Paraphrase Generation from Latent-Variable PCFGs for Semantic Parsing

Paraphrase Generation from Latent-Variable PCFGs for Semantic Parsing

Shashi Narayan, Siva Reddy and Shay B. Cohen
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh, EH8 9LE, UK
shashi.narayan@ed.ac.uk, siva.reddy@ed.ac.uk, scohen@inf.ed.ac.uk

Abstract

One of the limitations of semantic parsing approaches to open-domain question answering is the lexicosyntactic gap between natural language questions and knowledge base entries – there are many ways to ask a question, all with the same answer. In this paper we propose to bridge this gap by generating paraphrases of the input question with the goal that at least one of them will be correctly mapped to a knowledge-base query. We introduce a novel grammar model for paraphrase generation that does not require any sentence-aligned paraphrase corpus. Our key idea is to leverage the flexibility and scalability of latent-variable probabilistic context-free grammars to sample paraphrases. We do an extrinsic evaluation of our paraphrases by plugging them into a semantic parser for Freebase. Our evaluation experiments on the WebQuestions benchmark dataset show that the performance of the semantic parser improves over strong baselines.

1 Introduction

Semantic parsers map sentences onto logical forms that can be used to query databases (Zettlemoyer and Collins, 2005; Wong and Mooney, 2006), instruct robots (Chen and Mooney, 2011), extract information (Krishnamurthy and Mitchell, 2012), or describe visual scenes (Matuszek et al., 2012). In this paper we consider the problem of semantically parsing questions into Freebase logical forms for the goal of question answering. Current systems accomplish this by learning task-specific grammars (Berant et al., 2013), strongly-typed CCG grammars (Kwiatkowski et al., 2013; Reddy et al., 2014), or neural networks without requiring any grammar (Yih et al., 2015). These methods are sensitive to the words used in a question and their word order, making them vulnerable to unseen words and phrases. Furthermore, mismatch between natural language and Freebase makes the problem even harder. For example, Freebase expresses the fact that *"Czech is the official language of Czech Republic"* (encoded as a graph), whereas to answer a question like *"What do people in Czech Republic speak?"* one should infer *people in Czech Republic* refers to *Czech Republic* and *What* refers to the *language* and *speak* refers to the predicate *official language*.

We address the above problems by using paraphrases of the original question. Paraphrasing has shown to be promising for semantic parsing (Fader et al., 2013; Berant and Liang, 2014; Wang et al., 2015). We propose a novel framework for paraphrasing using latent-variable PCFGs (L-PCFGs). Earlier approaches to paraphrasing used phrase-based machine translation for text-based QA (Duboue and Chu-Carroll, 2006; Riezler et al., 2007), or hand annotated grammars for KB-based QA (Berant and Liang, 2014). We find that phrase-based statistical machine translation (MT) approaches mainly produce lexical paraphrases without much syntactic diversity, whereas our grammar-based approach is capable of producing both lexically and syntactically diverse paraphrases. Unlike MT based approaches, our system does not require aligned parallel paraphrase corpora. In addition we do not require hand annotated grammars for paraphrase generation but instead learn the grammar directly from a large scale question corpus.

The main contributions of this paper are two fold. First, we present an algorithm (§2) to generate paraphrases using latent-variable PCFGs. We use the spectral method of Narayan and Cohen (2015) to estimate L-PCFGs on a large scale question treebank. Our grammar model leads to a robust and an efficient system for paraphrase generation in open-domain question answering. While CFGs have been explored for paraphrasing using bilingual parallel corpus (Ganitkevitch et al., 2013), ours is the first implementation of CFG that uses only monolingual data. Second, we show that generated paraphrases can be used to improve semantic parsing of questions into Freebase logical forms (§3). We build on a strong baseline of Reddy et al. (2014) and show that our grammar model competes with MT baseline even without using any parallel paraphrase resources.

2 Paraphrase Generation Using Grammars

Our paraphrase generation algorithm is based on a model in the form of an L-PCFG. L-PCFGs are PCFGs where the nonterminals are refined with latent states that provide some contextual information about each node in a given derivation. L-PCFGs have been used in various ways, most commonly for syntactic parsing (Prescher, 2005; Matsuzaki et al., 2005; Petrov et al., 2006; Cohen et al., 2013; Narayan and Cohen, 2015; Narayan and Cohen, 2016).

In our estimation of L-PCFGs, we use the spectral method of Narayan and Cohen (2015), instead of using EM, as has been used in the past by Matsuzaki et al. (2005) and Petrov et al. (2006). The spectral method we use enables the choice of a set of feature functions that indicate the latent states, which proves to be useful in our case. It also leads to sparse grammar estimates and compact models.

The spectral method works by identifying feature functions for "inside" and "outside" trees, and then clusters them into latent states. Then it follows with a maximum likelihood estimation step, that assumes the latent states are represented by clusters obtained through the feature function clustering. For more details about these constructions, we refer the reader to Cohen et al. (2013) and Narayan and Cohen (2015).

The rest of this section describes our paraphrase generation algorithm.

2.1 Paraphrases Generation Algorithm

We define our paraphrase generation task as a sampling problem from an L-PCFG G_{syn}, which is estimated from a large corpus of parsed questions. Once this grammar is estimated, our algorithm follows a pipeline with two major steps.

We first build a word lattice W_q for the input question q.[1] We use the lattice to constrain our paraphrases to a specific choice of words and phrases that can be used. Once this lattice is created, a grammar G'_{syn} is then extracted from G_{syn}. This grammar is constrained to the lattice.

We experiment with three ways of constructing word lattices: naïve word lattices representing the words from the input question only, word lattices constructed with the Paraphrase Database (Ganitkevitch et al., 2013) and word lattices constructed with a bi-layered L-PCFG, described in §2.2. For example, Figure 1 shows an example word lattice for the question *What language do people in Czech Republic speak?* using the lexical and phrasal rules from the PPDB.[2]

Once G'_{syn} is generated, we sample paraphrases of the input question q. These paraphrases are further filtered with a classifier to improve the precision of the generated paraphrases.

L-PCFG Estimation We train the L-PCFG G_{syn} on the Paralex corpus (Fader et al., 2013). Paralex is a large monolingual parallel corpus, containing 18 million pairs of question paraphrases with 2.4M distinct questions in the corpus. It is suitable for our task of generating paraphrases since its large scale makes our model robust for open-domain questions. We construct a treebank by parsing 2.4M distinct questions from Paralex using the BLLIP parser (Charniak and Johnson, 2005).[3]

Given the treebank, we use the spectral algorithm of Narayan and Cohen (2015) to learn an L-PCFG

[1] Word lattices, formally weighted finite state automata, have been used in previous works for paraphrase generation (Langkilde and Knight, 1998; Barzilay and Lee, 2003; Pang et al., 2003; Quirk et al., 2004). We use an unweighted variant of word lattices in our algorithm.

[2] For our experiments, we extract rules from the PPDB-Small to maintain the high precision (Ganitkevitch et al., 2013).

[3] We ignore the Paralex alignments for training G_{syn}.

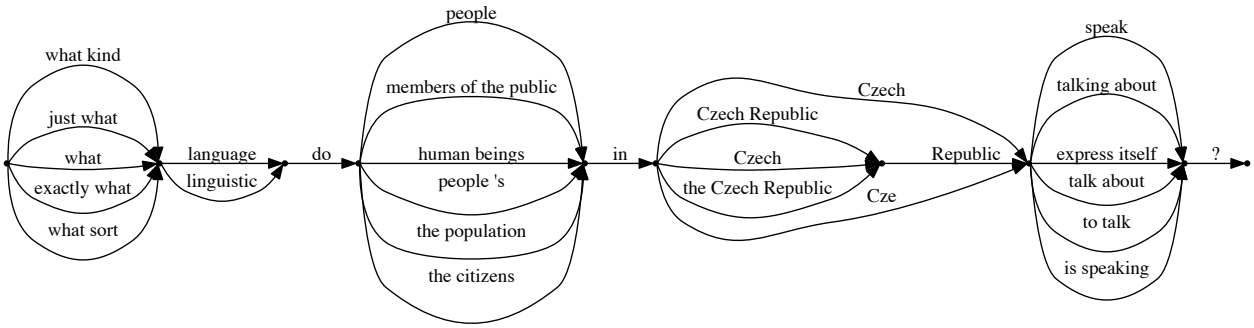


Figure 1: An example word lattice for the question *What language do people in Czech Republic speak?* using the lexical and phrasal rules from the PPDB.

for constituency parsing to learn G_{syn}. We follow Narayan and Cohen (2015) and use the same feature functions for the inside and outside trees as they use, capturing contextual syntactic information about nonterminals. We refer the reader to Narayan and Cohen (2015) for more detailed description of these features. In our experiments, we set the number of latent states to 24.

Once we estimate G_{syn} from the Paralex corpus, we restrict it for each question to a grammar G'_{syn} by keeping only the rules that could lead to a derivation over the lattice. This step is similar to lexical pruning in standard grammar-based generation process to avoid an intermediate derivation which can never lead to a successful derivation (Koller and Striegnitz, 2002; Narayan and Gardent, 2012).

Paraphrase Sampling Sampling a question from the grammar G'_{syn} is done by recursively sampling nodes in the derivation tree, together with their latent states, in a top-down breadth-first fashion. Sampling from the pruned grammar G'_{syn} raises an issue of oversampling words that are more frequent in the training data. To lessen this problem, we follow a *controlled sampling* approach where sampling is guided by the word lattice W_q. Once a word w from a path e in W_q is sampled, all other parallel or conflicting paths to e are removed from W_q. For example, generating for the word lattice in Figure 1, when we sample the word *citizens*, we drop out the paths *"human beings"*, *"people's"*, *"the population"*, *"people"* and *"members of the public"* from W_q and accordingly update the grammar. The controlled sampling ensures that each sampled question uses words from a single start-to-end path in W_q. For example, we could sample a question *what is Czech Republic 's language?* by sampling words from the path *(what, language, do, people 's, in, Czech, Republic, is speaking, ?)* in Figure 1. We repeat this sampling process to generate multiple potential paraphrases.

The resulting generation algorithm has multiple advantages over existing grammar generation methods. First, the sampling from an L-PCFG grammar lessens the lexical ambiguity problem evident in lexicalized grammars such as tree adjoining grammars (Narayan and Gardent, 2012) and combinatory categorial grammars (White, 2004). Our grammar is not lexicalized, only unary context-free rules are lexicalized. Second, the top-down sampling restricts the combinatorics inherent to bottom-up search (Shieber et al., 1990). Third, we do not restrict the generation by the order information in the input. The lack of order information in the input often raises the high combinatorics in lexicalist approaches (Kay, 1996). In our case, however, we use sampling to reduce this problem, and it allows us to produce syntactically diverse questions. And fourth, we impose no constraints on the grammar thereby making it easier to maintain bi-directional (recursive) grammars that can be used both for parsing and for generation (Shieber, 1988).

2.2 Bi-Layered L-PCFGs

As mentioned earlier, one of our lattice types is based on bi-layered PCFGs introduced here.

In their traditional use, the latent states in L-PCFGs aim to capture syntactic information. We introduce here the use of an L-PCFG with two layers of latent states: one layer is intended to capture the usual syntactic information, and the other aims to capture semantic and topical information by using a

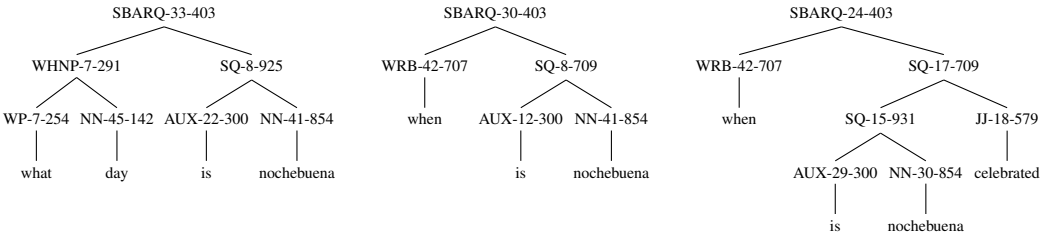


Figure 2: Trees used for bi-layered L-PCFG training. The questions *what day is nochebuena*, *when is nochebuena* and *when is nochebuena celebrated* are paraphrases from the Paralex corpus. Each nonterminal is decorated with a syntactic label and two identifiers, e.g., for WP-7-254, WP is the syntactic label assigned by the BLLIP parser, 7 is the syntactic latent state, and 254 is the semantic latent state.

large set of states with specific feature functions.[4]

To create the bi-layered L-PCFG, we again use the spectral algorithm of Narayan and Cohen (2015) to estimate a grammar G_{par} from the Paralex corpus. We use the word alignment of paraphrase question pairs in Paralex to map inside and outside trees of each nonterminals in the treebank to bag of word features. The number of latent states we use is 1,000.

Once the two feature functions (syntactic in G_{syn} and semantic in G_{par}) are created, each nonterminal in the training treebank is assigned two latent states (cluster identifiers). Figure 2 shows an example annotation of trees for three paraphrase questions from the Paralex corpus. We compute the parameters of the bi-layered L-PCFG G_{layered} with a simple frequency count maximum likelihood estimate over this annotated treebank. As such, G_{layered} is a combination of G_{syn} and G_{par}, resulting in 24,000 latent states (24 syntactic x 1000 semantic).

Consider an example where we want to generate paraphrases for the question *what day is nochebuena*. Parsing it with G_{layered} will lead to the leftmost hybrid structure as shown in Figure 2. The assignment of the first latent states for each nonterminals ensures that we retrieve the correct syntactic representation of the sentence. Here, however, we are more interested in the second latent states assigned to each nonterminals which capture the paraphrase information of the sentence at various levels. For example, we have a unary lexical rule `(NN-*-142 day)` indicating that we observe *day* with `NN` of the paraphrase type `142`. We could use this information to extract unary rules of the form `(NN-*-142` w`)` in the treebank that will generate words w which are paraphrases to *day*. Similarly, any node `WHNP-*-291` in the treebank will generate paraphrases for *what day*, `SBARQ-*-403`, for *what day is nochebuena*. This way we will be able to generate paraphrases *when is nochebuena* and *when is nochebuena celebrated* as they both have `SBARQ-*-403` as their roots.[5]

To generate a word lattice W_q for a given question q, we parse q with the bi-layered grammar G_{layered}. For each rule of the form X-m_1-$m_2 \rightarrow w$ in the bi-layered tree with $X \in \mathcal{P}$, $m_1 \in \{1, \ldots, 24\}$, $m_2 \in \{1, \ldots, 1000\}$ and w a word in q, we extract rules of the form X-*-$m_2 \rightarrow w'$ from G_{layered} such that $w' \neq w$. For each such (w, w'), we add a path w' parallel to w in the word lattice.

2.3 Paraphrase Classification

Our sampling algorithm overgenerates paraphrases which are incorrect. To improve its precision, we build a binary classifier to filter the generated paraphrases. We randomly select 100 distinct questions from the Paralex corpus and generate paraphrases using our generation algorithm with various lattice settings. We randomly select 1,000 pairs of input-sampled sentences and manually annotate them as "correct" or "incorrect" paraphrases.[6] We train our classifier on this manually created training data.[7] We

[4]For other cases of separating syntax from semantics in a similar way, see Mitchell and Steedman (2015).

[5]We found out that our G_{par} grammar is not fine-grained enough and often merges different paraphrase information into the same latent state. This problem is often severe for nonterminals at the top level of the bilayered tree. Hence, we rely only on unary lexical rules (the rules that produce terminal nodes) to extract paraphrase patterns in our experiments.

[6]We have 154 positive and 846 negative paraphrase pairs.

[7]We do not use the paraphrase pairs from the Paralex corpus to train our classifier, as they do not represent the distribution of our sampled paraphrases and the classifier trained on them performs poorly.

follow Madnani et al. (2012), who used MT metrics for paraphrase identification, and experiment with 8 MT metrics as features for our binary classifier. In addition, we experiment with a binary feature which checks if the sampled paraphrase preserves named entities from the input sentence. We use WEKA (Hall et al., 2009) to replicate the classifier of Madnani et al. (2012) with our new feature. We tune the feature set for our classifier on the development data.

3 Semantic Parsing using Paraphrasing

In this section we describe how the paraphrase algorithm is used for converting natural language to Freebase queries. Following Reddy et al. (2014), we formalize the semantic parsing problem as a graph matching problem, i.e., finding the Freebase subgraph (grounded graph) that is isomorphic to the input question semantic structure (ungrounded graph).

This formulation has a major limitation that can be alleviated by using our paraphrase generation algorithm. Consider the question *What language do people in Czech Republic speak?*. The ungrounded graph corresponding to this question is shown in Figure 3(a). The Freebase grounded graph which results in correct answer is shown in Figure 3(d). Note that these two graphs are non-isomorphic making it impossible to derive the correct grounding from the ungrounded graph. In fact, at least 15% of the examples in our development set fail to satisfy isomorphic assumption. In order to address this problem, we use paraphrases of the input question to generate additional ungrounded graphs, with the aim that one of those paraphrases will have a structure isomorphic to the correct grounding. Figure 3(b) and Figure 3(c) are two such paraphrases which can be converted to Figure 3(d) as described in §3.2.

For a given input question, first we build ungrounded graphs from its paraphrases. We convert these graphs to Freebase graphs. To learn this mapping, we rely on manually assembled question-answer pairs. For each training question, we first find the set of *oracle* grounded graphs—Freebase subgraphs which when executed yield the correct answer—derivable from the question's ungrounded graphs. These oracle graphs are then used to train a structured perceptron model. These steps are discussed in detail below.

3.1 Ungrounded Graphs from Paraphrases

We use GRAPHPARSER (Reddy et al., 2014) to convert paraphrases to ungrounded graphs. This conversion involves three steps: 1) parsing the paraphrase using a CCG parser to extract syntactic derivations (Lewis and Steedman, 2014), 2) extracting logical forms from the CCG derivations (Bos et al., 2004), and 3) converting the logical forms to an ungrounded graph.[8] The ungrounded graph for the example question and its paraphrases are shown in Figure 3(a), Figure 3(b) and Figure 3(c), respectively.

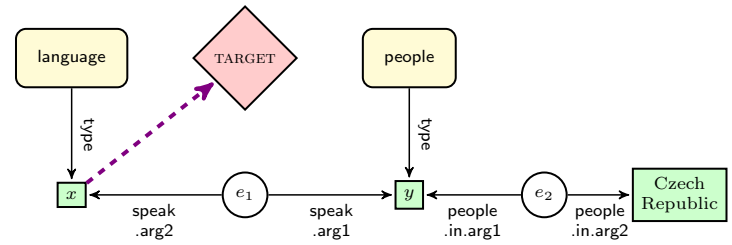
3.2 Grounded Graphs from Ungrounded Graphs

The ungrounded graphs are grounded to Freebase subgraphs by mapping entity nodes, entity-entity edges and entity type nodes in the ungrounded graph to Freebase entities, relations and types, respectively. For example, the graph in Figure 3(b) can be converted to a Freebase graph in Figure 3(d) by replacing the entity node *Czech Republic* with the Freebase entity CZECHREPUBLIC, the edge *(speak.arg$_2$, speak.in)* between x and *Czech Republic* with the Freebase relation *(location.country.official_language.2, location.country.official_language.1)*, the type node *language* with the Freebase type *language.human_language*, and the TARGET node remains intact. The rest of the nodes, edges and types are grounded to *null*. In a similar fashion, Figure 3(c) can be grounded to Figure 3(d), but not Figure 3(a) to Figure 3(d). If no paraphrase is isomorphic to the target grounded grounded graph, our grounding fails.

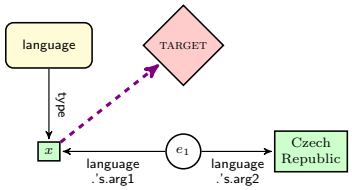
3.3 Learning

We use a linear model to map ungrounded graphs to grounded ones. The parameters of the model are learned from question-answer pairs. For example, the question *What language do people in Czech Republic speak?* paired with its answer {CZECHLANGUAGE}. In line with most work on question answering against Freebase, we do not rely on annotated logical forms associated with the question for training and treat the mapping of a question to its grounded graph as latent.

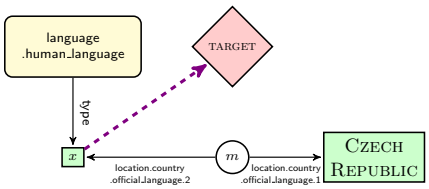
[8] Please see Reddy et al. (2014) for more details.



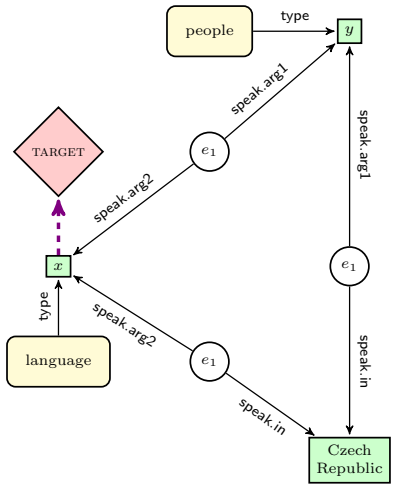
(a) Input sentence: What language do people in Czech Republic speak?



(c) Paraphrase: What is Czech Republic's language?



(d) Freebase grounded graph



(b) Paraphrase: What language do people speak in Czech Republic?

Figure 3: Ungrounded graphs for an input question and its paraphrases along with its correct grounded graph. The green squares indicate NL or Freebase entities, the yellow rectangles indicate unary NL predicates or Freebase types, the circles indicate NL or Freebase events, the edge labels indicate binary NL predicates or Freebase relations, and the red diamonds attach to the entity of interest (the answer to the question).

Let q be a question, let p be a paraphrase, let u be an ungrounded graph for p, and let g be a grounded graph formed by grounding the nodes and edges of u to the knowledge base $\mathcal{K}$ (throughout we use Freebase as the knowledge base). Following Reddy et al. (2014), we use beam search to find the highest scoring tuple of paraphrase, ungrounded and grounded graphs $(\hat{p}, \hat{u}, \hat{g})$ under the model $\theta \in \mathbb{R}^n$:

$$(\hat{p}, \hat{u}, \hat{g}) = \arg\max_{(p,u,g)} \theta \cdot \Phi(p, u, g, q, \mathcal{K}) \,,$$

where $\Phi(p, u, g, q, \mathcal{K}) \in \mathbb{R}^n$ denotes the features for the tuple of paraphrase, ungrounded and grounded graphs. The feature function has access to the paraphrase, ungrounded and grounded graphs, the original question, as well as to the content of the knowledge base and the denotation $|g|_{\mathcal{K}}$ (the denotation of a grounded graph is defined as the set of entities or attributes reachable at its TARGET node). See §4.3 for the features employed. The model parameters are estimated with the averaged structured perceptron (Collins, 2002). Given a training question-answer pair $(q, \mathcal{A})$, the update is:

$$\theta^{t+1} \leftarrow \theta^t + \Phi(p^+, u^+, g^+, q, \mathcal{K}) - \Phi(\hat{p}, \hat{u}, \hat{g}, q, \mathcal{K}) \,,$$

where (p^+, u^+, g^+) denotes the tuple of gold paraphrase, gold ungrounded and grounded graphs for q. Since we do not have direct access to the gold paraphrase and graphs, we instead rely on the set of *oracle tuples*, $\mathcal{O}_{\mathcal{K},\mathcal{A}}(q)$, as a proxy:

$$(p^+, u^+, g^+) = \arg\max_{(p,u,g) \in \mathcal{O}_{\mathcal{K},\mathcal{A}}(q)} \theta \cdot \Phi(p, u, g, q, \mathcal{K}) \,,$$

where $\mathcal{O}_{\mathcal{K},\mathcal{A}}(q)$ is defined as the set of tuples (p, u, g) derivable from the question q, whose denotation $|g|_{\mathcal{K}}$ has minimal F_1-loss against the gold answer $\mathcal{A}$. We find the oracle graphs for each question a priori by performing beam-search with a very large beam.

4 Experimental Setup

Below, we give details on the evaluation dataset and baselines used for comparison. We also describe the model features and provide implementation details.

4.1 Evaluation Data and Metric

We evaluate our approach on the WebQuestions dataset (Berant et al., 2013). WebQuestions consists of 5,810 question-answer pairs where questions represents real Google search queries. We use the standard train/test splits, with 3,778 train and 2,032 test questions. For our development experiments we tune the models on held-out data consisting of 30% training questions, while for final testing

we use the complete training data. We use average precision (avg P.), average recall (avg R.) and average F_1 (avg F_1) proposed by Berant et al. (2013) as evaluation metrics.[9]

4.2 Baselines

ORIGINAL We use GRAPHPARSER without paraphrases as our baseline. This gives an idea about the impact of using paraphrases.

MT We compare our paraphrasing models with monolingual machine translation based model for paraphrase generation (Quirk et al., 2004; Wubben et al., 2010). In particular, we use Moses (Koehn et al., 2007) to train a monolingual phrase-based MT system on the Paralex corpus. Finally, we use Moses decoder to generate 10-best distinct paraphrases for the test questions.

4.3 Implementation Details

Entity Resolution For WebQuestions, we use 8 handcrafted part-of-speech patterns (e.g., the pattern `(DT)?(JJ.?|NN.?){0,2}NN.?` matches the noun phrase *the big lebowski*) to identify candidate named entity mention spans. We use the Stanford CoreNLP caseless tagger for part-of-speech tagging (Manning et al., 2014). For each candidate mention span, we retrieve the top 10 entities according to the Freebase API.[10] We then create a lattice in which the nodes correspond to mention-entity pairs, scored by their Freebase API scores, and the edges encode the fact that no joint assignment of entities to mentions can contain overlapping spans. We take the top 10 paths through the lattice as possible entity disambiguations. For each possibility, we generate n-best paraphrases that contains the entity mention spans. In the end, this process creates a total of $10n$ paraphrases. We generate ungrounded graphs for these paraphrases and treat the final entity disambiguation and paraphrase selection as part of the semantic parsing problem.[11]

GRAPHPARSER Features. We use the features from Reddy et al. (2014). These include edge alignments and stem overlaps between ungrounded and grounded graphs, and contextual features such as word and grounded relation pairs. In addition to these features, we add two new real-valued features – the paraphrase classifier's score and the entity disambiguation lattice score.

Beam Search We use beam search to infer the highest scoring graph pair for a question. The search operates over entity-entity edges and entity type nodes of each ungrounded graph. For an entity-entity edge, there are two operations: ground the edge to a Freebase relation, or skip the edge. Similarly, for an entity type node, there are two operations: ground the node to a Freebase type, or skip the node. We use a beam size of 100 in all our experiments.

5 Results and Discussion

In this section, we present results from five different systems for our question-answering experiments: ORIGINAL, MT, NAIVE, PPDB and BILAYERED. First two are baseline systems. Other three systems use paraphrases generated from an L-PCFG grammar. NAIVE uses a word lattice with a single start-to-end path representing the input question itself, PPDB uses a word lattice constructed using the PPDB rules, and BILAYERED uses bi-layered L-PCFG to build word lattices. Note that NAIVE does not require any parallel resource to train, PPDB requires an external paraphrase database, and BILAYERED, like MT, needs a parallel corpus with paraphrase pairs. We tune our classifier features and GRAPHPARSER features on the development data. We use the best setting from tuning for evaluation on the test data.

Results on the Development Set Table 1 shows the results with our best settings on the development data. We found that oracle scores improve significantly with paraphrases. ORIGINAL achieves an oracle score of 65.1 whereas with paraphrases we achieve an F_1 greater than 70 across all the models. This shows that with paraphrases we eliminate substantial mismatch between Freebase and ungrounded graphs. This trend continues for the final prediction with the paraphrasing models performing better than the ORIGINAL.

[9] https://github.com/percyliang/sempre/blob/master/scripts/evaluation.py

[10] http://developers.google.com/freebase/

[11] To generate ungrounded graphs for a paraphrase, we treat each entity mention as a single word.

All our proposed paraphrasing models beat the MT baseline. Even the NAIVE model which does not use any parallel or external resource surpass the MT baseline in the final prediction. Upon error analysis, we found that the MT model produce too similar paraphrases, mostly with only inflectional variations. For the question *What language do people in Czech Republic speak*, the top ten paraphrases produced by MT are mostly formed by replacing words *language* with *languages*, do with *does*, *people* with *person* and *speak* with *speaks*. These paraphrases do not address the structural mismatch problem. In contrast, our grammar based models generate syntactically diverse paraphrases.

Our PPDB model performs best across the paraphrase models (avg F_1 = 47.9). We attribute its success to the high quality paraphrase rules from the external paraphrase database. For the BILAYERD model we found 1,000 latent semantic states is not sufficient for modeling topical differences. Though MT competes with NAIVE and BILAYERED, the performance of NAIVE is highly encouraging since it does not require any parallel corpus. Furthermore, we observe that the MT model has larger search space. The number of oracle graphs – the number of ways in which one can produce the correct Freebase grounding from the ungrounded graphs of the given question and its paraphrases – is higher for MT (77.2) than the grammar-based models (50–60).

Results on the Test Set Table 2 shows our final results on the test data. We get similar results on the test data as we reported on the development data. Again, the PPDB model performs best with an F_1 score of 47.7. The baselines, ORIGINAL and MT, lag with scores of 45.0 and 47.1, respectively. We also present the results of existing literature on this dataset. Among these, Berant and Liang (2014) also uses paraphrasing but unlike ours it is based on a template grammar (containing 8 grammar rules) and requires logical forms beforehand to generate paraphrases. Our PPDB outperforms Berant and Liang's model by 7.8 F_1 points. Yih et al. (2015) and Xu et al. (2016) use neural network models for semantic parsing, in addition to using sophisticated entity resolution (Yang and Chang, 2015) and a very large unsupervised corpus as additional training data. Note that we use GRAPHPARSER as our semantic parsing framework for evaluating our paraphrases extrinsically. We leave plugging our paraphrases to other existing methods and other tasks for future work.

Method	avg oracle F_1	# oracle graphs	avg F_1
ORIGINAL	65.1	11.0	44.7
MT	71.5	77.2	47.0
NAIVE	71.2	53.6	47.5
PPDB	71.8	59.8	47.9
BILAYERED	71.6	55.0	47.1

Table 1: Oracle statistics and results on the WebQuestions development set.

Method	avg P.	avg R.	avg F_1
Berant and Liang '14	40.5	46.6	39.9
Bast and Haussmann '15	49.8	60.4	49.4
Berant and Liang '15	50.4	55.7	49.7
Reddy et al. '16	49.0	61.1	50.3
Yih et al. '15	52.8	60.7	52.5
Xu et al. '16	53.1	65.5	53.3
This paper			
ORIGINAL	53.2	54.2	45.0
MT	48.0	56.9	47.1
NAIVE	48.1	57.7	47.2
PPDB	48.4	58.1	47.7
BILAYERED	47.0	57.6	47.2

Table 2: Results on WebQuestions test dataset.

Error Analysis The upper bound of our paraphrasing methods is in the range of 71.2–71.8. We examine the reason where we lose the rest. For the PPDB model, the majority (78.4%) of the errors are partially correct answers occurring due to incomplete gold answer annotations or partially correct groundings. Note that the partially correct groundings may include incorrect paraphrases. 13.5% are due to mismatch between Freebase and the paraphrases produced, and the rest (8.1%) are due to wrong entity annotations.

6 Conclusion

We described a grammar method to generate paraphrases for questions, and applied it to a question answering system based on semantic parsing. We showed that using paraphrases for a question answering system is a useful way to improve its performance. Our method is rather generic and can be applied to any question answering system.

Acknowledgements

The authors would like to thank Nitin Madnani for his help with the implementation of the paraphrase classifier. We would like to thank our anonymous reviewers for their insightful comments. This research was supported by an EPSRC grant (EP/L02411X/1), the H2020 project SUMMA (under grant agreement 688139), and a Google PhD Fellowship for the second author.

References

Regina Barzilay and Lillian Lee. 2003. Learning to paraphrase: An unsupervised approach using multiple-sequence alignment. In *Proceedings of NAACL-HLT*.

Hannah Bast and Elmar Haussmann. 2015. More accurate question answering on Freebase. In *Proceedings of CIKM*.

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*.

Johan Bos, Stephen Clark, Mark Steedman, James R. Curran, and Julia Hockenmaier. 2004. Wide-coverage semantic representations from a CCG parser. In *Proceedings of COLING*.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of ACL*.

David L. Chen and Raymond J. Mooney. 2011. Learning to interpret natural language navigation instructions from observations. In *Proceedings of AAAI*.

Shay B. Cohen, Karl Stratos, Michael Collins, Dean P. Foster, and Lyle Ungar. 2013. Experiments with spectral learning of latent-variable PCFGs. In *Proceedings of NAACL*.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of EMNLP*.

Pablo Ariel Duboue and Jennifer Chu-Carroll. 2006. Answering the question you wish they had asked: The impact of paraphrasing for question answering. In *Proceedings of NAACL-HLT*.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2013. Paraphrase-driven learning for open question answering. In *Proceedings of ACL*.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The Paraphrase Database. In *Proceedings of NAACL-HLT*.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. 2009. The WEKA data mining software: An update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18.

Martin Kay. 1996. Chart generation. In *Proceedings of ACL*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of ACL*.

Alexander Koller and Kristina Striegnitz. 2002. Generation as dependency parsing. In *Proceedings of ACL*.

Jayant Krishnamurthy and Tom Mitchell. 2012. Weakly supervised training of semantic parsers. In *Proceedings of EMNLP*.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of EMNLP*.

Irene Langkilde and Kevin Knight. 1998. Generation that exploits corpus-based statistical knowledge. In *Proceedings of ACL-COLING*.

Mike Lewis and Mark Steedman. 2014. A* CCG parsing with a supertag-factored model. In *Proceedings of EMNLP*.

Nitin Madnani, Joel Tetreault, and Martin Chodorow. 2012. Re-examining machine translation metrics for paraphrase identification. In *Proceedings of NAACL-HLT*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of ACL*.

Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic CFG with latent annotations. In *Proceedings of ACL*.

Cynthia Matuszek, Nicholas FitzGerald, Luke Zettlemoyer, Liefeng Bo, and Dieter Fox. 2012. A joint model of language and perception for grounded attribute learning. In *Proceedings of ICML*.

Jeff Mitchell and Mark Steedman. 2015. Orthogonality of syntax and semantics within distributional spaces. In *Proceedings of ACL*.

Shashi Narayan and Shay B. Cohen. 2015. Diversity in spectral learning for natural language parsing. In *Proceedings of EMNLP*.

Shashi Narayan and Shay B. Cohen. 2016. Optimizing spectral learning for parsing. In *Proceedings of ACL*.

Shashi Narayan and Claire Gardent. 2012. Structure-driven lexicalist generation. In *Proceedings of COLING*.

Bo Pang, Kevin Knight, and Daniel Marcu. 2003. Syntax-based alignment of multiple translations: Extracting paraphrases and generating new sentences. In *Proceedings of NAACL-HLT*.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of COLING-ACL*.

Detlef Prescher. 2005. Head-driven pcfgs with latent-head statistics. In *Proceedings of IWPT*.

Chris Quirk, Chris Brockett, and William B. Dolan. 2004. Monolingual machine translation for paraphrase generation. In *Proceedings of EMNLP*.

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.

Siva Reddy, Oscar Täckström, Michael Collins, Tom Kwiatkowski, Dipanjan Das, Mark Steedman, and Mirella Lapata. 2016. Transforming Dependency Structures to Logical Forms for Semantic Parsing. *Transactions of the Association for Computational Linguistics*, 4:127–140.

Stefan Riezler, Alexander Vasserman, Ioannis Tsochantaridis, Vibhu Mittal, and Yi Liu. 2007. Statistical machine translation for query expansion in answer retrieval. In *Proceedings of ACL*.

Stuart M. Shieber, Gertjan van Noord, Fernando C. N. Pereira, and Robert C. Moore. 1990. Semantic head-driven generation. *Computational Linguistics*, 16(1):30–42.

Stuart M. Shieber. 1988. A uniform architecture for parsing and generation. In *Proceedings of COLING*.

Yushi Wang, Jonathan Berant, and Percy Liang. 2015. Building a semantic parser overnight. In *Proceedings of ACL*.

Michael White. 2004. Reining in ccg chart realization. In Anja Belz, Roger Evans, and Paul Piwek, editors, *Natural Language Generation*, volume 3123 of *Lecture Notes in Computer Science*, pages 182–191. Springer Berlin Heidelberg.

Yuk Wah Wong and Raymond J. Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proceedings of NAACL*.

Sander Wubben, Antal van den Bosch, and Emiel Krahmer. 2010. Paraphrase generation as monolingual translation: Data and evaluation. In *Proceedings of INLG*.

Kun Xu, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2016. Question Answering on Freebase via Relation Extraction and Textual Evidence. In *Proceedings of ACL*.

Yi Yang and Ming-Wei Chang. 2015. S-MART: Novel tree-based structured learning algorithms applied to tweet entity linking. In *Proceedings of ACL*.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of ACL*.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of UAI*.