	MAP	W-MAP	MRR
Distributional (K&M 2015)	.284	.371	.379
Formal	.276	.469	.334
Combined	**.335**	**.477**	**.429**
Relative improvement	18%	29%	13%

Table 3: Development set results for our fill-in-the-blank task. The combined model significantly improves MAP over prior work.

Model	MAP	W-MAP	MRR
Distributional (K&M 2015)	.229	.275	.312
Formal	.355	.495	.419
Combined	**.370**	**.513**	**.469**
Relative improvement	62%	87%	50%

Table 4: Final test results set for our fill-in-the-blank task. The combined model improves over prior work by 50–87% on our metrics. These improvements over the baseline are *after* the baseline has been improved by the methods developed in this paper, shown in Table 1 and Table 2. The cumulative effect of the methods presented in this work is an improvement of over 120% in MAP.

Table 3 shows a comparison between the semantic parsing models on the development set. As can be seen, the combined model significantly improves performance over prior work, giving a relative gain in weighted MAP of 29%.

Table 4 shows that these improvements are consistent on the final test set, as well. The performance improvement seen by the combined model is actually larger on this set, with gains on our metrics ranging from 50% to 87%.

On both of these datasets, the difference in MAP between the combined model and the distributional model is statistically significant (by a paired permutation test, $p < 0.05$). The differences between the combined model and the formal model, and between the formal model and the distributional model, are not statistically significant, as each method has certain kinds of queries that it performs well on. Only the combined model is able to consistently outperform the distributional model on all kinds of queries.

6.5 Discussion

Our model tends to outperform the distributional model on queries containing predicates with exact or partial correlates in Freebase. For example, our model obtains nearly perfect average precision on the queries "French newspaper ___" and "Israeli prime minister ___," both of which can be exactly expressed in Freebase. The top features for *newspaper*(x) all indicate that x has type NEWSPAPER in Freebase, and the top features for *newspaper_N/N*(x, y) indicate that y is a newspaper, and that x is either the circulation area of y or the language of y.

The model also performs well on queries with partial Freebase correlates, such as "Microsoft head honcho ___", "The United States, ___'s closest ally", and "Patriots linebacker ___," although with somewhat lower average precision. The high weight features in these cases tend to provide useful hints, even though there is no direct correlate; for example, the model learns that "honchos" are people, and that they tend to be CEOs and film producers. There are also some areas where our model can be improved. First, in some cases, the edge sequence features used by the model are not expressive enough to identify the correct relation in Freebase. An example of this problem is the "linebacker" example above, where the features for *linebacker_N/N* can capture which athletes play for which teams, but not the *positions* of those athletes. Second, our model can under-perform on predicates with no close mapping to Freebase. An example where this problem occurs is the query "___ is a NASA mission." Third, there remains room to further improve the logical forms produced by the semantic parser, specifically for multi-word expressions. One problem occurs with multi-word noun modifiers, e.g., "Vice president Al Gore" is mapped to $\mathit{vice}(\text{AL GORE}) \wedge \mathit{president}(\text{AL GORE})$. Another problem is that there is no back-off with multi-word relations. For example, the predicate *head_honcho_N/N* was never seen in the training data, so it is replaced with *unknown*; however, it would be better to replace it with *honcho_N/N*, which *was* seen in the training data. Finally, although using connected entities in Freebase as additional candidates during inference is helpful, it often over- or under-generates candidates. A more tailored, per-query search process could improve performance.

7 Related work

There is an extensive literature on building semantic parsers to answer questions against a KB (**?**; **?**; **?**; **?**). Some of this work has used surface (or *ungrounded*) logical forms as an intermediate representation, similar to our work (**?**; **?**; **?**; **?**). The main difference between our work and these techniques is that they map surface logical forms to a single executable Freebase query, while we learn execution models for the surface logical forms directly, using a weighted combination of Freebase queries as part of the model. None of these prior works can assign meaning to language that is not directly representable in the KB schema. Choi, Kwiatkowski and Zettlemoyer (**?**) presented an information extraction system that performs a semantic parse of open-domain text, recognizing when a predicate cannot be mapped to Freebase. However, while they recognize when a predicate is not mappable to Freebase, they do not attempt to learn execution models for those predicates, nor can they answer questions using those predicates.

Yao and Van Durme (**?**) and Dong et al. (**?**) proposed question answering models that use similar features to those used in this work. However, they did not produce semantic parses of language, instead using methods that are non-compositional and do not permit complex queries. Finally, learning probabilistic databases in an open vocabulary semantic parser has a strong connection with KB completion. In addition to SFE (**?**), our work draws on work on embedding the entities and relations in a KB (**?**; **?**; **?**; **?**; **?**), as well as work on graph-based methods for reasoning with KBs (**?**; **?**; **?**; **?**). Our combination of embedding methods with graph-based methods in this paper is suggestive of how one could combine the two in methods for KB completion. Initial work exploring this direction has already been done by Toutanova and Chen (**?**).

8 Conclusion

Prior work in semantic parsing has either leveraged large knowledge bases to answer questions, or used distributional techniques to gain broad coverage over all of natural language. In this paper, we have shown how to gain both of these benefits by converting the queries generated by traditional semantic parsers into features which are then used in open vocabulary semantic parsing models. We presented a technique to do this conversion in a way that is scalable using graph-based feature extraction methods. Our combined model achieved relative gains of over 50% in mean average precision and mean reciprocal rank versus a purely distributional approach. We also introduced a better mapping from surface text to logical forms, and a simple method for using a KB to find candidate entities during inference. Taken together, the methods introduced in this paper improved mean average precision on our task from .163 to .370, a 127% relative improvement over prior work. This work suggests a new direction for semantic parsing research. Existing semantic parsers map language to a single KB query, an approach that successfully leverages a KB's predicate instances, but is fundamentally limited by its schema. In contrast, our approach maps language to a *weighted combination of queries* plus a distributional component; this approach is capable of representing a much broader class of concepts while still using the KB when it is helpful. Furthermore, it is capable of using the KB even when the meaning of the language cannot be exactly represented by a KB predicate, which is a common occurrence. We believe that this kind of approach could significantly expand the applicability of semantic parsing techniques to more complex domains where the assumptions of traditional techniques are too limiting. We are actively exploring applying these techniques to science question answering (**?**), for example, where existing KBs provide only partial coverage of the questions.

Quas veritatis voluptatem voluptate nulla ut error dolores temporibus fugiat, earum eius voluptatum?Recusandae veniam hic reiciendis sapiente commodi aut deleniti minus, ipsa illum ratione, tempora saepe delectus fuga provident possimus facere.Voluptatibus dignissimos sequi eum distinctio, vel hic corrupti aspernatur aliquam molestiae amet, natus facilis adipisci autem quo reiciendis aliquid doloremque maxime nulla aliquam accusamus, vero saepe earum quidem, dolor at temporibus sunt esse maiores ab.Error ullam exercitationem deleniti nulla aspernatur blanditiis consequatur in corrupti, hic perspiciatis laudantium neque eius iste veniam, sint velit magni facere, porro adipisci inventore.Unde reiciendis dolorum eveniet laborum harum id repellendus mollitia, repellendus quae animi modi tempore iste numquam assumenda, quia commodi suscipit minima maiores cum repellendus quos incidunt blanditiis, dignissimos cupiditate libero quod, voluptate maiores error aspernatur optio accusantium eaque

sint tempore.Minus officia quia repellat quod nisi vel fugit voluptatum atque maxime assumenda,