Knowledge-Aware Natural Language Understanding

Pradeep Dasigi

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Eduard Hovy (Chair)
Chris Dyer
William Cohen
Luke Zettlemoyer

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Keywords: natural language understanding, knowledge, neural networks, end-to-end models, semantic parsing, question answering

Abstract

Natural Language Understanding (NLU) systems need to encode human generated text (or speech) and reason over it at a deep semantic level. Any NLU system typically involves two main components: The first is an **encoder**, which composes words (or other basic linguistic units) within the input utterances compute encoded representations, which are then used as features in the second component, a **predictor**, to reason over the encoded inputs and produce the desired output. We argue that the utterances themselves do not contain all the information needed for understanding them and identify two kinds of additional knowledge needed to fill the gaps: **background knowledge** and **contextual knowledge**. The goal of this thesis is to build end-to-end NLU systems that encode inputs along with relevant background knowledge, and reason about them in the presence of contextual knowledge.

The first part of the thesis deals with encoding background knowledge. While distributional methods for encoding sentences have been used to represent meaning of words in context, there are other aspects of semantics that are out of their reach. These are related to commonsense or real world information which is part of shared human knowledge but is not explicitly present in the input. We address this limitation by having the encoders also encode background knowledge, and present two approaches for doing so. First, we leverage explicit symbolic knowledge from WordNet to learn ontology-grounded token-level representations of words. We show sentence encodings based on our token representations outperform those based on off-the-shelf word embeddings at predicting prepositional phrase attachment and textual entailment. Second, we look at cases where the required background knowledge cannot be stated symbolically. We model selectional restrictions verbs place on their semantic role fillers to deal with one such case. We use this model to encode events, and show that these representations are better at detecting anomalies in newswire texts than sentence representations produced by LSTMs.

The second part focuses on reasoning with contextual knowledge. We look at Question-Answering (QA) tasks where reasoning can be expressed as sequences of discrete operations, (i.e. semantic parsing problems), and the answer can be obtained by executing the sequence of operations (or logical form) grounded in some context. We do not assume the availability of logical forms, and build weakly supervised semantic parsers. This training setup comes with significant challenges since it involves searching over an exponentially large space of logical forms. To deal with these challenges, we propose 1) using a grammar to constrain the output space of the semantic parser; 2) leveraging a lexical coverage measure to ensure the relevance of produced logical forms to input utterances; and 3) a novel iterative training scheme that alternates between searching for logical forms, and maximizing the likelihood of the retrieved ones, thus effectively transferring the knowledge from simpler logical forms to more complex ones. We build neural encoder-decoder models for semantic parsing that use these techniques, and show state-of-the-art results on two complex QA tasks grounded in structured contexts

Overall, this thesis presents a general framework for NLU with encoding and reasoning as the two core components, and how additional knowledge can augment them. While the tasks presented in this thesis are hard language understanding challenges themselves, they also serve as examples to highlight the role of background and contextual knowledge in encoding and reasoning components. The models built for the tasks provide empirical evidence for the need for additional knowledge, and pointers for building effective knowledge-aware models for other NLU tasks.

Acknowledgments

This work is not mine alone. Several people have contributed to the work that went into this document, either directly or indirectly. Firstly, I am immensely grateful to my advisor, Eduard Hovy, for teaching me to never shy away from, and even get excited about working on important, yet frustratingly hard problems. While I admit that the problems that ended up being solved in this thesis are only a fraction of the ones that I had set out to solve, that I even attempted to tackle them is largely because of him. I am also thankful to Chris Dyer, William Cohen, and Luke Zettlemoyer. I could not have asked for a better committee. I was fortunate to have received valuable insights from them in shaping my thesis, limited only by my ability to utilize them.

I learned a great deal from my talented collaborators and mentors: Waleed Ammar, Matt Gardner, and Jayant Krishnamurthy. Waleed stood by me through several failed attempts at getting the ontology-aware encoders to work, and Jayant and Matt are largely responsible for driving me towards semantic parsing. For all this, I am greatly thankful to them. I spent a large chunk of my time as a PhD student visiting the Allen Institute for Artificial Intelligence, and I am highly grateful to Oren Etzioni and the rest of AI2, both for letting me stay, and for inspiring me to work on impactful problems.

My friends have made my time over the last five years much more enjoyable. Thanks to Sujay, Chandan, Swabha, and Suchin for the fun board game and movie nights, brunches, and hikes, both in Pittsburgh and in Seattle. Thanks to Kartik, Keerthi, and Mrinmaya for the numerous lunches and workout sessions.

I am thankful to my parents for teaching me the value of knowledge, instilling in me the right priorities, and making me a person capable of getting a PhD. I would also like to thank my brother, Praveen, for driving me towards research in Computer Science, and for being a voice of sanity and a source of wisdom. Finally, I would like to thank my dear wife Modhurima, for providing me a solid emotional grounding over the past four years, for giving meaning to life outside of work, and for all the love.

Contents

1 Introduction **1**

1.1 Natural Language Understanding . . . 1

1.1.1 Definition . . . 1

1.1.2 Parts of an NLU system . . . 2

1.2 Knowledge . . . 4

1.2.1 Background Knowledge for Encoding . . . 4

1.2.2 Contextual Knowledge for Reasoning . . . 5

1.3 Knowledge-Aware NLU . . . 7

1.3.1 Better encoding with background knowledge . . . 7

1.3.2 Reasoning with contextual knowledge . . . 8

1.3.3 Evaluating NLU Performance . . . 9

1.4 Thesis Contributions and Outline . . . 9

I Encoding with Background Knowledge **13**

2 Related Work: Learning to Encode **15**

2.1 Representation Learning for Lexical Semantics . . . 15

2.1.1 From distributional to distributed representations . . . 15

2.2 Incorporating Knowledge . . . 16

2.2.1 Multi-prototype word vectors . . . 16

2.2.2 Relying on symbolic knowledge . . . 16

2.2.3 Contextualized word vectors and relying on deeper models . . . 17

2.3 Selectional Preference . . . 17

2.4 WordNet as a source for Selectional Preferences . . . 18

3 Encoding Sentences with Background Knowledge from Ontologies **19**

3.1 Introduction . . . 19

3.2 Ontology-Aware Token Embeddings . . . 20

3.3 WordNet-Grounded Context-Sensitive Token Embeddings . . . 21

3.3.1 WordNet Grounding . . . 21

3.3.2 Context-Sensitive Token Embeddings . . . 22

3.4 PP Attachment . . . 24

3.4.1 Problem Definition . . . 25

3.4.2 Model Definition 25
3.4.3 Experiments 26
3.5 Textual Entailment 31
3.5.1 Data preprocessing 31
3.5.2 Problem Definition 31
3.5.3 Model Definition 32
3.5.4 Experiments 32
3.5.5 Analysis 32
3.6 Conclusion 33
3.6.1 Summary 33
3.6.2 Future Work 34

4 Leveraging Selectional Preferences as Background Knowledge for Encoding Events 35
4.1 Understanding Events 35
4.2 Model for Semantic Anomaly Detection 36
4.2.1 Training 38
4.3 Data 39
4.3.1 Annotation 39
4.3.2 Language Model Separability 40
4.4 Results 40
4.5 Conclusion 43

II Reasoning with Contextual Knowledge 45

5 Related Work: Learning to Reason 47
5.1 Learning to Reason with Latent Variables 47
5.2 Semantic Parsing 48
5.2.1 Formal Definition 48
5.2.2 Variations 49
5.3 Weak Supervision 50
5.3.1 Maximum marginal likelihood training 52
5.3.2 Reinforcement learning methods 53
5.3.3 Structured learning algorithms 53
5.3.4 Bridging objectives 54

6 Constrained Decoding for Semantic Parsing 55
6.1 Need for Grammar Constraints 55
6.2 Need for Entity Linking 57
6.3 Grammar-Constrained Neural Semantic Parser 57
6.3.1 Encoder 58
6.3.2 Decoder 60
6.4 Experiments with WIKITABLEQUESTIONS 62
6.4.1 Context and Logical Form Representation 62

6.4.2 Training with Offline Search 63
6.4.3 Experimental Setup 64
6.4.4 Results 64
6.4.5 Type Constraints 65
6.4.6 Entity Embedding and Linking 66
6.4.7 DPD Training 66
6.4.8 Error Analysis 67
6.5 Conclusion 67

7 Training Semantic Parsers using Iterative Coverage-Guided Search 69
7.1 Coverage-guided search 70
7.2 Iterative search 70
7.3 Task Details 71
7.3.1 Logical form languages 73
7.3.2 Lexicons for coverage 73
7.4 Experiments 76
7.4.1 Model 76
7.4.2 Experimental setup 77
7.4.3 Main results 78
7.4.4 Effect of coverage-guided search 79
7.4.5 Effect of iterative search 80
7.5 Related Work 82
7.6 Conclusion 82

8 Conclusion 85
8.1 Summary 85
8.2 Future work in Knowledge-Aware Encoding 86
8.3 Future work in Knowledge-Aware Reasoning 86

Bibliography 89

List of Figures

1.1 Example from WIKITABLEQUESTIONS, a task requiring reasoning over structured contexts . . . 6
1.2 Example from Cornell Natural Language Visual Reasoning (NLVR), a task providing supervision in the form of binary labels . . . 7

3.1 Example of ontology-aware lexical representation . . . 21
3.2 An example grounding for the word *pool* . . . 22
3.3 Steps for computing the context-sensitive token embedding for the word *'pool'*, as described in Section3.3.2. . . . 23
3.4 Two sentences illustrating the importance of lexicalization in PP attachment decisions. . . . 24
3.5 Effect of training data size on test accuracies of OntoLSTM-PP and LSTM-PP. . . 30
3.6 Visualization of soft attention weights assigned by OntoLSTM to various synsets in Preposition Phrase Attachment . . . 30
3.7 Visualization of soft attention weights assigned by OntoLSTM to various synsets in Textual Entailment . . . 33

4.1 Anomaly Classification Pipeline with Neural Event Model . . . 38
4.2 Comparison of perplexity scores for different labels. Perplexity scores are measured using a Masked LM model with BERT . . . 41
4.3 ROC curves for the LSTM baseline and NEM . . . 42

5.1 Example from WIKITABLEQUESTIONS, a dataset that does not provide logical form supervision . . . 50

6.1 Example context from WIKITABLEQUESTIONS (Pasupat and Liang, 2015) . . . 56
6.2 Overview of our type-constrained neural semantic parsing model . . . 58
6.3 The derivation of a logical form using the type-constrained grammar. The holes in the left column have empty scope, while holes in the right column have scope $\Gamma = \{(x : \texttt{r})\}$ . . . 61

7.1 Example from NLVR dataset showing one sentence associated with two worlds and corresponding binary labels, and translation of the sentence above in our logical form language. Also shown are the actions triggered by the lexicon from the utterance . . . 72
7.2 Complete grammar used by our parser for the NLVR domain . . . 74

7.3 Example from WIKITABLEQUESTIONS dataset showing an utterance, a world, associated denotation, corresponding logical form, and actions triggered by the lexicon. 75

List of Tables

3.1 Prepositional Phrase Attachment results with OntoLSTM in comparison with other strong baselines 27

3.2 Results from RBG dependency parser with features coming from various PP attachment predictors and oracle attachments. 29

3.3 Effect of removing sense priors and context sensitivity (attention) from the model. 30

3.4 Results of OntoLSTM on the SNLI dataset 33

4.1 Annotation Statistics 39

4.2 Classification Performance and Comparison with Baselines 42

4.3 Anomaly Detection Competence 43

6.1 Development and test set accuracy of our semantic parser compared to prior work on WIKITABLEQUESTIONS 65

6.2 Development accuracy of our semantic parser compared to sequence-to-sequence and sequence-to-tree models. 66

6.3 Development accuracy of ablated parser variants trained without parts of the entity linking module. 66

6.4 Development accuracy of our semantic parser when trained with varying numbers of logical forms produced by dynamic programming on denotations. 67

7.1 Coverage Lexicon for NLVR 75

7.2 Coverage Lexicon for WIKITABLEQUESTIONS 76

7.3 Comparison of single model performances of Iterative Search with previously reported single model performances 78

7.4 Comparison of iterative search with static MML, iterative MML, and the previous best result from Liang et al. (2018), all trained on the official split 1 of WIKITABLEQUESTIONS and tested on the official test set. 79

7.5 Comparison of our approach with previously published approaches. We show accuracy and consistency on the development set, and public (Test-P) and hidden (Test-H) test sets. 79

7.6 Effect of coverage guidance on NLVR parsers trained with and without initialization from an MML model. Metrics shown are accuracy and consistency on the public test set. 80

7.7 Effect of iterative search (S) and maximization (M) on NLVR. % cov. is the percentage of training data for which the S step retrieves consistent logical forms. 80

7.8 Iterative search on WikiTableQuestions. M and S refer to Maximization and Search steps. 81
7.9 Complexity of logical forms produced at different iterations, from iteration 0 to iteration 3; each logical form could not be produced at the previous iterations . . 81

Chapter 1

Introduction

1.1 Natural Language Understanding

1.1.1 Definition

Natural Language Understanding (NLU) is the study of building machines that understand human languages. It has been a long-standing problem in the field of Artificial Intelligence. An NLU system should process human generated text (or speech) at a deep semantic level, represent the semantics of the processed inputs, and reason over them to perform a given task. NLU is thus an umbrella term that can be defined in terms of the tasks that fall under it, and these are tasks that require computational systems to perform some amount of nontrivial language comprehension.

NLU is a subfield of Natural Language Processing (NLP) and the distinction between NLP tasks that fall under NLU and those that do not can be made in terms of the depth of the semantics that needs to be modeled for performing those tasks. For example, a well-performing part-of-speech (POS) tagger for English can be built mostly using surface level lexical features (Toutanova et al., 2003a), and POS tagging can thus be considered a non-NLU task. In contrast, identifying whether two given sentences are paraphrases might require modeling syntactic similarity and lexical relations between the words in the sentences (Das and Smith, 2009), thus making paraphrase identification an NLU task. Drawing such a distinction is important for characterizing the class of tasks that fall within the scope of NLU, and subsequently this thesis. However, that is not an easy exercise since one cannot define a strict subset of NLP tasks that can be identified as NLU. That is because there are several tasks that fall somewhere between simpler tasks like POS tagging and more complex ones like paraphrase identification in terms of the complexity of semantics. Moreover, several tasks require varying levels semantic complexity to solve the sub-tasks involved. For example, some sub-tasks within relation extraction, can be solved by simpler pattern matching methods such as Hearst patterns (Hearst, 1992), whereas harder relation extraction sub-tasks might require modeling complex cross-sentence relations (Peng et al., 2017). Similarly, coreference resolution ranges from simple pronominal anaphora resolution, all the way till the complex problems in the Winograd Schema Challenge (Levesque et al., 2012), requiring commonsense reasoning. Another example is of syntactic (dependency) parsing of English, where recognizing the subject of the main verb in a sentence is often an easier sub-problem than identifying the noun or verb in the sentence to which a prepositional phrase attaches, and latter has been studied as a task in itself.

Before the advent of representation-learning techniques, NLU systems were often designed as pipelines that depended on the outputs of more fundamental NLP tasks such as POS tagging and Named Entity Recognition (NER). In contrast, recent NLU systems have moved away from pipeline architectures, and often involve end-to-end learning of intermediate features. While this makes the boundaries of NLU fuzzier, the core intuitions still remain, and the distinction can still be made in terms of complexity of models, with modeling non-NLU tasks generally requiring simpler models (Wang et al., 2015a) than NLU tasks (Chen et al., 2017), given sufficiently large datasets. In this thesis, we rely heavily on representation learning and deep learning techniques. Our focus is to study how we can improve the quality of the semantics extracted from the inputs, and make reasoning more effective. Accordingly, we will choose a subset of tasks that are closer to the NLU end of the spectrum described above, and build computational systems for them.

Now that we have informally characterized the tasks we are interested in, let us identify what goes into building systems for those tasks. Consider Question Answering (QA), which refers to answering questions about a span of text, or other structured knowledge representations like knowledge bases. QA systems are required to encode the meaning of the provided inputs such that the relevant bits of information can be retrieved to answer questions. Recognizing Textual Entailment (RTE), another NLU task, refers to the problem of identifying whether the truth value of some text provided as a hypothesis follows from that of another text provided as a premise, and it is usually done by extracting relevant features from the hypothesis and the premise to see if there is enough overlap between them in the right direction. Similarly, Sentiment Analysis requires automatically categorizing the opinions expressed in the input utterances towards specific targets. This involves extracting appropriate affective states and subjective information, such that a sentiment classifier can be built using that information as features. To perform well at at each of these tasks, the NLU systems should encode the semantics of the input to support the kind of reasoning appropriate for the task. This observation leads us to identifying the parts of a generic NLU system as follows.

1.1.2 Parts of an NLU system

In each of the examples above, it can be noticed that there are two common steps: **encoding** and **reasoning**. The encoder extracts task-relevant features from the input, and the reasoning module performs the appropriate computation on top of the features given by the encoder to produce the desired result. Given this insight, let us attempt to describe a generic NLU system using the following two equations:

$$\mathbf{e} = \texttt{encode}(\mathbf{I}) \tag{1.1}$$

$$\mathbf{o} = \texttt{reason}(\mathbf{e}) \tag{1.2}$$

where $\mathbf{I}$ is the set of textual inputs to the NLU system. For example, $\mathbf{I}$ are single sentences in Sentiment Analysis and pairs of sentences in RTE. $\mathbf{e}$ are intermediate encoded representations of the inputs (which may or may not be task specific), and $\mathbf{o}$ are the final task specific predictions. For example, in Sentiment Analysis or RTE, $\mathbf{o}$ are categorical labels indicating the sentiment or entailment respectively.

Encoding We define encoding in NLU as the process of extracting information (often called features) from the input texts that is required for task-specific reasoning. In older feature-rich methods for NLU, Equation 1.1 used to be a mapping of the inputs to a hand designed feature space, typically containing patterns over word classes based on part-of-speech (Corley and Mihalcea, 2005) or Wordnet synsets (Moldovan and Rus, 2001); shallow linguistic features like dependencies (Bos and Markert, 2005); named entity information (Tatu and Moldovan, 2005); or other features depending on the task. The choice of features was left to the discretion of the ML-practitioners designing the NLU systems. In such systems, the modeling emphasis was more on the reasoning component, and the encoding component did not involve any learning. More recent systems (Bahdanau et al., 2014b; Weston et al., 2014; Hermann et al., 2015; Xiong et al., 2016; Bowman et al., 2016; Yang et al., 2016, among many others) use representation learning or deep learning techniques to also learn the parameters of the `encode` function, and typically this is done jointly with learning the parameters of the `reason` function, to ensure that learned representations are relevant to the task.

Reasoning This step in an NLU system is the one that produces the final prediction. It is called *reasoning* because it is expected to emulate human reasoning, particularly the non-trivial aspects of it. The term is used in AI quite broadly to mean different things. However, according to most of those definitions, reasoning involves processing available information (in its encoded form) to identify patterns or insights that are not trivially obvious. In traditional (or symbolic) AI systems, reasoning typically involved logical inference. That is, given some knowledge expressed as rules, principles of formal logic would be used to infer more rules from them, which may be regarded as the newer insights drawn. In more modern AI systems, reasoning is both non-symbolic and probabilistic. It is non-symbolic because reasoning operates on elements in a continuous space (i.e., the encoded inputs) instead of symbols in a discrete space, and it is probabilistic because the inferred insights do not simply evaluate to "true" or "false", but have degrees of truth, or probabilities associated with them. The complexity of reasoning depends on the task at hand. For example, in tasks like Sentiment Analysis or RTE, where the output is one of multiple classes, assuming that the encoder does a good job at extracting the relevant features from input, reasoning involves applying a classifier for the output classes that operates on the features extracted (Pang et al., 2002). However, in the case of semantic parsing for question answering over a knowledge graph, reasoning requires determining how parts of the question can be linked to nodes and edges in the knowledge graph, and how they can be composed to obtain a complete translation of the question into a semantic space defined by the graph, so that the translation, or a logical form, can be executed against the graph to produce the answer (Zettlemoyer and Collins, 2005a).

We attempted to describe a generic NLU system in Equations 1.1 and 1.2, with the assumption that the information provide by the utterances $\mathbf{I}$ alone is sufficient for encoding their meaning and reasoning over them. However, this is seldom the case, and the generic formulation we have so far is missing another key input. Humans use language as a medium to transfer information. However, since language is aimed at other humans, we often rely on prior knowledge that we share — both about the world, and also about how other humans reason — and communicate efficiently with each other without explicitly making references to this shared knowledge. This makes the job of building NLU systems challenging, since they need to process human utterances more or less in isolation. The utterances themselves seldom contain all the information needed to comprehend them. The missing information is either part of the background knowledge about the world humans inherently possess, or is provided by the surrounding context in which the utterance is spoken or written. How can NLU systems avail this additional knowledge?

1.2 Knowledge

NLU systems need to access or infer the relevant background knowledge and contextual knowledge to fully comprehend human language. In principle, a sophisticated model should be able to infer this knowledge given enough annotated data. Thus, one might argue that NLU system builders should focus on obtaining more data to train fully supervised systems. Our counter-argument to that is based on the practicality of building NLU systems, and is twofold: Firstly, it is often difficult to get sufficient high quality labeled data for most NLU tasks in practice. Any additional knowledge that can augment the information provided by the data could help build a better model for the task at hand. Secondly, if knowledge about some aspects of semantics that need to be modeled can be easily obtained, it is inefficient to not use it, and build a model from scratch hoping that it would learn that knowledge from data. The modeling capacity could instead be used for capturing more nuanced aspects of semantics that the data provide. We will revisit these counter-arguments later in the thesis as we describe concrete tasks.

Given these considerations, it is valuable to study the kinds of knowledge that NLU generally requires, with the goal of determining the following:

1. What knowledge is required to fill in the gaps in the inputs?
2. How can we acquire the missing knowledge?
3. How can the additional knowledge be incorporated into NLU systems?
4. How well does the additional knowledge improve task performance in practice?

Since our ultimate goal is to improve the performance of NLU systems, we take a pragmatic approach for defining the kinds of knowledge required by NLU systems, and further classify them based on our means for acquiring them or ways of incorporating them into NLU system. At a high level, we draw a distinction between two kinds of knowledge: background and contextual, the former is related to encoding input utterances, and the latter is related to reasoning over them. We now describe them in detail.

1.2.1 Background Knowledge for Encoding

Background knowledge refers to the domain-specific information needed by NLU systems to fill in the gaps in human utterances to effectively encode them.

Consider encoding the semantics of the following two sentences:

She ate spaghetti with butter.

She ate spaghetti with chopsticks.

Any effective NLU system that processes these sentences should encode the differences between the functions of the phrases *with butter* and *with chopsticks* in the two sentences, to identify that *butter* is an accompaniment, and *chopsticks* are instruments. To evaluate this capability, we can measure the performance of an NLU system at the task of prepositional-phrase (PP) attachment. Ideally, it should predict that *with butter* attaches to *spaghetti* in the first sentence, and *with chopsticks* attaches to *ate*.

In general domains, the missing knowledge constitutes real world facts and commonsense information, and in specialized domains, it might include more esoteric information. For example, an NLU system processing biomedical texts might need information from gene and protein ontologies (Bodenreider, 2004), and one that is trained to pass an elementary science exam might need access to relevant scientific facts (Clark et al., 2013).

In any case, a system that does not explicitly have access to this knowledge will at best memorize co-occurrence patterns in the input words. Using pre-trained representations to represent input words, as it has been done in several NLU systems recently, avoids the memorization issue to some extent, since the representations provide additional knowledge to the model in the form of co-occurrence statistics obtained from large corpora. However, it is unclear whether distributional information at the word level alone can generally substitute for the kind of knowledge described above. Fortunately, for such knowledge, one can leverage the information a lexical ontology like WordNet (Miller, 1995) gives. For example, WordNet specifies that *butter* is a *diary product*, and *chopsticks* are *tableware*.

However, not all background knowledge can be explicitly stated. A large portion of what we call commonsense cannot be expressed in the form of concrete relations like in the example above. The following example illustrates the point. Consider the events described by the three sentences:

Man recovering after being shot by cops.

Man recovering after being bitten by a dog.

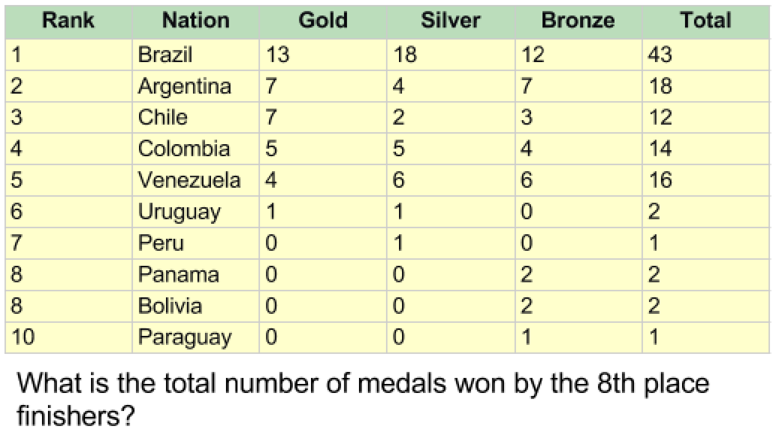
Man recovering after being shot by a dog.

Clearly, one would find the third sentence more unusual compared to the first and the second, the reason being that a *dog* performing a *shooting* action does not agree with our knowledge about the general nature of the world. While this information is not clearly written down anywhere, humans know it implicitly. An NLU system that is designed to differentiate between these sentences should be provided with the information that *dog* in the role of the agent for *shooting* is unlikely, while it can be the patient for *shooting*, and *man* fits equally well as agent and patient for *shooting*. The challenge then is to automatically acquire this kind of background knowledge.

Based on these observations, we define two kinds of background knowledge: **explicit** and **implicit** based on their availability in external resources.

1.2.2 Contextual Knowledge for Reasoning

Meaning is often context-dependent. Contexts provide the circumstances under which the meaning of utterances is finally decoded. Reasoning in those contexts has the pre-requisite of grounding the inputs in them. For example, an automated reading comprehension system that is built to answer questions in some context, after encoding the inputs, is expected to effectively match them against against some representation of the context provided, to extract the information



Rank	Nation	Gold	Silver	Bronze	Total
1	Brazil	13	18	12	43
2	Argentina	7	4	7	18
3	Chile	7	2	3	12
4	Colombia	5	5	4	14
5	Venezuela	4	6	6	16
6	Uruguay	1	1	0	2
7	Peru	0	1	0	1
8	Panama	0	0	2	2
8	Bolivia	0	0	2	2
10	Paraguay	0	0	1	1

What is the total number of medals won by the 8th place finishers?

Figure 1.1: Example from WIKITABLEQUESTIONS, a task requiring reasoning over structured contexts

queried by the question. Hence, even after filling in the gaps due to missing background knowledge, an NLU system may need additional information to link them to (or ground them in) the contexts. We refer to this additional information as **contextual knowledge**. Like background knowledge, this kind of knowledge may not be explicitly stated, but unlike background knowledge, this is context-specific and affects how reasoning can be performed in context.

Depending on the type of contexts over which the given task requires reasoning, different forms of contextual knowledge may be needed. In this thesis, we focus on the case where explicit relations exist among bits of information in the contexts so that they can be represented as knowledge graphs, and reasoning over them can be performed as a series of discrete operations, making it an instance of *Semantic Parsing*. Contextual knowledge in our case is required to link parts of the inputs to nodes or edges in the graph, and to set syntactic and semantic constraints on the logical forms produced by the semantic parser.

Figure 1.1 shows a question taken from the WIKITABLEQUESTIONS dataset (Pasupat and Liang, 2015), and an associated table from a Wikipedia article, which provides the necessary context to answer it. This an example of a QA task where the context is structured as a table. As it can be seen from the example, answering this question requires linking *place* in the question to the *Rank* column, *8th* to the cells with *8* in them, and *total* to a sum operation, and then doing multi-step reasoning over the table: finding the rows where the value under the *Rank* column is *8*; extracting the values under the *Total* column from those rows; and calculating their sum.

Consider another example of a similar problem in Figure 1.2, that requires evaluating whether the given statement is true or false in the given structured context (the image). Reasoning in this case involves determining that *box with multiple items* refers to the presence of muliple objects within one of the three large boxes, *item has different color* the objects defined by their property *color* being different from the rest of the objects in the same box, and *one item* refers to the count of such an object being *1*. Further, reasoning also involves determining the correct order in which these operations need to be performed. All of this requires contextual knowledge that is not explicitly present in the context.

In this thesis, we focus on tasks where clear structure exists in contexts. In cases where the

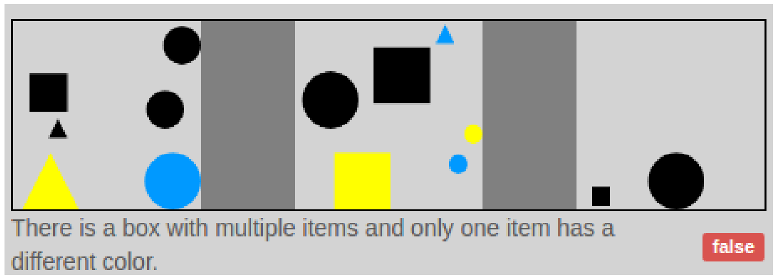

There is a box with multiple items and only one item has a different color.
false

Figure 1.2: Example from Cornell Natural Language Visual Reasoning (NLVR), a task providing supervision in the form of binary labels

contexts are unstructured (Hill et al., 2015; Richardson et al.; Peñas et al., 2013; Breck et al., 2001), reasoning is not as explicit as was shown in the examples above. Accordingly, the kinds of contextual knowledge required for those tasks cannot be explicitly stated. While there are interesting challenges in solving those tasks, and automatically acquiring the contextual knowledge required for them, they are beyond the scope of this thesis.

1.3 Knowledge-Aware NLU

Given that knowledge, both background and contextual, plays an important role in several real-world language understanding tasks, we need to reconsider our definition of the generic NLU pipeline. Without background knowledge, the encoding of inputs will be incomplete, and the NLU systems might not generalize beyond the specific patterns seen in the training data to unseen test cases. Without contextual knowledge, and an effective method to link encoded inputs to contexts, the NLU systems will simply not have sufficient information to produce the desired results. We thus modify our NLU equations as follows.

$$\mathbf{e} = \texttt{encode\_with\_background}(\mathbf{I}, \mathbf{K}_e) \tag{1.3}$$

$$\mathbf{o} = \texttt{reason\_in\_context}(\mathbf{e}, \mathbf{K}_r) \tag{1.4}$$

where $\mathbf{K}_e$ and $\mathbf{K}_r$ represent the knowledge required for encoding and reasoning respectively. They come from different sources and augment the NLU systems in different ways.

1.3.1 Better encoding with background knowledge

$\mathbf{K}_e$ is additional knowledge used to fill in the semantic gaps in the inputs, and thus obtain better encoded representations of them. One common source of such background knowledge in AI is knowledge bases and ontologies, which encode it explicitly. Examples include hypernym trees from WordNet for incorporating sense and generalization information about concepts while composing sentences and subgraph features from Freebase to encode relations between entities seen in the input text. Moldovan and Rus (2001) and Krymolowski and Roth are examples of a feature-rich systems that encoded input sentences in the context of external knowledge. Both systems used WordNet features and other related information about the semantic classes of the words in the input in NLU tasks. While Krymolowski and Roth built a system based on SNoW (Carlson et al., 1999) for predicting Prepositional Phrase Attachment, Moldovan and Rus (2001) built a QA system. In Chapter 3 we describe a representation-learning application of knowledge-aware encoding where we show the advantages of incorporating WordNet information in recurrent neural networks for encoding sentences.

Adding external knowledge inputs to NLU systems is not straightforward. Firstly, while linking text being read to some structured background knowledge in a KB, an automated system usually faces considerable ambiguity. For example, with lexical ontologies like WordNet, we get useful type hierarchies like *parent is-a ancestor is-a person* and *pool is-a body-of-water* and so on, but one has to deal with sense ambiguity: *pool* can also be a game. Another challenge stems from the fact that most KBs represent meaning in a symbolic fasion, whereas most modern NLU systems are based on neural network models, and represent textual inputs in a continuous space. Chapter 3 focuses on learning distributions over the discrete concepts of the KB conditioned on the context, to deal with exceptions in language.

Explicit sources of knowledge are often incomplete, or may not contain the kind of knowledge needed to fill the gaps to obtain representations need for the task at hand. An alternative source of background knowledge is the co-occurrence of slot-fillers, given an appropriate structured representation of the inputs: Assuming we have correctly identified the structure within the inputs, we can leverage large corpora to obtain expected semantics of the fillers of the roles of defined by the structure, and use those expectations to judge the given inputs. This idea is very similar to Selectional Preference, the notion that a verb places semantic restrictions on the subjects and objects it can take (Katz and Fodor, 1963; Wilks, 1975). This notion was successfully used in modeling semantics within syntactic structures for sense disambiguation (Resnik, 1997) and metaphor identification (Shutova et al., 2013), among others. In Chapter 4, we take this idea further to model selectional preferences within predicate argument structures, to learn representations of of the implicit knowledge required to distinguish anomalous newswire events from normal ones.

1.3.2 Reasoning with contextual knowledge

$\mathbf{K}_r$ inputs allow NLU systems to reason using additional contextual knowledge, that which is used for grounding encoded inputs in the context. Within this thesis, we focus on those tasks where reasoning can be viewed as semantic parsing problems. The contexts are often structured in this case. There exist both traditional (Zelle and Mooney, 1996a; Zettlemoyer and Collins, 2005a, 2007, among others) and neural network based methods (Dong and Lapata, 2016a; Andreas et al., 2016a; Liang et al., 2016; Neelakantan et al., 2016) for solving these problems.

Depending on the kind of supervision provided, training semantic parsers might necessitate acquiring contextual knowledge indirectly. If supervision does not include the intermediate logical forms (Berant et al., 2013b; Pasupat and Liang, 2015; Krishnamurthy et al., 2017), training involves some form of a search over the logical form space to find those that correspond to the sequence of operations that result in the correct answer. In that case, relevant contextual knowledge would be the restrictions of the logical form space, and it can be used to constrain the search process, (Xiao et al., 2016; Krishnamurthy et al., 2017), thus making reasoning more efficient.

In the example shown in Figure 1.1, the supervision comes only from the answers. This can be used to drive the search process, since the requirement that the resulting logical form should evaluate to the given answer restricts the search space to a large extent. Figure 1.2 also shows an example, this one from Suhr et al. (2017), where the supervision comes only from the answers. However, since the answers are binary, they do not restrict the search space as much. In fact, given a space of valid logical forms, exactly 50% of them evaluate to the given answer, but only a minute proportion of them are true translations of the given sentence. In this case, when supervison provided is weaker, we have to rely more on contextual knowledge to ensure that the produced logical form involves reasoning relevant to the input sentence.

In this thesis, we first describe a type driven neural semantic parsing framework in Chapter 6, where we define a concrete grammar over the space of logical forms, and use the constraints from the grammar to drive the search process. We also define an entity linking mechanism trained joinly with the parser to allow the parser to reason about previously unseen entities. Then, in Chapter 7, we deal with the issue of weaker supervision from binary labels. The contextual knowledge in this case is a compositional logical form language we define, where the functions are easily mappable to natural language utterances. We exploit the properties of this language to propose a lexicon-guided coverage mechanism that drives the search process towards logical forms that use operations relevant to input utterances. We show that this technique is necessary for domains with binary denotations. Additionally, we present an iterative search method, that alternates between searching for programs that give correct answers, and maximizing the likelihood of retrieved ones, thereby exploiting the compositionality of the target language to produce increasingly complex programs.

1.3.3 Evaluating NLU Performance

The effectiveness of NLU systems is often measured in terms of their performance at the intended task. That is, the entire NLU system is often evaluated based on the outputs of the *Reasoning* step. The inherent assumption behind taking a task-oriented approach towards building NLU systems is that an improvement in the task performance correlates with an improvement in the quality of the semantic representation, and the understanding capability of the computational system. We design our evaluation pipelines in this thesis based on this assumption.

For example, in Chapter 3, where we incorporate background knowledge from WordNet into sentence level encoders, we evaluate the effect of the proposed improvement by measuring the task performance with and without the knowledge incorporated from WordNet. Similarly, in Chapter 4, we argue that encoding selectional preferences results in better event representations, and we evaluate that claim by comparing the performance of a model that encodes selectional preferences with that of another that does not, at the task of predicting newswire event anomalies.

We measure the quality of the encoded representations by running controlled experiments that measure task performance. While interpreting the encoded representations is also an active research topic, it is beyond the scope of this thesis.

1.4 Thesis Contributions and Outline

The following are the primary contributions of this thesis:

0. We present a general framework for NLU, with *encoding* and *reasoning* as the core components.
   We make the following claims, and provide empirical evidence for them:
1. While encoding utterances, incorporating relevant symbolic background knowledge results in better representations of inputs.
2. When relevant background knowledge is not explicitly stated in symbolic form, it can be induced by modeling selectional preferences within appropriate predicate argument structures extracted from the input utterances.
3. For reasoning tasks that can be cast as problems involving search over a space of discrete compositional operations, incorporating knowledge about the syntax and semantics of the operations into the search process results in more effective reasoning modules.
4. Knowledge about the compositionality of the target operator space can be exploited to define a training procedure that can effectively deal with the problem of spuriousness, a significant challenge when training models with weak supervision.

The document is organized as follows:

Chapter 2 describes the previous work done in encoding utterances at various levels of granularity, and the attempts to incorporate background knowledge into the learned representations.

In Chapter 3, we describe a method to incorporate explicit knowledge from WordNet towards obtaining better sentence representations. Our model looks at WordNet synsets and at the hypernym hierarchies of the words being processed, making the encoder aware of their different senses and the corresponding type information. Concretely, we transform a popular variant of recurrent neural-network model, one with Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997), to incorporate ontological information. The ontology-aware LSTM (OntoLSTM) learns to attend to the appropriate sense, and the relevant type (either the most specific concept or a generalization by choosing a hypernym of the word), conditioned on the context and the end-objective. We show that the sentence representationsproduced by OntoLSTM are better than those produced by LSTMs when used with identical prediction components for predicting textual entailment and preposition phrase attachment. We also visualize the attention scores assigned to the hypernyms of words in the input sentences and show that OntoLSTM is indeed learning useful generalizations of words that help the learned representations perform better at the end task.

In Chapter 4 we encode events as predicate argument structures derived from a semantic role labeler. In this chapter, we rely on selectional preferences between verbs and their semantic role fillers as the background knowledge, and show how they are useful in predicting anomalous events. Our event encoder, Neural Event Model (NEM) captures semantics deeper than the surface-level fluency. We describe an annotation effort that resulted in newswire headlines manually labeled with the degree of surprise associated with them and show that NEM outperforms a baseline LSTM model that encodes sentences at anomaly detection.

Chapter 5 provides context for reasoning over grounded language with discrete operations, and various training methods used for dealing with the issue of weak supervision when the reaoning problem is viewed as learning semantic parsers from question-answer pairs.

Chapter 6 deals with reasoning over tables as structured contexts. We describe a type-driven neural semantic parser aimed at answering compositional questions about Wikipedia tables. It is an encoder-decoder model that generates well-typed logical forms that can be executed against graph representations of the tables in context. The parser also includes entity embedding and linking modules that are trained jointly using QA supervision. We show results on the WIKITABLEQUESTIONS dataset.

In Chapter 7, we describe the lexicon-coverage guided search process to deal with lack of sufficient supervision in domains with binary labels like Cornell NLVR. The chapter also includes an iterative search and maximization algorithm where we exploit the compositionality of the logical form language to produce increasingly complex logical forms. We show that this technique is generally applicable to training semantic parsers with weak supervision, and show state of the art results on NLVR and WIKITABLEQUESTIONS.

Finally, in Chapter 8, we describe several potential directions for future research, building on top of the tasks we studied in this thesis, and the kinds of knowledge we exploited. As a concrete example, we discuss how the techniques presented in Chapters 6 and 7 can be extended to unstructured contexts, and the challenges that come with doing so, as a motivation for future work in this area.

Part I

Encoding with Background Knowledge

Chapter 2

Related Work: Learning to Encode

Before we describe our approaches for incorporating background knowledge in encoders for Natural Language Understanding in Chapters 3 and 4, let us first set the context by describing prior work done in this space. Our work is related to various lines of research within the NLP community: incorporating external knowledge into representation learning for lexical sewmantics; dealing with synonymy and homonymy in word representations both in the context of distributed embeddings and more traditional vector spaces; hybrid models of distributional and knowledge based semantics; and selectional preferences and their relation with syntactic and semantic relations.

2.1 Representation Learning for Lexical Semantics

We first briefly review the shift from distributional representations of lexical semantics to learning distributed representations, and then list various attempts at injecting additional knowledge into the representations to make them generalize better to new contexts and domains.

2.1.1 From distributional to distributed representations

Successful early efforts in computational representation of meaning have been driven by the distributional hypothesis (Firth, 1957), which claimed that linguistic items with similar distributions have similar meanings. Those representations were primarily based on co-occurrence statistics of relevant pairs of items, depending on the semantics being represented. One of the earliest methods for producing vector representations of words based on their co-occurrence with documents in a corpus was Latent Semantic Analysis (Deerwester et al., 1990), and used for Information Retrieval. Another similar method was, Hyperspace Analogue to Language (HAL) (Lund and Burgess, 1996), that aggregated the statistics for words within pre-specified window sizes of their co-occurrence as tokens with other tokens (instead of documents) within a large corpus.

A common issue with these approaches is having to deal with sparsity. Since any corpus, no matter how big it is, can provide only a limited number of contexts, generalizing the information gathered from co-occurrence statistics to unseen contexts is a challenge. To deal with this issue, dimensionality reduction techniques like Singular Value Decomposition (SVD) were commonly used.

More recently, word representation techniques have also included a learning component. Consequently, instead of starting from co-occurrence vectors, and optionally reducing their dimensionality, the vectors themselves are *learned* based on some objective from a (proxy) task that requires modeling distributional semantics. For example, Collobert and Weston (2008) trained a multi-task model, while jointly learning word representations, and it has been shown that those representations are generally useful for various downstream applications. Word embeddings were a by-product of the model by Collobert and Weston (2008), but in later work by Mikolov et al. (2013a), it was shown that general purpose word representations can be learned using much simpler, dedicated models. These were the Word2Vec models, where word representations were learned with the objective of predicting their contexts (Skipgram model), or vice versa (Continuous Bag of Words or CBOW model). The contexts are represented as bags of words, thus not explicitly encoding the order of words.

2.2 Incorporating Knowledge

2.2.1 Multi-prototype word vectors

It has been well established that a single vector per word cannot capture its meaning in a variety of contexts. To address this limitation, many efforts were focused on building multi-prototype vector space models of meaning (Reisinger and Mooney, 2010; Huang et al., 2012; Chen et al., 2014; Jauhar et al., 2015; Neelakantan et al., 2015; Arora et al., 2016, etc.). The target of all these approaches is obtaining multi-sense word vector spaces, either by incorporating sense tagged information or other kinds of external context. The number of vectors learned is based on the preset number of senses. Belanger and Kakade (2015) proposed a Gaussian linear dynamical system for estimating token-level word embeddings, and Vilnis and McCallum (2015) proposed mapping each word type to a density instead of a point in a space to account for uncertainty in meaning. These approaches do not make use of lexical ontologies and are not amenable for joint training with a downstream NLP task.

Related to the idea of concept embeddings is the work by Rothe and Schütze (2015), which estimated WordNet synset representations, given pre-trained type-level word embeddings. In contrast, our work focuses on estimating token-level word embeddings as context sensitive distributions of concept embeddings.

2.2.2 Relying on symbolic knowledge

There is a large body of work that tried to improve word embeddings using symbolic knowledge from external resources. Yu and Dredze (2014) extended the CBOW model (Mikolov et al., 2013b) by adding an extra term in the training objective for generating words conditioned on similar words according to a lexicon. Jauhar et al. (2015) extended the Skipgram model (Mikolov et al., 2013b) by representing word senses as latent variables in the generation process, and used a structured prior based on the ontology. Faruqui et al. (2015) used belief propagation to update pre-trained word embeddings on a graph that encodes lexical relationships in the ontology. Similarly, Johansson and Pina (2015) improved word embeddings by representing each sense of the word in a way that reflects the topology of the semantic network they belong to, and then representing the words as convex combinations of their senses. In contrast to previous work that was aimed at improving *type level* word representations, we propose an approach for obtaining *context-sensitive* embeddings at the *token level*, while jointly optimizing the model parameters for the NLP task of interest.

2.2.3 Contextualized word vectors and relying on deeper models

Another way of injecting additional knowledge into vector representations of words is to get it from the internal states of models. This technique was introduced by Peters et al. (2018), with their Embeddings from Language Models (ELMo) idea. The representation learning model encodes the order of words in the context like Collobert and Weston (2008), since they use a language modeling objective. What makes ELMo unique is that instead of representing words statically using some internal state of a language model, they are represented as dynamic functions of internal states from all the layer of a deep language model, with the parameters of the functions being learned jointly from the task objective. The use of a language model makes the word representations contextualized since each word gets a different representation given its context. The fact that the parameters of the function are learned based on the end-task objective essentially finetunes the representations for the task. We will describe a model with similar motivations in Chapter 3, but instead of exploiting the syntactic and semantic patterns learned from a large corpus, by a deep language model like ELMo, the model we present exploits explicit ontological information obtained from WordNet (Miller, 1995). See Section 3.4.3 for an empirical comparison of our model with ELMo.

Following the success of ELMo, the idea of pre-training a language model, and discriminatively fine-tuning the learned representations to other tasks was also used by Radford et al., while replacing the LSTMs used in the neural network architecture for the language modeling task with Transformers (Vaswani et al., 2017), thus benefiting from the efficiency that comes with Transformers. This idea was further extended by Devlin et al. (2018), who used a bidirectional variant of the model, that jointly modeled the left and right contexts.

2.3 Selectional Preference

Selectional preference, a notion introduced by Wilks (1973), refers to the paradigm of modeling semantics of natural language in terms of the restrictions a verb places on its arguments. For example, the knowledge required to identify one of the following sentences as meaningless can be encoded as preferences (or restrictions) of the verb.

Man eats pasta.

Poetry eats computers.

The restrictions include *eat* requiring its subject to be animate, its object to be edible, and so on. Though the notion was originally proposed for a verb and its dependents in the context of dependency grammars, it is equally applicable to other kinds of semantic relations.

The idea is that identifying the right sense of the word can be guided by the preferences of the other words it is related to. Resnik (1997) illustrates this through the example of disambiguating the sense of the word *letter* in the phrase *write a letter*. While *letter* has multiple senses, as an object, the verb *write* "prefers" the sense closer to *reading matter* more than others. At the phrasal level, selectional preferences are useful in encoding complex phenomena like metaphor. Metaphorical usage of words usually involves violating selectional restrictions. For example the phrase *kill an idea* is a metaphor, because *kill* does not usually take abstract concepts as objects. The phrase itself is common and one can easily attribute a metaphorical sense to the verb kill and resolve the violation. Wilks (2007a), Wilks (2007b), Krishnakumaran and Zhu (2007), Shutova et al. (2013), among others discuss the selectional preference view of metaphor identification. Automatic acquisition of selectional preferences from text is a well studied problem. Resnik (1996) proposed the first broad coverage computational model of selectional preferences. The approach is based on using WordNet's hierarchy to generalize across words. It measures selectional association strength of an triple (v, r, c) with verb v, relation r and an argument class c as the Kullback-Leibler divergence between the the distributions $P(c|v, r)$ c and $P(c|r)$ normalized over all classes of arguments that occur with the (v, r) pair. Abe and Li (1996) also use WordNet, and model selectional preferences by minimizing the length tree cut through the noun hierarchy. Ciaramita and Johnson (2000) encode the hierarchy as a Bayesian network and learn selectional preferences using probabilistic graphical methods. Rooth et al. (1999) adopt a distributional approach, and use latent variable models to induce classes of noun-verb pairs, and thus learn their preferences. Methods by Erk (2007); Erk et al. (2010) are also distributional, and have been described earlier in this paper. Séaghdha (2010) proposed the use of Latent Dirichlet Allocation (LDA), to model preferences as topics. Van de Cruys (2009) used non-negative tensor factorization, and modeled subject-verb-object triples as a three-way tensor. The tensor is populated with co-occurrence frequencies, and the selectional preferences are measured from decomposed representations of the three words in the triple. Van de Cruys (2014) trains neural networks to score felicitous triples higher than infelicitous ones.

Erk et al. (2010) also model selectional preferences using vector spaces. They measure the goodness of the fit of a noun with a verb in terms of the similarity between the vector of the noun and some "exemplar" nouns taken by the verb in the same argument role. Baroni and Lenci (2010) also measure selectional preference similarly, but instead of exemplar nouns, they calculate a prototype vector for that role based on the vectors of the most common nouns occurring in that role for the given verb. Lenci (2011) builds on this work and models the phenomenon that the expectations of the verb or its role-fillers change dynamically given other role fillers.

2.4 WordNet as a source for Selectional Preferences

Resnik (1993) showed the applicability of semantic classes and selectional preferences to resolving syntactic ambiguity. Zapirain et al. (2013) applied models of selectional preferences automatically learned from WordNet and distributional information, to the problem of semantic role labeling. Resnik (1993); Brill and Resnik (1994); Agirre (2008) and others have used WordNet information towards improving prepositional phrase attachment predictions.

Chapter 3

Encoding Sentences with Background Knowledge from Ontologies

3.1 Introduction

As described in Chapter 1, the utterances provided as input to an NLU system do not contain all the information needed to fully comprehend them, and an important missing piece is the common-sense information that humans usually leave out for the sake of brevity. In this chapter, we show how we can leverage knowledge-bases to encode this missing information. As opposed to the model described in Chapter 4, the one described here assumes we have the background knowledge explicitly present in a symbolic knowledge base.

The model shown here is one specific kind of knowledge augmented encoder. We restrict ourselves to lexical background knowledge here, specifically of the kind that specifies sense and conceptual information of words. We use WordNet (Miller, 1995) to obtain this information. For example, WordNet contains the information that the word *bugle* as a noun has three senses, one of which is a brass instrument and another is a type of herb. It also shows that *clarion*, *trombone*, *French horn* etc. are also types of brass instruments. Clearly, when the word is used in an utterance, all this information is omitted because the reader is expected to know a lot of it, or atleast be able to infer it from the context. However, this is not true in the case of computational systems that process human language. The goal of this chapter is to develop a model of computational model of language that incorporates information from WordNet to obtain better word representations.

Our model is an ontology-aware recurrent neural network, an encoder that computes token level word representations grounded in an ontology (Wordnet). The kind of background knowledge we deal with here is that of various senses of words, and the generalizations of the concepts that correspond to each sense. We encode that information by using Wordnet to find the collection of semantic concepts manifested by a word type, and represent a word type as a collection of concept embeddings. We show how to integrate the proposed ontology-aware lexical representation with recurrent neural networks (RNNs) to model token sequences. Section 3.4.3 and Section 3.5.4 provide empirical evidence that the WordNet-augmented representations encode lexical background knowledge useful for two tasks that require semantic understanding: predicting prepositional phrase attachment and textual entailment.

3.2 Ontology-Aware Token Embeddings

Types vs. Tokens In accordance with standard terminology, we make the following distinction between types and tokens in this chapter: By word types, we mean the surface form of the word, whereas by tokens we mean the instantiation of the surface form in a context. For example, the same word type *'pool'* occurs as two different tokens in the sentences *"He sat by the pool."* and *"He swam in the pool."*

Type-level word embeddings map a word type (i.e., a surface form) to a dense vector of real numbers such that similar word types have similar embeddings. When pre-trained on a large corpus of unlabeled text, they provide an effective mechanism for generalizing statistical models to words which do not appear in the labeled training data for a downstream task. Most word embedding models define a single vector for each word type. However, a fundamental flaw in this design is their inability to distinguish between different meanings and abstractions of the same word. For example, *pool* has a different sense in the sentence *"He played a game of pool."*, compared to the two examples shown above, but type-level word embeddings typically assign the same representation for all three of them. Similarly, the fact that *'pool'* and *'lake'* are both kinds of water bodies is not explicitly incorporated in most type-level embeddings. Furthermore, it has become a standard practice to tune pre-trained word embeddings as model parameters during training for an NLP task (e.g., Chen and Manning, 2014; Lample et al., 2016), potentially allowing the parameters of a frequent word in the labeled training data to drift away from related but rare words in the embedding space.

In this chapter, we represent a word token in a given context by estimating a context-sensitive probability distribution over relevant concepts in WordNet (Miller, 1995) and use the expected value (i.e., weighted sum) of the concept embeddings as the token representation. Effectively, we map each word type to a grid of concept embeddings (see Fig. 3.1), which are shared across many words. The word representation is computed as a distribution over the concept embeddings from the word's grid. We show how these distributions can be learned conditioned on the context when the representations are plugged into RNNs. Intuitively, commonsense knowledge encoded as WordNet relations could potentially help with language understanding tasks. But mapping tokens to entities in WordNet is a challenge. One needs to at least disambiguate the sense of the token before being able to use the relevant information. Similarly, not all the hypernyms defined by WordNet may be useful for the task at hand as some may be too general to be informative.

We take a task-centric approach towards doing this, and learn the token representations jointly with the task-specific parameters. In addition to providing context-sensitive token embeddings, the proposed method implicitly regularizes the embeddings of related words by forcing related words to share similar concept embeddings. As a result, the representation of a rare word which does not appear in the training data for a downstream task benefits from all the updates to related words which share one or more concept embeddings. While the results presented in this thesis are from a model that relies on WordNet, and we exploit the order of senses given by WordNet, the approach is, in principle applicable to any ontology, with appropriate modifications. Here, we do not assume the inputs are sense tagged.

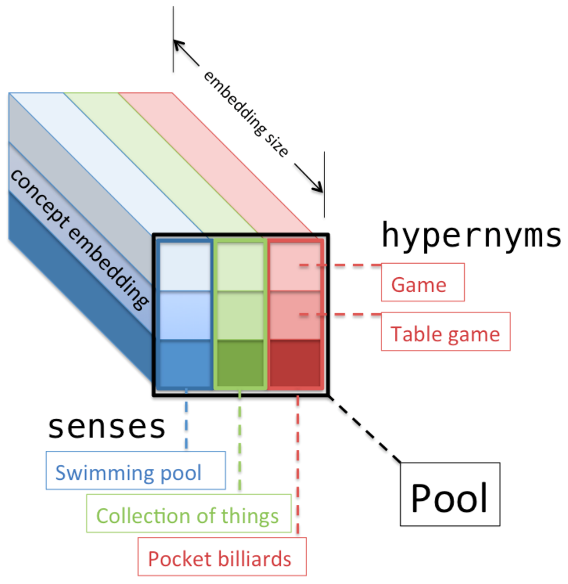


Figure 3.1: Example of ontology-aware lexical representation

We evaluate the proposed embeddings in two applications. The first is predicting prepositional phrase (PP) attachments (see Section 3.4), a challenging problem which emphasizes the selectional preferences between words in the PP and each of the candidate head words. The second application is Textual Entailment (see Section 3.5), a task that benefits from hypernymy features from WordNet, providing generalization information. Our empirical results and detailed analysis (see Section3.4.3 and Section 3.4.3) show that the proposed embeddings effectively use WordNet to improve the accuracy of PP attachment predictions.

3.3 WordNet-Grounded Context-Sensitive Token Embeddings

In this section, we focus on defining our context-sensitive token embeddings. We first describe our grounding of word types using WordNet concepts. Then, we describe our model of context-sensitive token-level embeddings as a weighted sum of WordNet concept embeddings.

3.3.1 WordNet Grounding

We use WordNet to map each word type to a set of synsets, including possible generalizations or abstractions. Among the labeled relations defined in WordNet between different synsets, we focus on the hypernymy relation to help model generalization and selectional preferences between words, which is especially important for predicting PP attachments (Resnik, 1993). To ground a word type, we identify the set of (direct and indirect) hypernyms of the WordNet senses of that word. A simplified grounding of the word 'pool' is illustrated in Figure 3.2. This grounding is key to our model of token embeddings, to be described in the following subsections.

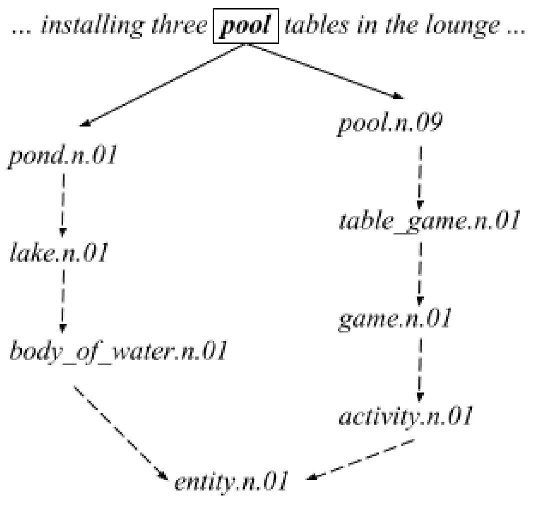


Figure 3.2: An example grounding for the word *pool*

3.3.2 Context-Sensitive Token Embeddings

Our goal is to define a context-sensitive model of token embeddings which can be used as a drop-in replacement for traditional type-level word embeddings.

Notation. Let *Senses*(w) be the list of synsets defined as possible word senses of a given word type w in WordNet, and *Hypernyms*(s) be the list of hypernyms for a synset s [1]. Figure 3.2 shows an example. In the figure, solid arrows represent possible senses and dashed arrows represent hypernym relations. Note that the same set of concepts are used to ground the word *'pool'* regardless of its context. Other WordNet senses for *'pool'* were removed from the figure for simplicity. According to figure,

$$\begin{aligned} \textit{Senses}(\text{pool}) &= [\text{pond.n.01, pool.n.09}], \text{and} \\ \textit{Hypernyms}(\text{pond.n.01}) &= [\text{pond.n.01, lake.n.01,} \\ &\quad \text{body\_of\_water.n.01, entity.n.01}] \end{aligned}$$

Each WordNet synset s is associated with a set of parameters $\mathbf{v}_s \in \mathbb{R}^n$ which represent its embedding. This parameterization is similar to that of Rothe and Schütze (2015).

Embedding model. Given a sequence of tokens $\boldsymbol{t}$ and their corresponding word types $\boldsymbol{w}$, let $\mathbf{u}_i \in \mathbb{R}^n$ be the embedding of the word token t_i at position i. Unlike most embedding models, the token embeddings $\mathbf{u}_i$ are not parameters. Rather, $\mathbf{u}_i$ is computed as the expected value of concept embeddings used to ground the word type w_i corresponding to the token t_i:

$$\mathbf{u}_i = \sum_{s \in \textit{Senses}(w_i)} \sum_{s' \in \textit{Hypernyms}(s)} p(s, s' \mid \boldsymbol{t}, \boldsymbol{w}, i)\, \mathbf{v}_{\mathbf{s}'} \tag{3.1}$$

[1] For notational convenience, we assume that $s \in \textit{Hypernyms}(s)$.

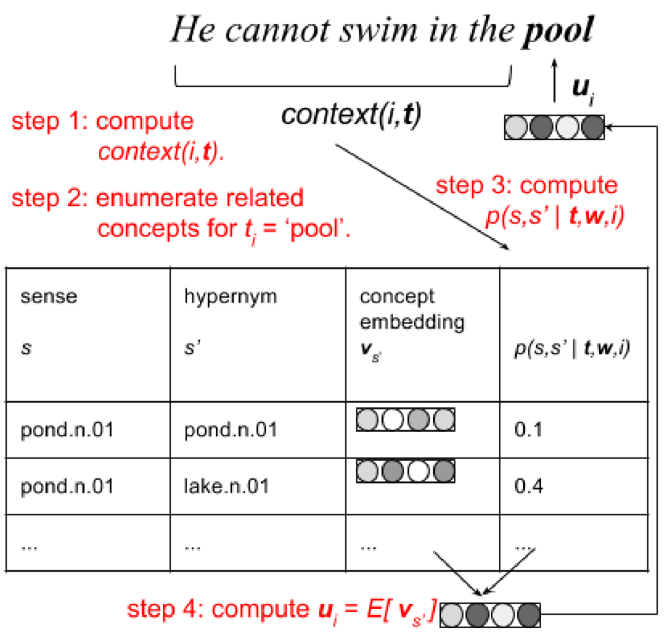

He cannot swim in the pool
u_i
step 1: compute context(i,t).
context(i,t)
step 2: enumerate related concepts for t_i = 'pool'.
step 3: compute p(s,s' | t,w,i)
sense s
hypernym s'
concept embedding $v_{s'}$
p(s,s' | t,w,i)
pond.n.01
pond.n.01
0.1
pond.n.01
lake.n.01
0.4
...
step 4: compute u_i = E[$v_{s'}$]

Figure 3.3: Steps for computing the context-sensitive token embedding for the word *'pool'*, as described in Section3.3.2.

such that

$$\sum_{s \in \mathit{Senses}(w_i)} \sum_{s' \in \mathit{Hypernyms}(s)} p(s, s' \mid \boldsymbol{t}, \boldsymbol{w}, i) = 1$$

The distribution that governs the expectation over synset embeddings factorizes into two components:

$$p(s, s' \mid \boldsymbol{t}, \boldsymbol{w}, i) \propto \lambda_{w_i} \exp^{-\lambda_{w_i}\ \mathit{rank}(s, w_i)} \times \mathit{MLP}([\mathbf{v}_{s'}; \mathit{context}(i, \boldsymbol{t})]) \tag{3.2}$$

The first component, $\lambda_{w_i} \exp^{-\lambda_{w_i}\ \mathit{rank}(s,w_i)}$, is a sense prior which reflects the prominence of each word sense for a given word type. Here, we exploit[2] the fact that WordNet senses are ordered in descending order of their frequencies, obtained from sense tagged corpora, and parameterize the sense prior like an exponential distribution. $rank(s, w_i)$ denotes the rank of sense s for the word type w_i, thus $rank(s, w_i) = 0$ corresponds to s being the first sense of w_i. The scalar parameter (λ_{w_i}) controls the decay of the probability mass, which is learned along with the other parameters in the model. Note that sense priors are defined for each word type (w_i), and are shared across all tokens which have the same word type.

$\mathit{MLP}([\mathbf{v}_{s'}; \mathit{context}(i, \boldsymbol{t})])$, the second component, is what makes the token representations context-sensitive. It scores each concept in the WordNet grounding of w_i by feeding the concatenation of the concept embedding and a dense vector that summarizes the textual context into a

[2]Note that for ontologies where such information is not available, our method is still applicable but without this component. We show the effect of using a uniform sense prior in Section 3.4.3.

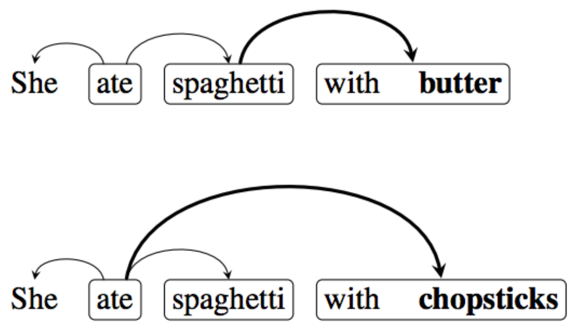


Figure 3.4: Two sentences illustrating the importance of lexicalization in PP attachment decisions.

multilayer perceptron (*MLP*) with two $\tanh$ layers followed by a $softmax$ layer. This component is inspired by the soft attention often used in neural machine translation (Bahdanau et al., 2014b).[3] The definition of the *context* function is dependent on the encoder used to encode the context. We describe a specific instantiations of this function in Section 3.4 and Section 3.5.

To summarize, Figure 3.3 illustrates how to compute the embedding of a word token $t_i =$ *'pool'* in a given context:

1. compute a summary of the context *context*$(i, \boldsymbol{t})$,
2. enumerate related concepts for t_i,
3. compute $p(s, s' \mid \boldsymbol{t}, \boldsymbol{w}, i)$ for each pair (s, s'), and
4. compute $\mathbf{u}_i = \mathbb{E}[\mathbf{v}_{s'}]$.

In the following section, we describe our model for predicting PP attachments, including our definition for *context* for this problem.

3.4 PP Attachment

Disambiguating PP attachments is an important and challenging NLP problem. Since modeling hypernymy and selectional preferences is critical for successful prediction of PP attachments (Resnik, 1993), it is a good fit for evaluating our WordNet-grounded context-sensitive embeddings.

Figure 3.4, reproduced from Belinkov et al. (2014), illustrates an example of the PP attachment prediction problem. In the top sentence, the PP *'with butter'* attaches to the noun *'spaghetti'*. In the bottom sentence, the PP *'with chopsticks'* attaches to the verb *'ate'*. The accuracy of a competitive English dependency parser at predicting the head word of an ambiguous prepositional phrase is 88.5%, significantly lower than the overall unlabeled attachment accuracy

[3]Although soft attention mechanism is typically used to explicitly represent the importance of each item in a sequence, it can also be applied to non-sequential items.

of the same parser (94.2%).[4]

This section formally defines the problem of PP attachment disambiguation, describes our baseline model, then shows how to integrate the token-level embeddings in the model.

3.4.1 Problem Definition

We follow Belinkov et al. (2014)'s definition of the PP attachment problem. Given a preposition p and its direct dependent d in the prepositional phrase (PP), our goal is to predict the correct head word for the PP among an ordered list of candidate head words $\boldsymbol{h}$. Each example in the train, validation, and test sets consists of an input tuple $\langle \boldsymbol{h}, p, d \rangle$ and an output index k to identify the correct head among the candidates in $\boldsymbol{h}$. Note that the order of words that form each $\langle \boldsymbol{h}, p, d \rangle$ is the same as that in the corresponding original sentence.

3.4.2 Model Definition

Both our proposed and baseline models for PP attachment use bidirectional RNN with LSTM cells (bi-LSTM) to encode the sequence $\boldsymbol{t} = \langle h_1, \ldots, h_K, p, d \rangle$.

We score each candidate head by feeding the concatenation of the output bi-LSTM vectors for the head h_k, the preposition p and the direct dependent d through an MLP, with a fully connected $\tanh$ layer to obtain a non-linear projection of the concatenation, followed by a fully-connected *softmax* layer:

$$\begin{aligned} p(h_k \text{is head}) = \textit{MLP}_{\textit{attach}}([&\textit{lstm\_out}(h_k); \\ &\textit{lstm\_out}(p); \\ &\textit{lstm\_out}(d)]) \end{aligned} \tag{3.3}$$

To train the model, we use cross-entropy loss at the output layer for each candidate head in the training set. At test time, we predict the candidate head with the highest probability according to the model in Eq. 3.3, i.e.,

$$\hat{k} = \arg\max_k p(h_k \text{is head} = 1). \tag{3.4}$$

This model is inspired by the Head-Prep-Child-Ternary model of Belinkov et al. (2014). The main difference is that we replace the input features for each token with the output bi-RNN vectors.

We now describe the difference between the proposed and the baseline models. Generally, let *lstm_in*(t_i) and *lstm_out*(t_i) represent the input and output vectors of the bi-LSTM for each token $t_i \in \boldsymbol{t}$ in the sequence. The outputs at each timestep are obtained by concatenating those of the forward and backward LSTMs.

[4]See Table 3.2 for detailed results.

Baseline model. In the baseline model, we use type-level word embeddings to represent the input vector *lstm_in*(t_i) for a token t_i in the sequence. The word embedding parameters are initialized with pre-trained vectors, then tuned along with the parameters of the bi-LSTM and *MLP*$_{attach}$. We call this model **LSTM-PP**.

Proposed model. In the proposed model, we use token level word embedding as described in Section 3.3 as the input to the bi-LSTM, i.e., *lstm_in*$(t_i) = \mathbf{u}_i$. The context used for the attention component is simply the hidden state from the previous timestep. However, since we use a bi-LSTM, the model essentially has two RNNs, and accordingly we have two context vectors, and associated attentions. That is, *context*$_f$*(i, **t**)* $=$ *lstm_in*(t_{i-1}) for the forward RNN and *context*$_b$*(i, **t**)* $=$ *lstm_in*(t_{i+1}) for the backward RNN. Consequently, each token gets two representations, one from each RNN. The synset embedding parameters are initialized with pre-trained vectors and tuned along with the sense decay (λ_w) and MLP parameters from Eq. 3.2, the parameters of the bi-LSTM and those of *MLP*$_{attach}$. We call this model **OntoLSTM-PP**.

3.4.3 Experiments

Dataset and evaluation. We used the English PP attachment dataset created and made available by Belinkov et al. (2014). The training and test splits contain 33,359 and 1951 labeled examples respectively. As explained in Section3.4.1, the input for each example is

1. an ordered list of candidate head words
2. the preposition, and
3. the direct dependent of the preposition.

The head words are either nouns or verbs and the dependent is always a noun. All examples in this dataset have at least two candidate head words. As discussed in Belinkov et al. (2014), this dataset is a more realistic PP attachment task than the RRR dataset (Ratnaparkhi et al., 1994). The RRR dataset is a binary classification task with exactly two head word candidates in all examples. The context for each example in the RRR dataset is also limited which defeats the purpose of our context-sensitive embeddings.

Model specifications and hyperparameters. For efficient implementation, we use mini-batch updates with the same number of senses and hypernyms for all examples, padding zeros and truncating senses and hypernyms as needed. For each word type, we use a maximum of S senses and H indirect hypernyms from WordNet. In our initial experiments on a held-out development set (10% of the training data), we found that values greater than $S = 3$ and $H = 5$ did not improve performance. We also used the development set to tune the number of layers in *MLP*$_{attach}$ separately for the *OntoLSTM-PP* and *LSTM-PP*, and the number of layers in the attention MLP in *OntoLSTM-PP*. When a synset has multiple hypernym paths, we use the shortest one. Finally, words types which do not appear in WordNet are assumed to have one unique sense per word type with no hypernyms. Since the POS tag for each word is included in the dataset, we exclude WordNet synsets which are incompatible with the POS tag. The synset embedding parameters are initialized using the synset vectors obtained by running AutoExtend (Rothe and Schütze, 2015) on 100-dimensional GloVe (Pennington et al., 2014) vectors for WordNet 3.1. We refer

System	**Initialization**	**Resources**	**Parameters (millions)**	**Test Acc.**
HPCD (full)	Syntactic-SG	WordNet, VerbNet	-	88.7
LSTM-PP	GloVe	-	1.3	84.3
LSTM-PP	GloVe-retro	WordNet	1.3	84.8
LSTM-PP w/ ELMo	GloVe + ELMo	LM trained on 1B words	95	89.3 ± 0.3
OntoLSTM-PP	GloVe-extended	WordNet	2	89.3 ± 0.5

Table 3.1: Prepositional Phrase Attachment results with OntoLSTM in comparison with other strong baselines

to this embedding as *GloVe-extended*. Representation for the OOV word types in LSTM-PP and OOV synset types in *OntoLSTM-PP* were randomly drawn from a uniform 100-d distribution. Initial sense prior parameters (λ_w) were also drawn from a uniform 1-d distribution.

Baselines. In our experiments, we compare our proposed model, *OntoLSTM-PP* with four baselines:

1. *LSTM-PP* initialized with GloVe embedding
2. *LSTM-PP* initialized with GloVe vectors retrofitted to WordNet using the approach of Faruqui et al. (2015) (henceforth referred to as *GloVe-retro*). Note that while the representations from Faruqui et al. (2015) also use WordNet information like we do, their representations are still static (i.e. not contextualized), and they do not embed WordNet synsets, but instead use the neighborhood information from WordNet to improve word-type embeddings.
3. *LSTM-PP* using ELMo (Peters et al., 2018) embeddings, with the token embeddings initialized with GloVe. We use the original ELMo model[5] with the language model trained on the 1 billion word benchmark (Chelba et al., 2013).
4. the best performing standalone PP attachment system from Belinkov et al. (2014), referred to as *HPCD (full)* in the paper. *HPCD (full)* is a neural network model that learns to compose the vector representations of each of the candidate heads with those of the preposition and the dependent, and predict attachments. The input representations are enriched using syntactic context information, POS, WordNet and VerbNet (Kipper et al., 2008) information and the distance of the head word from the PP is explicitly encoded in composition architecture. In contrast, we do not use syntactic context, VerbNet and distance information, and do not explicitly encode POS information.

PP Attachment Results

Table 3.1 shows that our proposed token level embedding scheme *OntoLSTM-PP* outperforms the better variant of our baseline *LSTM-PP* (with GloVe-retro intialization) by an absolute accuracy difference of 4.9%, or a relative error reduction of 32%. *OntoLSTM-PP* also outperforms *HPCD*

[5] `https://allennlp.org/elmo`

(full), the previous best result on this dataset. *HPCD (full)* is from the original paper, and it uses syntactic SkipGram. *GloVe-retro* is GloVe vectors retrofitted (Faruqui et al., 2015) to WordNet 3.1, and *GloVe-extended* refers to the synset embeddings obtained by running AutoExtend (Rothe and Schütze, 2015) on GloVe.

Initializing the word embeddings with *GloVe-retro* (which uses WordNet as described in Faruqui et al. (2015)) instead of GloVe amounts to a small improvement, compared to the improvements obtained using *OntoLSTM-PP*. This result illustrates that our approach of dynamically choosing a context sensitive distribution over synsets is a more effective way of making use of WordNet.

Using ELMo embeddings in the *LSTM-PP* model closes the gap between the LSTM models and *OntoLSTM-PP*. Table 3.1 shows mean and standard deviation of the test accuracy over 4 models trained with different random seeds. It can be seen that the accuracies of the OntoLSTM-PP model and the LSTM-PP model with ELMo are not significantly different. However, it has to be noted that the model with ELMo has more than 40 times more parameters than the OntoLSTM model.

Effect on dependency parsing. Following Belinkov et al. (2014), we used RBG parser (Lei et al., 2014), and modified it by adding a binary feature indicating the PP attachment predictions from our model.

We compare four ways to compute the additional binary features:

1. the predictions of the best standalone system *HPCD (full)* in Belinkov et al. (2014)
2. the predictions of our baseline model *LSTM-PP*
3. the predictions of our improved model *OntoLSTM-PP*
4. the gold labels *Oracle PP*.

Table 3.2 shows the effect of using the PP attachment predictions as features within a dependency parser. We note there is a relatively small difference in unlabeled attachment accuracy for all dependencies (not only PP attachments), even when gold PP attachments are used as additional features to the parser. However, when gold PP attachment are used, we note a large potential improvement of 10.46 points in PP attachment accuracies (between the PPA accuracy for *RBG* and *RBG + Oracle PP*), which confirms that adding PP predictions as features is an effective approach. Our proposed model *RBG + OntoLSTM-PP* recovers 15% of this potential improvement, while *RBG + HPCD (full)* recovers 10%, which illustrates that PP attachment remains a difficult problem with plenty of room for improvements even when using a dedicated model to predict PP attachments and using its predictions in a dependency parser.

We also note that, although we use the same predictions of the *HPCD (full)* model in Belinkov et al. (2014)[6], we report different results than Belinkov et al. (2014). For example, the unlabeled attachment score (UAS) of the baselines *RBG* and *RBG + HPCD (full)* are 94.17 and 94.19, respectively, in Table 3.2, compared to 93.96 and 94.05, respectively, in Belinkov et al. (2014). This is due to the use of different versions of the RBG parser.[7]

[6]The authors kindly provided their predictions for 1942 test examples (out of 1951 examples in the full test set). In Table 3.2, we use the same subset of 1942 test examples.

[7]We use the latest commit (SHA: e07f74) on the GitHub repository of the RGB parser.

System	Full UAS	PPA Acc.
RBG	94.17	88.51
RBG + HPCD (full)	94.19	89.59
RBG + LSTM-PP	94.14	86.35
RBG + OntoLSTM-PP	94.30	90.11
RBG + Oracle PP	94.60	98.97

Table 3.2: Results from RBG dependency parser with features coming from various PP attachment predictors and oracle attachments.

Analysis

We now analyze different aspects of our model in order to develop a better understanding of its behavior.

Effect of context sensitivity and sense priors. We now show some results that indicate the relative strengths of two components of our context-sensitive token embedding model. The second row in Table 3.3 shows the test accuracy of a system trained without sense priors (that is, making $p(s|w_i)$ from Eq. 3.1 a uniform distribution), and the third row shows the effect of making the token representations context-insensitive by giving a similar attention score to all related concepts, essentially making them type level representations, but still grounded in WordNet. As it can be seen, removing context sensitivity has an adverse effect on the results. This illustrates the importance of the sense priors and the attention mechanism.

It is interesting that, even without sense priors and attention, the results with WordNet grounding is still higher than that of the two LSTM-PP systems in Table 3.1. This result illustrates the regularization behavior of sharing concept embeddings across multiple words, which is especially important for rare words.

Effect of training data size. Since *OntoLSTM-PP* uses external information, the gap between the model and *LSTM-PP* is expected to be more pronounced when the training data sizes are smaller. To test this hypothesis, we trained the two models with different amounts of training data and measured their accuracies on the test set. The plot is shown in Figure 3.5. As expected, the gap tends to be larger at smaller data sizes. Surprisingly, even with 2000 sentences in the training data set, *OntoLSTM-PP* outperforms *LSTM-PP* trained with the full data set. When both the models are trained with the full dataset, *LSTM-PP* reaches a training accuracy of 95.3%, whereas *OntoLSTM-PP* reaches 93.5%. The fact that *LSTM-PP* is overfitting the training data more, indicates the regularization capability of *OntoLSTM-PP*.

Qualitative analysis. To better understand the effect of WordNet grounding, we took a sample of 100 sentences from the test set whose PP attachments were correctly predicted by OntoLSTM-PP but not by LSTM-PP. A common pattern observed was that those sentences contained words not seen frequently in the training data. Figure 3.6 shows two such cases where OntoLSTM-PP predicts the head correctly and LSTM-PP does not, along with weights by OntoLSTM-PP to

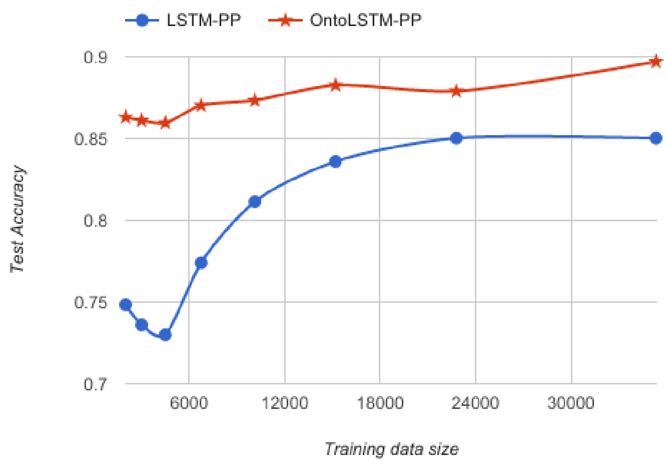


Figure 3.5: Effect of training data size on test accuracies of OntoLSTM-PP and LSTM-PP.

Model	**PPA Acc.**
full	89.7
w/o sense priors	88.4
w/o attention	87.5

Table 3.3: Effect of removing sense priors and context sensitivity (attention) from the model.

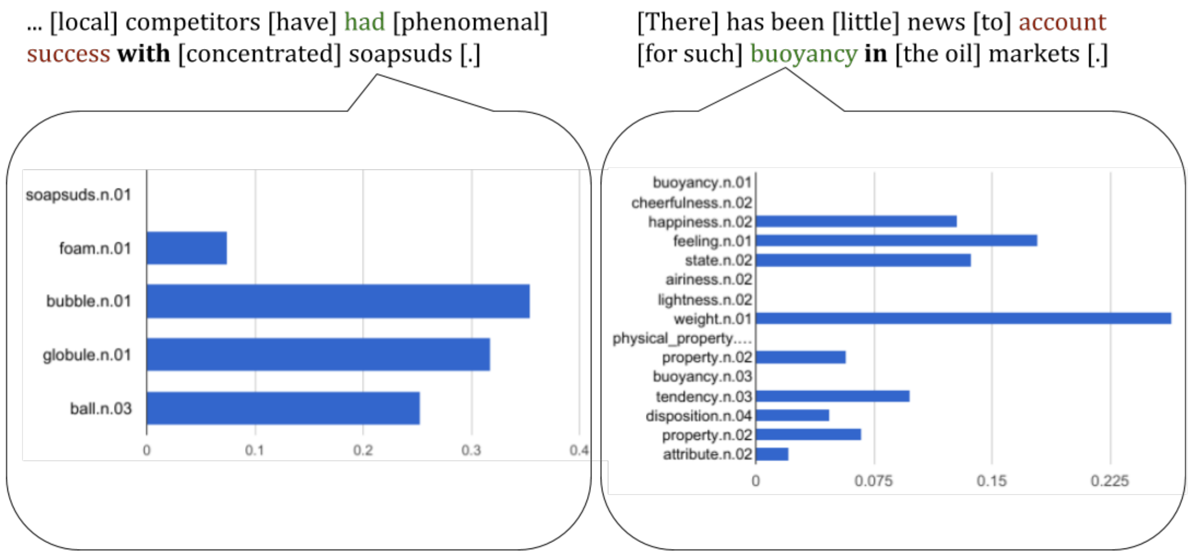


Figure 3.6: Visualization of soft attention weights assigned by OntoLSTM to various synsets in Preposition Phrase Attachment

synsets that contribute to token representations of infrequent word types. The prepositions are shown in bold, LSTM-PP's predictions in red and OntoLSTM-PP's predictions in green. Words that are not candidate heads or dependents are shown in brackets. In both cases, the weights assigned by OntoLSTM-PP to infrequent words are also shown. The word types *soapsuds* and *buoyancy* do not occur in the training data, but OntoLSTM-PP was able to leverage the parameters learned for the synsets that contributed to their token representations. Another important observation is that the word type *buoyancy* has four senses in WordNet (we consider the first three), none of which is the metaphorical sense that is applicable to *markets* as shown in the example here. Selecting a combination of relevant hypernyms from various senses may have helped OntoLSTM-PP make the right prediction. This shows the value of using hypernymy information from WordNet. Moreover, this indicates the strength of the hybrid nature of the model, that lets it augment ontological information with distributional information.

Parameter space. We note that the vocabulary sizes in OntoLSTM-PP and LSTM-PP are comparable as the synset types are shared across word types. In our experiments with the full PP attachment dataset, we learned embeddings for 18k synset types with OntoLSTM-PP and 11k word types with LSTM-PP. Since the biggest contribution to the parameter space comes from the embedding layer, the complexities of both the models are comparable.

3.5 Textual Entailment

We evaluate the capability of OntoLSTM to learn useful generalizations of concepts by testing it on the task of textual entailment. We use the Stanford Natural Language Inference (SNLI) dataset (Bowman et al., 2015).

3.5.1 Data preprocessing

Since the dataset does not contain POS tag information unlike the PP attachment dataset, we run Stanford's POS tagger (Toutanova et al., 2003b) on the dataset. To construct the tensor representation, we map the POS-tagged word to the first two synsets in WordNet (i.e. $S = 2$), and extract the most direct five hypernyms (i.e., $H = 5$). When a synset has multiple hypernym paths, we use the shortest one. In preliminary experiments, we found that using more word senses or more hypernyms per word sense does not improve the performance. Like with the PPA dataset, words which do not appear in WordNet are assumed to have one unique synset per word type with no hypernyms.

3.5.2 Problem Definition

Given a pair of sentences (premise and hypothesis), the model predicts one of three possible relationships between the two sentences: *entailment*, *contradiction* or *neutral*. We use the standard train, development and test splits of the SNLI corpus, which consists of 549K, 10K and 10K labeled sentence pairs, respectively. For example, the following sentence pair is labeled *entailment*:

- **Premise** *Children and parents are splashing water at the pool.*
- **Hypothesis** *Families are playing outside.*

3.5.3 Model Definition

Following Bowman et al. (2016), our model for textual entailment uses two LSTMs with tied parameters to read the premise and the hypothesis of each example, then compute the vector h which summarizes the sentence pair as the following concatenation (originally proposed by Mou et al. (2016)):

$$h = \left[h_{\text{pre}}; h_{\text{hyp}}; h_{\text{pre}} - h_{\text{hyp}}; h_{\text{pre}} * h_{\text{hyp}}\right]$$

where h_{pre} is the output hidden layer at the end of the premise token sequence, h_{hyp} is the output hidden layer at the end of the hypothesis token sequence. The summary h then feeds into two fully connected ReLU layers of size 1024, followed by a softmax layer of size 3 to predict the label. The word embeddings and concept embeddings are of size 300, and the hidden layers 150.

The proposed model uses the WordNet grounded token embeddings, and we call it **OntoLSTM-TE**. We compare it with two baselines: The first one is a simpler variant of *OntoLSTM-TE*, and does not use attention over synsets. We call it **OntoLSTM-TE$_{\text{uni}}$**. The second uses type level word representations, and we call it **LSTM-TE**.

The models are trained to maximize log-likelihood of correct labels in the training data, using ADAM (Kingma and Ba, 2014) with early stopping (up to twenty epochs).

3.5.4 Experiments

Table 3.4 shows the classification results. We also report previously published results of a similar neural architecture as an extra baseline: the 300-dimensional LSTM RNN encoder model in Bowman et al. (2016).[8] They use GloVe as pretrained word embeddings while we jointly learn word/concept embeddings in the same model. Bowman et al. (2016)'s LSTM outperforms *LSTM-TE* model by 0.9 absolute points on the test set. This may be due to the difference in input representations. Since we learn the synset representations in *OntoLSTM-TE*, comparison with our LSTM implementations is more sensible. The *OntoLSTM-TE* outperforms *LSTM-TE* by 1.4 absolute points on the test set, illustrating the utility of the ontology-aware word representations in a controlled experiment.

3.5.5 Analysis

Generalization attention: Fig. 3.7 shows one example from the test set of the SNLI dataset where hypernymy information is useful. The *LSTM-TE* model shares no parameters between the lexical representations of *book* and *manuals*, and fails to predict the correct relationship (i.e., entailment). However, *OntoLSTM-TE* leverages the fact that *book* and *manuals* share the common hypernyms *book.n.01* and *publication.n.01*, and assigns relatively high attention probabilities to

[8]We note this is not the state of the art on this dataset. This is the simplest sentence encoding based model for this task.

Model	**GloVe**	**Train**	**Test**
LSTM-TE	No	90.5	79.7
OntoLSTM-TE$_{\text{uni}}$	No	90.6	81.0
OntoLSTM-TE	No	90.5	81.1
LSTM (Bowman et al.)	Yes	83.9	80.6

Table 3.4: Results of OntoLSTM on the SNLI dataset

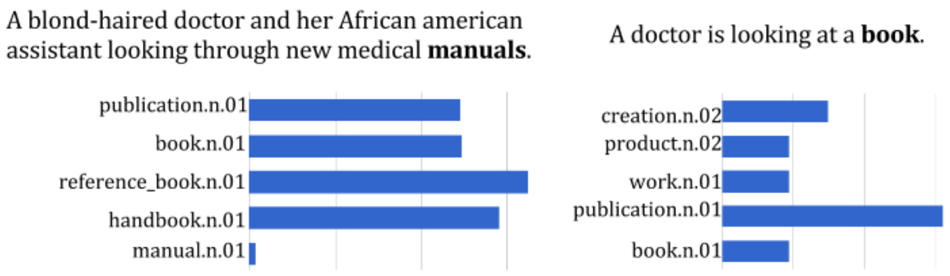


Figure 3.7: Visualization of soft attention weights assigned by OntoLSTM to various synsets in Textual Entailment

these hypernyms, resulting in a similar lexical representation of the two words, and a correct prediction of this example.

The following is an example that *LSTM-TE* gets right and *OntoLSTM-TE* does not:

- Premise: *Women of India performing with blue streamers, in beautiful blue costumes.*
- Hypothesis: *Indian women perform together in gorgeous costumes*

Indian in the second sentence is an adjective, and *India* in the first is an noun. By design, WordNet does not link words of different parts of speech. Moreover, the adjective hierarchies in WordNet are very shallow and *gorgeous* and *beautiful* belong to two different synsets, both without any hypernyms. Hence *OntoLSTM-TE* could not use any generalization information in this problem.

Parameter space: As mentioned earlier in the case of PP attachment problem, the model size in *LSTM-TE* and both variants of *OntoLSTM-TE* are comparable (11.1M and 12.7M parameters, respectively).

3.6 Conclusion

3.6.1 Summary

In this chapter, we showed a procedure for grounding of lexical items which acknowledges the semantic ambiguity of word types using WordNet and a way to learn a context-sensitive distribution over their representations. We also showed how to integrate the proposed representation with recurrent neural networks for modeling sentences, showing that the proposed WordNet-grounded context-sensitive token embeddings outperforms standard type-level embeddings for predicting PP attachments and textual entailment. For PP attachments, we also showed that our model performs comparable to a baseline model with ELMo embeddings, even though our model has only a fraction of the number of parameters as the ELMo model. This result suggests that symbolic ontologies like WordNet may be able to provide knowledge that lets us build simpler models than the ones that rely on large language models.

3.6.2 Future Work

We encoded a specific kind of background knowledge using WordNet in this work, but the methods described here can be extended to other kinds of structured knowledge sources as well. For example, one can effectively encode entities in text, using structured information from Freebase. Such a model may help tasks like factual question answering.

We compared the performance of our model that relies on knowledge from symbolic knowledge bases to one that relies on large language models in Table 3.1. However, it has to be noted that the two kinds of knowledge are compatible with each other, and potential future work includes models with make use to both kinds of knowledge.

One potential issue with the approach suggested in this chapter is linking ambiguity. That is, deciding which entities in the background knowledge base, or concepts in the background ontology can be linked to the tokens in the input utterance. In this chapter, we did not have to deal with this issue because linking words to WordNet concepts is a trivial process. However, when extending this method to other kinds of knowledge bases, one may need to perform explicit entity or concept linking.

Chapter 4

Leveraging Selectional Preferences as Background Knowledge for Encoding Events

In Chapter 3, we described how explicit symbolic knowledge can help build better encoders of natural language. An assumption we made there is that the background knowledge needed to augment the encoder can be stated symbolically. Accordingly, for the tasks we built systems in Chapter 3: identifying prepositional phrase attachments, and recognizing textual entailment, ontological information from WordNet proved to be useful background knowledge. In this chapter, we relax this assumption, and explore the use of background knowledge that is not necessarily symbolic. To do so, we will turn to the task of modeling newswire events. The background knowledge we use is automatically acquired by modeling selectional preferences in predicate argument structures.

As we described in Chapter 1, our goal here is to fill the gaps in semantics of inputs by explicitly encoding commonsense information, which is usually omitted in human utterances. Here, we do not rely on external sources of knowledge, but instead exploit the information provided by co-occurrence of fillers of slots given by the structure to capture this knowledge. This notion was called Selectional Preference (Wilks, 1973) in earlier literature (see Chapter 2 for more information). While this notion was initially restricted to verbs and their arguments in dependency structures, we hypothesize that it extends to other kinds of structures as well, and consequently, it can be used for modeling appropriate implicit semantic knowledge. We obtain the structured representations of the input utterances from a Semantic Role Labeler, a tool which identifies the eventive predicate in a given input sentence, and its corresponding role fillers. We show that the implicit background knowledge automatically acquired from selectional preferences is helpful in modeling events, by evaluating on a task that requires commonsense knowledge about events.

4.1 Understanding Events

Events are fundamental linguistic elements in human language. Thus understanding events is a prerequisite for deeper semantic analysis of language, and any computational system of human language should have a model of events. One can view an event as an occurrence of a certain action caused by an agent, affecting a patient at a certain time and place. It is the combination of the entities filling the said roles that define an event. Furthermore, certain combinations of these role fillers agree with the general state of the world, and others do not. For example, consider the events described by the following sentences.

Man recovering after being bitten by his dog.

Man recovering after being shot by cops.

The *agent* of the *biting* action is *dog* in the first sentence, and that of the *shooting* action in the second sentence is *cops*. The patient in both sentences is *man*. Both combinations of role-fillers result in events that agree with most people's world view. That is not the case with the event in the following sentence.

Man recovering after being shot by his dog.

This combination does not agree with our expectations about the general state of affairs, and consequently one might find this event strange. While all three sentences above are equally valid syntactically, it is our knowledge about the role fillers —both individually and specifically in combination— that enables us to differentiate between normal and anomalous events. Hence we hypothesize that *anomaly is a result of unexpected or unusual combination of semantic role fillers*.

In this chapter, we introduce the problem of automatic detection of anomalous events and propose a novel event model that can learn to differentiate between normal and anomalous events. We generally define anomalous events as those that are unusual compared to the general state of affairs and might invoke surprise when reported. An automatic anomaly detection algorithm has to encode the goodness of semantic role filler coherence. This is a hard problem since determining what a good combination of role fillers is requires deep semantic and pragmatic knowledge. Moreover, manual judgment of anomaly itself may be difficult and people often may not agree with each other in this regard. We describe the difficulty in human judgment in greater detail in Section 4.3.1. Automatic detection of anomaly requires encoding complex information, which poses the challenge of sparsity due to the discrete representations of meaning, that are words. These problems range from polysemy and synonymy at the lexical semantic level to entity and event coreference at the discourse level.

We define an event as the pair $(V, \mathbf{A})$, where V is the predicate or a semantic verb[1], and $\mathbf{A}$ is the set of its semantic arguments like agent, patient, time, location, so on. Our aim is to obtain a representation of the event that is composed from those of individual words, while explicitly guided by the semantic role structure. This representation can be understood as an embedding of the event in an event space.

4.2 Model for Semantic Anomaly Detection

Neural Event Model (NEM) is a supervised model that learns to differentiate between anomalous and normal events by classifying dense representations of events, otherwise known as event

[1]By semantic verb, we mean an action word whose syntactic category is not necessarily a verb. For example, in *Terrorist attacks on the World Trade Center..*, *attacks* is not a verb but is still an action word.

embeddings. We treat the problem as a binary classification problem, with *normal* and *anomalous* being the two classes. The event embeddings are computed using a structured composition model that represents an event as the composition of the five slot-fillers: **action**, **agent**, **patient**, **time** and **location**. Figure 4.1 shows the pipeline for anomalous event detection using NEM. We first identify the fillers for the five slots mentioned above, by running a PropBank (Palmer et al., 2005) style Semantic Role Labeler (SRL). We use the following mapping of roles to obtain the event structure:

- V $\rightarrow$ action
- A0 $\rightarrow$ agent
- A1 $\rightarrow$ patient
- AM-TMP $\rightarrow$ time
- AM-LOC $\rightarrow$ location

This structured input is then passed to NEM. The model has three components:

Argument Composition This component encodes all the role fillers as vectors in the same space. This is accomplished by embedding the words in each role filler, and then passing them through a Recurrent Neural Network (RNN) with Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) cells. Note that the same LSTM is used for all the slots. Concretely,

$$a_s = \text{LSTM}(I_s) \tag{4.1}$$

where I_s denotes the ordered set of word vector representations in in the filler of slot s, and a_s represents the vector representation of that slot filler.

Event Composition The slot filler representations are then composed using slot specific projection weights to obtain a single event representation. Concretely,

$$\bar{a}_s = \tanh(W_s^\intercal a_s + b_s) \quad \text{where } s \in \{c, g, p, t, l\} \tag{4.2}$$

$$e = \bar{a}_c + \bar{a}_g + \bar{a}_p + \bar{a}_t + \bar{a}_l \tag{4.3}$$

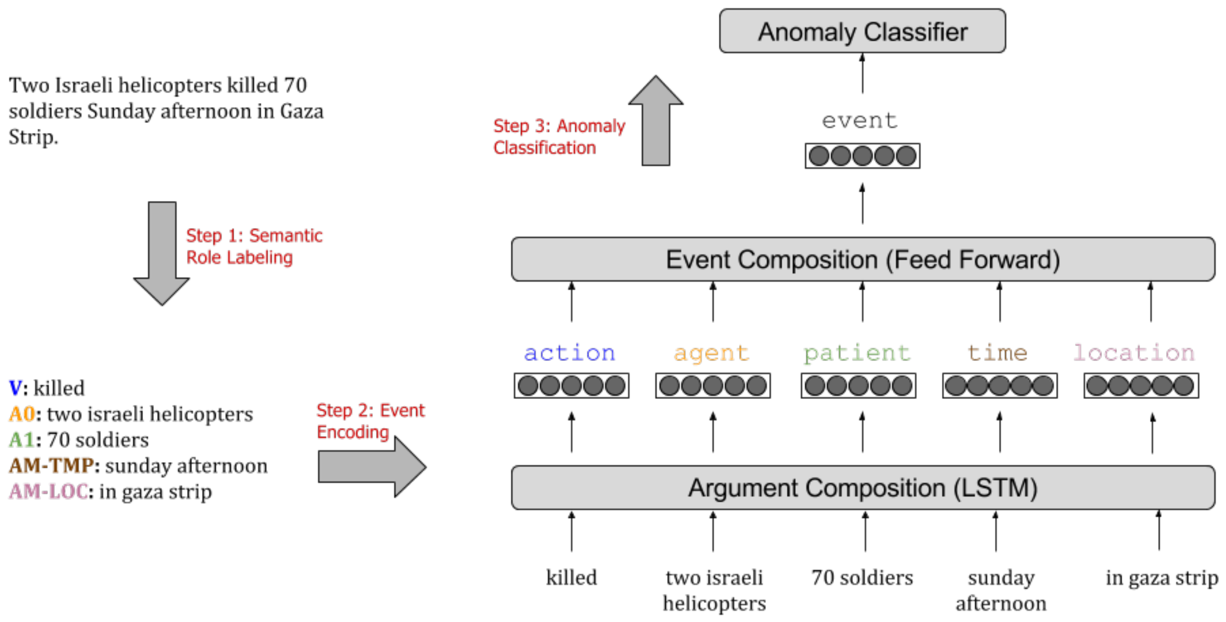
$W_s \in \mathbb{R}^{d \times d}$ and $b_s \in \mathbb{R}^d$ are slot specific projection weights and biases respectively, with s denoting one of action, agent, patient, time and location. d is the dimensionality of the word embeddings. Event composition involves performing non-linear projections of all slot-filler representations, and finally obtaining an event representation e.

Anomaly Classification The event representation is finally passed through a softmax layer to obtain the predicted probabilities for the two classes.

$$p_e = \text{softmax}(W_e^\intercal e + b_e) \tag{4.4}$$

$$\tag{4.5}$$

$W_e \in \mathbb{R}^{d \times 2}$ and $b_e \in \mathbb{R}^2$ are the weight and bias values of the softmax layer respectively. As it can be seen, the layer projects the event representation into a two-dimensional space before applying the softmax non-linearity.



Two Israeli helicopters killed 70 soldiers Sunday afternoon in Gaza Strip.
Step 1: Semantic Role Labeling
V: killed
A0: two israeli helicopters
A1: 70 soldiers
AM-TMP: sunday afternoon
AM-LOC: in gaza strip
Step 2: Event Encoding
Step 3: Anomaly Classification
Anomaly Classifier
event
Event Composition (Feed Forward)
action
agent
patient
time
location
Argument Composition (LSTM)
killed
two israeli helicopters
70 soldiers
sunday afternoon
in gaza strip

Figure 4.1: Anomaly Classification Pipeline with Neural Event Model

4.2.1 Training

Given the output from NEM, we define

$$p_e^1(\theta) = P(\text{anomalous}|e;\theta) \tag{4.6}$$

$$p_e^0(\theta) = P(\text{normal}|e;\theta) \tag{4.7}$$

where θ are the parameters in the model and compute the label loss based on the prediction at the end of anomaly classification as a cross entropy loss as follows:

$$J(\theta) = -l \log p_e^1(\theta) + (1-l)\log p_e^0(\theta) \tag{4.8}$$

where l is the reference binary label indicating whether the event is normal or anomalous. The three components of the model described above: argument composition, event composition and anomaly classification are all trained jointly. That is, we optimize the following objective.

$$\theta^* = \arg\min_{\theta} J(\theta) \tag{4.9}$$

where θ includes all W_s and b_s from all the slots, W_e and b_e from event composition and also all the LSTM parameters from argument composition. We optimize this objective using ADAM (Kingma and Ba, 2014). The model is implemented[2] using Keras (Chollet, 2015). Note that the model described here is an extension of the one described in our previously published paper (Dasigi and Hovy, 2014). The main improvements here include using an LSTM-RNN as the encoder in argument composition instead of a simple RNN, and jointly training argument composition, event composition and anomaly classification, whereas the older model included training argument composition separately using an unsupervised objective.

[2]The code is publicly available at `https://github.com/pdasigi/neural-event-model`

Total number of annotators	22
Normal annotations	56.3%
Strange annotations	28.6%
Highly unusual annotations	10.3%
Cannot Say annotations	4.8%
Avg. events annotated per worker	344
4-way Inter annotator agreement (α)	0.34
3-way Inter annotator agreement (α)	0.56

Table 4.1: Annotation Statistics

4.3 Data

We crawl 3684 "weird news" headlines available publicly on the website of NBC news[3], such as the following:

India weaponizes world's hottest chili.

Man recovering after being shot by his dog.

Thai snake charmer puckers up to 19 cobras.

For training, we assume that the events extracted from this source, called NBC Weird Events (NWE) henceforth, are anomalous. NWE contains 4271 events extracted using SENNA's (Collobert et al., 2011) SRL. We use 3771 of those events as our negative training data. Similarly, we extract events also from headlines in the AFE section of Gigaword, called Gigaword Events (GWE) henceforth. We assume these events are normal. To use as positive examples for training event composition, we sample roughly the same number of events from GWE as our negative examples from NWE. From the two sets, we uniformly sample 1003 events as the test set and validate the labels by getting crowd-sourced annotations.

4.3.1 Annotation

We post the annotation of the test set containing 1003 events as Human Intelligence Tasks (HIT) on Amazon Mechanical Turk (AMT). We break the task into 20 HITs and ask the workers to select one of the four options: *highly unusual*, *strange*, *normal*, and *cannot say* for each event. We ask them to select *highly unusual* when the event seems too strange to be true, *strange* if it seems unusual but still plausible, and *cannot say* only if the information present in the event is not sufficient to make a decision. We publicly release this set of 1003 annotated events for evaluating future research.

Table 4.1 shows some statistics of the annotation task. We compute the Inter-Annotator Agreement (IAA) in terms of Kripendorff's alpha (Krippendorff, 1980). The advantage of using

[3]http://www.nbcnews.com/html/msnbc/3027113/3032524/4429950/4429950_1.html

this measure instead of the more popular Kappa is that the former can deal with missing information, which is the case with our task since annotators work on different overlapping subsets of the test set. The 4-way IAA shown in the table corresponds to agreement over the original 4-way decision (including *cannot say*') while the 3-way IAA is measured after merging the *highly unusual* and *strange* decisions.

Additionally we use MACE (Hovy et al., 2013) to assess the quality of annotation. MACE models the annotation task as a generative process of producing the observed labels conditioned on the true labels and the competence of the annotators, and predicts both the latent variables. The average of competence of annotators, a value that ranges from 0 to 1, for our task is 0.49 for the 4-way decision and 0.59 for the 3-way decision.

We generate true label predictions produced by MACE, discard the events for which the prediction remains to be *cannot say*, and use the rest as the final labeled test set. This leaves 949 events as our test dataset, of which only 41% of the labels are *strange* or *highly unusual*. It has to be noted that even though our test set has equal size samples from both NWE and GWE, the true distribution is not uniform.

4.3.2 Language Model Separability

Given the annotations, we test to see if the sentences corresponding to anomalous events can be separated from normal events using a language model trained on large amounts of unlabeled data. We do this using a Masked Language Model built on top of BERT (Devlin et al., 2018), with pretrained weights from BERT-Base, and measure the perplexity of the sentences in the test set. Figure 4.2 shows a comparison of the perplexity scores for different labels. If the language model features are enough to separate different classes of sentences, one would expect the sentences corresponding to *strange* and *highly unusual* labels to have higher perplexity ranges than *normal* sentences, because the language model is built from a dataset that is expected to have a distribution of sentences where majority of them contain normal events. As can be seen in Figure 4.2, except for a few outliers, most data points in all the categories are in similar perplexity ranges. Hence, sentences with different labels cannot be separated based on language model features.

4.4 Results

We merged the two anomaly classes (strange and highly unusual) and calculated accuracy and F-score (for the anomaly class) of the binary classifier. For empirical analysis, we compare against the following three models:

PMI Model We compare the performance of our model against a baseline that is based on how well the semantic arguments in the event match the selectional preferences of the predicate. We measure selectional preference using Point-wise Mutual Information (PMI) (Church and Hanks, 1990) of the head words of each semantic argument with the predicate. The baseline model is built as follows. We perform dependency parsing using MaltParser (Nivre et al., 2007) on the sentences in the training data used in the first phase of training to obtain the head words of the

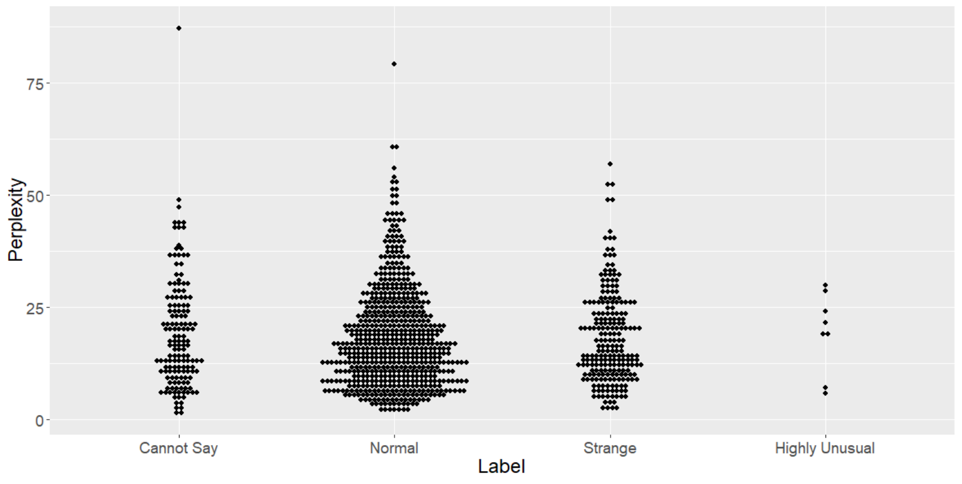


Figure 4.2: Comparison of perplexity scores for different labels. Perplexity scores are measured using a Masked LM model with BERT

semantic arguments. We then calculate the PMI values of all the pairs $< h_A, p >$ where h is the head word of argument A and p is the predicate of the event. For training our baseline classifier, we use the labeled training data from the event composition phase. The features to this classifier are the PMI measures of the $< h_A, p >$ pairs estimated from the larger dataset. The classifier thus trained to distinguish between anomalous and normal events is applied to the test set.

LSTM Model To investigate the importance of our proposed event representation, we also implemented a LSTM baseline that directly encodes sentences. The following equations describe the model.

$$w = \text{LSTM}(I) \tag{4.10}$$

$$\bar{w} = \tanh(W_w^\intercal w + b_w) \tag{4.11}$$

$$p_u = \text{softmax}(W_u^\intercal \bar{w} + b_u) \tag{4.12}$$

Note that this model is comparable in complexity, and has the same number of non-linear transformations as NEM. $\bar{w}$ represents a word level composition instead of an argument-level composition in NEM, and p_u is a probability distribution defined over an <u>u</u>nstructured composition as features. Training of this baseline model is done in the same way as NEM.

OntoLSTM Model In addition to the LSTM baseline, we also compare against the OntoLSTM model we presented in Chapter 3. The point of this comparison is to evaluate whether we gain anything by leveraging implicit background knowledge that cannot be provided by the kind of explicit background knowledge WordNet. We use the same setup for this baseline as we do for the LSTM model, while only changing the encode used.

Table 4.2 shows the results and a comparison with the three baselines. The accuracy of the PMI-based classifier is lower than 50%, which is the expected accuracy of a classifier that assigns labels randomly. As seen in the table, the LSTM model proves to be a strong baseline, but it

	Accuracy	**F-Score**
PMI Baseline	45.2%	45.1%
LSTM Baseline	82.5%	77.4%
OntoLSTM Baseline	**84.9%**	78.5%
NEM	**84.9%**	**79.7%**

Table 4.2: Classification Performance and Comparison with Baselines

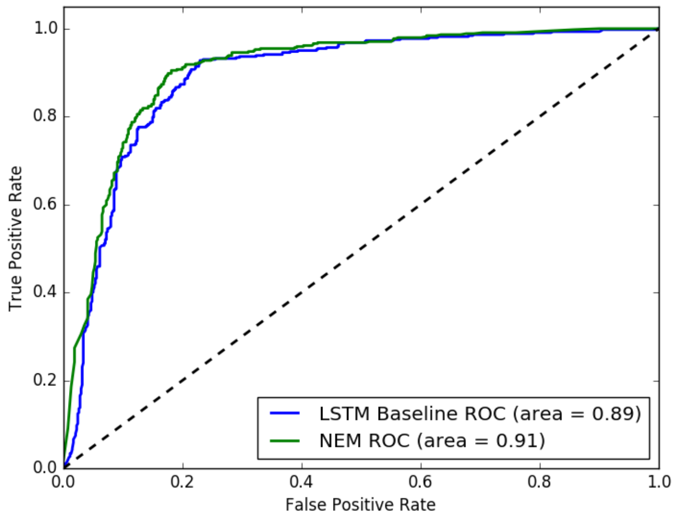


Figure 4.3: ROC curves for the LSTM baseline and NEM

under performs in comparison with NEM, showing the value of our proposed event composition model. The difference between the performance of NEM and the LSTM baseline is statistically significant with $p < 0.0001$. The OntoLSTM baseline does better than the LSTM model, but the fact that our NEM model still outperforms it shows that this task requires modeling implicit background knowledge, not all of which can be obtained from WordNet.

Figure 4.3 shows the ROC curves for NEM and the LSTM baseline.

To further compare NEM with human annotators, we give to MACE, the binary predictions produced by NEM, and those by the LSTM baseline along with the annotations and measure the competence. For the sake of comparison, we also give to MACE, a list of random binary labels as one of the annotations to measure the competence of a hypothetical worker that made random choices. These results are reported in Table 4.3. Unsurprisingly, the competence of the random classifier is very low, and is worse than the least competent human. It can be seen that MACE predicts both the LSTM baseline and NEM to be more competent than human average. This is an encouraging result, especially given that this is a hard problem even for humans.

Human highest	0.896
Human average	0.633
Human lowest	0.144
Random	0.054
LSTM Baseline	0.743
NEM	0.797

Table 4.3: Anomaly Detection Competence

4.5 Conclusion

In this chapter, we have looked at how selectional preferences can be leveraged to model events and represent them in such a way as to distinguish anomalous events from normal events. Results from our controlled experiments comparing the proposed model with one that directly encodes whole sentences have shown the value of using the proposed structured composition.

The model we have shown in this chapter is just one instantiation of using selectional preferences to capture implicit knowledge. In addition to some of the traditional uses of selectional preferences in dependency structures described in Chapter 2, there also exists a large body of work in applications of selectional preferences to relations among entities. Knowledge base completion and relation extraction (Mintz et al., 2009; Sutskever et al., 2009; Nickel et al., 2011; Bordes et al., 2011; Socher et al., 2013; Gardner et al., 2014, among others), fall under this category.

There are limitations to the kind of implicit knowledge that selectional preferences over processed inputs can capture. There is often a need for encoding the kinds of factual or domain-specific commonsense knowledge that are not eventive in nature. For example, understanding texts grounded in real-world information might need knowledge about the physical properties and limitations of objects (Forbes and Choi, 2017). Similarly, understanding procedural texts in a particular domain might require knowledge about state changes and actions (Tandon et al., 2018). These kinds of knowledge may not be explicitly found in knowledge bases, like the commonsense knowledge we leverage in this chapter. However, unlike the eventive knowledge we modeled in this chapter, commonsense knowledge of the two kinds above may not be extracted from text simply using selectional preferences.

Part II

Reasoning with Contextual Knowledge

Chapter 5

Related Work: Learning to Reason

In this chapter, we will discuss reasoning, particularly the kind that involves explicit discrete operations. The work we describe in Chapters 6 and 7 employs discrete operation based reasoning, and this chapter provides ncessary background.

5.1 Learning to Reason with Latent Variables

Reasoning involves making predictions related to the task at hand. Quite often, the process goes through certain intermediate operations that lead to the final prediction. Statistical reasoning models may model the intermediate operations as estimating latent variables on which the final predictions are conditioned. The latent variables may be continuous or discrete, and depending on the task at hand, one choice may be more appropriate than the other. Using continuous latent variables makes training easier since it is generally straight forward to use back-propagation. Discrete latent variable make training harder, as we discuss later in this chapter, but it may be worth dealing with the difficulty in some cases. Let us look at two examples to discuss the pros and cons of both the options.

First, consider an example from the Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2016), with the following context.

> *Nikola Tesla (10 July 1856 – 7 January 1943) was a Serbian American inventor, electrical engineer, mechanical engineer, physicist, and futurist best known for his contributions to the design of the modern alternating current (AC) electricity supply system.*

One question associated with this context is *In what year was Nikola Tesla born?*

Recent reading comprehension systems (Seo et al., 2016; Yu et al., 2018, among others) built for answering questions like the one above, have sub-components that measure the relevance of various parts of the queries to various parts of the corresponding contexts. This is typically done using a soft attention mechanism (Bahdanau et al., 2014b), that produces (normalized) scores of query-context relevance. Relevance in these models is a *continuous* latent variable, and thus its estimation is a continuous intermediate operation. This setup lets the models learn diverse task specific patterns for matching contexts and queries. For example, the model might assign a higher relevance score to the span *what year* in the question, and each of the spans *1856* and *1943* in the context. This is a case where the choice of continuous latent variables is more appropriate.

On the other hand, consider the task of answering the following question given the context below, an example taken from Liang (2016):

Question: *What is the largest prime less than 10?*

Context: *Primes:* $\{$*2, 3, 5, 7, 11, 13,*$\ldots\}$

Clearly, this task requires comparing whether each of the values is less than 10, and finding the largest among a given set of numbers. If we wanted to model the intermediate operations in this task as continuous latent variables, one potential way of doing so would be the following: The model scores the relevance of each span in the context to the span *less than 10* in the question, hopefully learning to score those with numbers that are actually less than 10 higher than those with numbers that are greater. Similarly we hope to the model would also learn to score spans with greater numbers higher than those with smaller numbers, given the span *largest*. That is, the model should essentially learn the semantics of *less than* and *largest*. This direction is an active research area, and is explored in some recent efforts like (Andreas et al., 2016b; Trask et al., 2018; Hudson and Manning, 2018, among others).

In this thesis, we argue that if the task requires just a finite set of well-defined operations (like *less-than*, *max*,etc.), it would be more data-efficient to start with those operations as a set of discrete latent variables. They could provide a useful inductive bias to the model, which could then model the probability of performing a specific operation at a time step, given some representation of the context, the utterance, and the history of operations performed previously. While decoding, when we have the most likely sequence of operations, executing them to produce an answer would be a deterministic operation. Reasoning with semantic parsing and program induction (Krishnamurthy and Mitchell, 2012; Berant et al., 2013b; Artzi and Zettlemoyer, 2013; Kushman et al., 2014; Krishnamurthy et al., 2017, among others) fall under this category.

In this thesis, we model our reasoning tasks as semantic parsing problems. We do not assume that the correct sequence of discrete operations, or in other words, the program or the logical form is not available during training. We train weakly supervised semantic parsers for question, context, denotation triples. First, we will introduce related work in semantic parsing, and then move to the challenges introduced by the lack of direct supervision from logical forms.

5.2 Semantic Parsing

Semantic parsing is the problem of translating human language into a formal language that can be executed against a context. A typical semantic parsing task is question answering against a database, which is accomplished by translating questions into executable logical forms (i.e., programs) that output their answers.

5.2.1 Formal Definition

We formally define the semantic parsing task as follows. Given a dataset where the i^{th} instance is the triple $\{x_i, w_i, d_i\}$, representing a sentence x_i, the world w_i associated with the sentence, and the corresponding denotation d_i, our goal is to find y_i, the translation of x_i in an appropriate logical form language, such that $[\![y_i]\!]^{w_i} = d_i$; i.e., the execution of y_i in world w_i produces the correct denotation d_i. A semantic parser defines a distribution over logical forms given an input utterance: $p(y_i|x_i;\theta)$.

5.2.2 Variations

Semantic parsers vary along a few important dimensions:

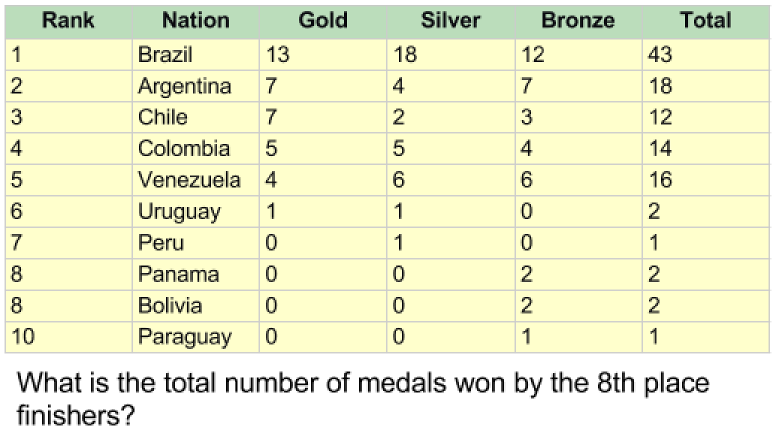
Formalism Early work on semantic parsing used lexicalized grammar formalisms such as Combinatory Categorial Grammar Zettlemoyer and Collins (2005b, 2007); Kwiatkowski et al. (2011, 2013); Krishnamurthy and Mitchell (2012); Artzi and Zettlemoyer (2013) and others Liang et al. (2011b); Berant et al. (2013a); Zhao and Huang (2015); Wong and Mooney (2006, 2007). These formalisms have the advantage of only generating well-typed logical forms, but the disadvantage of introducing latent syntactic variables that make learning difficult. Another approach is to treat semantic parsing as a machine translation problem, where the logical form is linearized then predicted as an unstructured sequence of tokens (Andreas et al., 2013). This approach is taken by recent neural semantic parsers (Jia and Liang, 2016a; Dong and Lapata, 2016b; Locascio et al., 2016; Ling et al., 2016). This approach has the advantage of predicting the logical form directly from the question without latent variables, which simplifies learning, but the disadvantage of ignoring type constraints on logical forms.

In Chapter 6, we introduce a type-constrained neural semantic parser that inherits the advantages of both approaches: it only generates well-typed logical forms and has no syntactic latent variables as every logical form has a unique derivation. Other recent work that explored ideas similar ideas to ours in the context of Python code generation is Yin and Neubig (2017b) and Rabinovich et al. (2017a).

Entity Linking Identifying the entities mentioned in a question is a critical sub-problem of semantic parsing in broad domains and proper entity linking can lead to large accuracy improvements Yih et al. (2015). However, semantic parsers have typically ignored this problem by assuming that entity linking is done beforehand (as the neural parsers above do) or using a simple parameterization for the entity linking portion (as the lexicalized parsers do).

The parser we describe in Chapter 6 explicitly includes an entity linking module that enables it to model the highly ambiguous and implicit entity mentions in WIKITABLEQUESTIONS Pasupat and Liang (2015).

Complexity of contexts and questions Early work that used semantic parsing dealt with data sets such as GEOQUERY (Zelle and Mooney, 1996b) and ATIS (Dahl et al., 1994). They have small domains with only a handful of different predicates. Some of the more recent data sets for question answering against Freebase have a much broader domain, but simple questions (Berant et al., 2013a; Cai and Yates, 2013). WIKITABLEQUESTIONS, one of the datasets we work with in this thesis, is both general domain and comes with complex questions that require nested reasoning, leading to highly compositional logical forms.



Rank	Nation	Gold	Silver	Bronze	Total
1	Brazil	13	18	12	43
2	Argentina	7	4	7	18
3	Chile	7	2	3	12
4	Colombia	5	5	4	14
5	Venezuela	4	6	6	16
6	Uruguay	1	1	0	2
7	Peru	0	1	0	1
8	Panama	0	0	2	2
8	Bolivia	0	0	2	2
10	Paraguay	0	0	1	1

What is the total number of medals won by the 8th place finishers?

Figure 5.1: Example from WIKITABLEQUESTIONS, a dataset that does not provide logical form supervision

5.3 Weak Supervision

Semantic parsers can be trained from labeled logical forms (Zelle and Mooney, 1996b; Zettlemoyer and Collins, 2005b) or question-answer pairs (Liang et al., 2011b; Berant et al., 2013a). In terms of creating datasets, question-answer pairs were considered easier to obtain than labeled logical forms, though recent work has demonstrated that logical forms can be collected efficiently and are more effective (Yih et al., 2016b). However, a key advantage of training with question-answer pairs is that they are agnostic to the domain representation and logical form language (e.g., lambda calculus or λ-DCS).

In terms of modeling, most of the early methods used for training semantic parsers required the training data to come with annotated logical forms (Zelle and Mooney, 1996c; Zettlemoyer and Collins, 2005a). More recent research has focused on training semantic parsers with *weak supervision* (Liang et al., 2011b; Berant et al., 2013b), or trying to automatically infer logical forms from denotations (Pasupat and Liang, 2016). However, matching the performance of a fully supervised semantic parser with only weak supervision remains a significant challenge (Yih et al., 2016a).

Spuriousness The most serious challenge in building semantic parsers with supervision only from denotations is that the search for logical forms results in a large set of those that execute to the correct denotation, but only coincidentally so. These are commonly referred to as spurious logical forms.

The issue is illustrated by the example from WIKITABLEQUESTIONS shown in Figure 5.1. If we searched for logical forms that execute to the correct answer (i.e., 4), we will end up with several logical forms like the following:

- `(sum ((reverse total) (rank 8)))`
- `(sum ((reverse bronze) (rank 8)))`
- `((reverse bronze) (nation colombia))`

- `(count (gold 0))`

The logical form language here (and in Chapter 6) is the λ-DCS based language introduced by Pasupat and Liang (2015) for WIKITABLEQUESTIONS. The column names are (directed) relations from cells to rows. So `(gold 0)` refers to all the rows where the *Gold* column contains 0. Given this, it can be see that the first logical form above is the true translation of the question in Figure 5.1, whereas the second one returns the total number of *Bronze* medals won by *Nations* ranked *8*, the third one returns the number of *Bronze* medals won by *Colombia*, and the last one returns the number of *Nations* that won 0 *Gold* medals, all of which return the correct answer 4. In fact, an exhaustive search could easily return several hundreds of such logical forms, most of which will not be translations of original question. Depending on the amount of supervision provided by the denotations, this issue could be even more serious. For example, in the NLVR dataset (Suhr et al., 2017) (one of the datasets we evaluate on in Chapter 7), the denotations are binary (True or False), and in that case, any random logical form will have a 50% chance of executing to the correct answer. Training a semantic parser on such data will cause the model to quickly overfit to training examples, and the algorithm used should be designed in such a way that the model is biased away from spurious logical forms.

It is worth distinguishing the kind of spuriousness we describe here with another kind of spuriousness that is more commonly discussed in the context of syntactic parsing. The one we described so far is a consequence of the fact that the parsed logical forms can be executed to produce a denotation, and several logical forms with varying semantics can execute to produce the same denotation. Hence, this may be referred to as **semantic spuriousness**. However, depending on the way we define the grammar, we can end up with multiple logical forms with the same semantics, but with different syntax. This may be referred to as **syntactic spuriousness**, and results in the well known *spurious ambiguity* in syntactic parses. Consider the following examples that illustrate the distinction. Let us say we have a different question, but on the same table shown in Figure 5.1: *What is the rank of the country that won no gold medals and one silver medal?*, and a semantic parser produced the following parses:

- `(rank (and (gold 0) (silver 1)))`
- `(rank (and (silver 1) (gold 0)))`
- `(count (silver (> 0)))`

Note that all three logical forms would produce the same answer (*7*), but the first two logical forms have the same semantics. That there can be two different parses which have the same semantics is a consequence of syntactic spuriousness. While both kinds of spuriousness can affect the training algorithm, in the context of weakly supervised semantic parsing, semantic spuriousness is much more serious since answer-driven search algorithms typically produce a lot more semantically spurious logical forms. Consequently, we will focus on semantic spuriousness in the rest of this thesis, and whenever we refer to spuriousness, it will be of the semantic kind.

We now describe prior techniques used for training semantic parsers with weak supervision, and comment on how the issue of spuriousness can be handled in each of them.

5.3.1 Maximum marginal likelihood training

Most work on training semantic parsers from denotations maximizes the likelihood of the denotation given the utterance:

$$\max_{\theta} \prod_{i=1}^{N} p(d_i|x_i;\theta) \tag{5.1}$$

The semantic parsing model itself defines a distribution over *logical forms*, however, not *denotations*, so this maximization must be recast as a *marginalization* over logical forms that evaluate to the correct denotation:

$$\max_{\theta} \prod_{i=1}^{N} \sum_{y_i \in Y_i | [\![y_i]\!]^{w_i} = d_i} p(y_i|x_i;\theta) \tag{5.2}$$

This objective function is called *maximum marginal likelihood* (MML). Y_i here is the set of all valid logical forms in the world corresponding to the i^{th} data instance. The summation over Y_i is in general intractable to perform during training because the set Y_i is too large to enumerate, so it is only approximated.

There are two ways that this approximation has been done in prior work: (1) using a beam search to get the best scoring parses according to the model, hoping that at least some of those parses will yield correct denotations, and using those parses to approximate the inner summation; or (2) performing some kind of (typically bounded-length) heuristic search up front to find a set of logical forms that evaluate to the correct denotation, and using those logical forms to approximate the inner summation. We will call the first method *dynamic MML*, because the logical forms used for the summation change according to the current model parameters, and the second method *static MML*, as the set of logical forms used for training is fixed. Almost all prior work uses dynamic MML (e.g., Liang et al. (2011b); Berant et al. (2013b); Goldman et al. (2018)).

The main benefit of dynamic MML is that it adapts its training signal over time. As the model learns, it can increasingly focus its probability mass on a small set of very likely logical forms. The main benefit of static MML is that there is no need to search during training, so there is a consistent training signal even at the start of training, and it can be made much more efficient than dynamic MML.

Both static and dynamic MML have drawbacks, however. Dynamic MML uses the model to perform search, and it may not find any logical forms that evaluate to the correct denotation, or it may only find spurious logical forms. This problem is exacerbated when the beam size is small, or when the model is in the early stages of training. Without strong lexical cues to guide the model, it can be very difficult for the model to learn anything at all, as it blindly searches a very large space of logical forms without any training signal until its random search happens upon a correct logical form. Traditionally, these lexical cues were provided by the parser's lexicon, though with the advent of neural semantic parsers that remove the lexicon, new techniques for providing lexical hints need to be developed (Goldman et al., 2018).

Static MML, on the other hand, relies heavily on an initial heuristic search to find a good set of likely logical forms that evaluate to the correct denotation. The particulars of this heuristic search can have a large impact on performance; a smaller candidate set will yield a better training signal, but only if the candidate set contains the logical form that matches the semantics of the utterance. In the limit of a single logical form, this becomes a fully-supervised learning problem. However, as the size of the candidate set decreases, the likelihood of the set actually containing the *correct* logical form also decreases. In finding a set of candidate logical forms, we thus need to strike a balance, keeping the set large enough that it is likely to contain the correct logical form, but small enough to be computationally tractable and to provide a strong training signal. In Chapter 6, we use static MML, and leverage the dynamic programming technique of Pasupat and Liang (2016) to get a candidate set of logical forms.

5.3.2 Reinforcement learning methods

When training models with discrete latent variables, a popular choice is to use Reinforcement Learning (RL) techniques. There is a fair amount of prior work that used RL algorithms, particularly policy gradient methods for training weakly supervised semantic parsers. Examples include Andreas et al. (2016a); Liang et al. (2016); Guu et al. (2017); Liang et al. (2018).

These methods optimize the following objective:

$$\max_{\theta} \sum_{i=1}^{N} \sum_{y_i \sim p(y_i|x_i;\theta)} p(y_i|x_i;\theta)\mathcal{R}(y_i) \tag{5.3}$$

The problem of spuriousness can be tackled by defining a reward function that uses some lexical or compositional cues to penalize spurious logical forms, or designing a policy that is explicitly biased away from them. For example, Liang et al. (2018) use a memory-augmented policy to explicitly store high reward programs in a memory buffer, to bias the policy towards those.

Let us now compare objective function above with the MML objective in Equation 5.2. If we define a reward function:

$$\mathcal{R}_{MML}(y_i) = \begin{cases} 1, & \text{if } [\![y_i]\!]^{w_i} = d_i \\ 0, & \text{otherwise} \end{cases} \tag{5.4}$$

we can rewrite the MML objective in Equation 5.2 as:

$$\max_{\theta} \prod_{i=1}^{N} \sum_{y_i \in Y_i} p(y_i|x_i;\theta)\mathcal{R}_{MML}(y_i) \tag{5.5}$$

When rewritten like this, the MML objective in Equation 5.5 is a lot more similar to the RL objective in Equation 5.3. There are two differences that remain: First, in MML, we are maximizing a product of summations, whereas in the RL objective, it is a sum of summations. Second, while we use beam search or a bounded offline search to approximate the inner summation in MML, we use sampling in RL methods.

5.3.3 Structured learning algorithms

Since semantic parsing is a structured prediction problem, algorithms used for other structured prediction problems may be applicable. There exists some prior work (Iyyer et al., 2017; Guu et al., 2017) in using structured learning algorithms for semantic parsing. In Chapter 7, we consider the use of *Minimum Bayes Risk* (MBR) (Goodman, 1996; Goel and Byrne, 2000; Smith and Eisner, 2006) training, which we briefly describe here. MBR trains a model to minimize the expected value of an arbitrary cost function $\mathcal{C}$ as follows:

$$\min_{\theta} \sum_{i=1}^{N} \mathbb{E}_{p(y_i|x_i;\theta)} \mathcal{C}(y_i) \tag{5.6}$$

When applied to semantic parsing, estimating $p(y_i|x_i;\theta)$ could be intractable, again for the same reasons as it is for MML. So we use beam search here, and approximate p as $\tilde{p}$ by *re-normalizing* the probabilities assigned to all logical forms on the beam.

The fact that we can define $\mathcal{C}$ as an arbitrary cost function makes this objective a good choice for weakly supervised semantic parsing. This lets us incorporate additional knowledge to guide the training process, thus effectively dealing with the issue of spuriousness. In Chapter 7, we define the cost function as a linear combination of a coverage measure and denotation accuracy. The coverage measure scores how much any given logical form, y_i overlaps with the functions triggered by the utterance x_i. We show that this additional loss function provides a stronger training signal than a denotation based loss alone.

It may be noted that the MBR objective is also very similar to the RL objective shown in Equation 5.3. If we expand the expectation, and define a reward function as follows

$$\mathcal{R}_{MBR}(y_i) = -\mathcal{C}(y_i) \tag{5.7}$$

the MBR objective becomes

$$\max_{\theta} \sum_{i=1}^{N} \sum_{y_i \in Y} p(y_i|x_i;\theta) \mathcal{R}_{MBR}(y_i) \tag{5.8}$$

So like MML, we use beam search to approximate the inner summation, and like the policy gradient methods in RL, we use an arbitrary reward function.

5.3.4 Bridging objectives

Given the similarities among the objectives discussed so far, prior work has explored bridging objectives to exploit the benefits of each of them, and adapting them to specific problems.

Guu et al. (2017) note the similarity between MML and RL objectives, and define a new objective that addresses shortcomings of both MML and RL in dealing with spurious logical forms when applied to weakly supervised semantic parsing. First, beam search in MML has high bias when it comes to exploring paths that lead to the correct answer, and they address this issue by injecting random noise into exploration, like it is common in policy gradient methods. Second, the gradient weights in RL are proportional to their current probability, causing spurious paths to keep getting higher probability. They propose meritocratic gradient updates to ensure all programs that obtain rewards get upweighted equally, inspired by the fact that MML normalizes its "rewards" for each example. Iyyer et al. (2017) bridge RL techniques with structured learning methods and propose a reward-guided search technique for weakly supervised semantic parsing.

In Chapter 7, we describe a technique for bridging MML and MBR, to iteratively increase the complexity of logical forms that the parser can search for during training.

Chapter 6

Constrained Decoding for Semantic Parsing

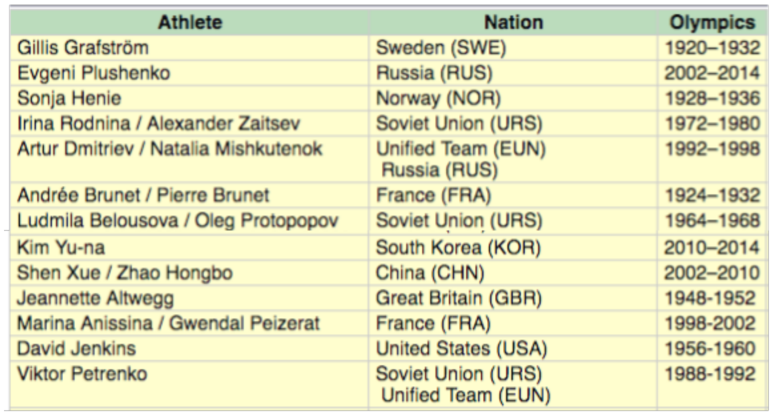
In this chapter, we present an application of knowledge-guided reasoning. We build a neural semantic parser for question answering over a challenging dataset, and show how the model benefits from additional knowledge. In particular, we will discuss how sequence-to-sequence models, which have been shown to work very well for modeling sequences in language, should be adapted to work for semantic parsing. Recent work has shown that such models can be used for semantic parsing by encoding the question then predicting each token of the logical form in sequence (Jia and Liang, 2016a; Dong and Lapata, 2016b). These approaches, while effective, have two major limitations. First, they treat the logical form as an unstructured sequence, thereby ignoring type constraints on well-formed programs. Second, they do not deal with entity linking, which is a critical sub-problem of semantic parsing (Yih et al., 2015). To address these issues, we modify the encoder-decoder model to:

1. Constrain the decoder to only produce syntactically and semantically valid logical forms
2. Incorporate context-awareness into the encoder, and entity linking into the decoder, to effectively score previously unseen entities by linking them to tokens in the utterance, and jointly train these entity embedding anr linking modules from QA supervision.

We assume weak supervision, and train using Static MML (see Section 5.3.1) on pairs of questions and approximate sets of logical forms that yield the correct answer, but are not necessarily true translations of the questions. However, the two main contributions of this chapter are also applicable to the fully supervised setup. We now describe in detail the problems we solve in this chapter.

6.1 Need for Grammar Constraints

Sequence-to-sequence models (Sutskever et al., 2014), have been successfully used for Machine Translation (Sutskever et al., 2014; Cho et al., 2014; Bahdanau et al., 2014a; Wiseman and Rush, 2016; Artetxe et al., 2017, among many others), Summarization (Nallapati et al., 2016; Paulus et al., 2017, among others), and other kinds of text generation tasks like dialogue generation (Li et al., 2017, for one) and generation with style control (Ficler and Goldberg, 2017, for one).



Athlete	Nation	Olympics
Gillis Grafström	Sweden (SWE)	1920–1932
Evgeni Plushenko	Russia (RUS)	2002–2014
Sonja Henie	Norway (NOR)	1928–1936
Irina Rodnina / Alexander Zaitsev	Soviet Union (URS)	1972–1980
Artur Dmitriev / Natalia Mishkutenok	Unified Team (EUN) Russia (RUS)	1992–1998
Andrée Brunet / Pierre Brunet	France (FRA)	1924–1932
Ludmila Belousova / Oleg Protopopov	Soviet Union (URS)	1964–1968
Kim Yu-na	South Korea (KOR)	2010–2014
Shen Xue / Zhao Hongbo	China (CHN)	2002–2010
Jeannette Altwegg	Great Britain (GBR)	1948-1952
Marina Anissina / Gwendal Peizerat	France (FRA)	1998-2002
David Jenkins	United States (USA)	1956-1960
Viktor Petrenko	Soviet Union (URS) Unified Team (EUN)	1988-1992

Figure 6.1: Example context from WIKITABLEQUESTIONS (Pasupat and Liang, 2015)

In all these cases, and also most other cases where encoder-decoder models are used for transducing language inputs, the targets are in natural language as well. That is not the case when the task is semantic parsing, since the targets may be formal meaning representations like λ-calculus (Zettlemoyer and Collins, 2005a, 2007; Kwiatkowski et al., 2011) or λ-DCS (Liang et al., 2011b; Berant and Liang, 2014; Wang et al., 2015b; Pasupat and Liang, 2015), or domain specific meaning representations (Dahl et al., 1994; Zelle and Mooney, 1996a; Kate et al., 2005), or even programming languages (Yin and Neubig, 2017b; Rabinovich et al., 2017a; Iyer et al., 2017; Suhr et al., 2018). This distinction lets us make an efficient task-specific modeling choice. When the target is a formal language, we can easily define a grammar that disallows illegal sequences of target-side tokens, and explicitly incorporate that grammar into the decoder. That way, the decoder will only score valid target-token sequences.

Consider Figure 6.1, which shows part of a table that provides context for answering some questions in the WIKITABLEQUESTIONS dataset. One of the questions is *Which athlete was from South Korea after the year 2010?* When we build a semantic parser for this dataset, the model learns to map questions like these to corresponding logical forms like the following:

```
((reverse athlete) (and (nation south_korea) (year ((reverse
date) (>= 2010-mm-dd) ))))
```

If we used a seq2seq model for this task, and simply defined the target vocabulary to be tokens like { `(`, `)`, `reverse`, `athlete`, `>=`, ... }, then that would let the model produce any sequence of those tokens, including even those that are syntactically invalid (say with mismatched parentheses), or semantically invalid (say with the functions in the logical form taking arguments of incorrect types). Given enough data, the model could learn to assign lower scores to such sequences. But the model still has to spend some learning capacity on distinguishing valid sequences from invalid ones, while we could easily encode the knowledge required to make that distinction as hard constraints, in the form of a grammar. In this chapter we show that doing so lets the model learn the difficult aspects of the semantic parsing more effectively. Using a grammar that is automatically induced from the types of predicates and entities used in logical forms, we transform the decoder into a transition based transducer (i.e., instead of producing tokens like the ones shown above, the decoder produces transition actions from the grammar that incrementally build the logical form top down; details in Section 6.3.2, and an example in Figure 6.3.)

6.2 Need for Entity Linking

While building semantic parsers for unrestricted domains like Wikipedia tables (which is the case with WIKITABLEQUESTIONS), one of the most important challenges is to having to deal with previously unseen entities. This problem gets exacerbated when the model also learns representations for input tokens, since we will not have enough training data to learn representations for most of the entities the model encounters at test time.

To deal with this issue, we first embed a knowledge graph extracted from the table that provides context to answer a given question, such that entities in the graph are embedded as a function of their type and neighborhood. Second, we make the encoded representations of the question context-aware, by augmenting the input representations of the tokens with a *link embedding*, which is an expected value of all the entities to which the given token can be softly linked[1]. Third, we include an attention based entity linking module in the decoder, and score transitions that produce table-specific entities using using this module, thus effectively sidestepping having to learn representations for them.

We train our model using the Static MML objective. As described in Section 5.3.1 in Chapter 5, the objective relies on a precomputed set of approximate logical forms that produce the correct answer, but are not necessarily true translations of the question. We use Dynamic Programming on Denotations (DPD) followed by pruning based on human annotations on fictitious tables, as described in Pasupat and Liang (2016) to obtain that set.

We evaluate our parser on WIKITABLEQUESTIONS. This data set has a broad variety of entities and relations across different tables, along with complex questions that necessitate long logical forms. On this data set, our parser achieves a question answering accuracy of 43.3% and an ensemble of 5 parsers achieves 45.9%. We further perform several ablation studies that demonstrate the importance of both grammar constraints and entity linking to achieving high accuracy on this task.

6.3 Grammar-Constrained Neural Semantic Parser

This section describes our semantic parsing model. The input to our model is a natural language question and a context in which it is to be answered. The model predicts the answer to the question by semantically parsing it to a logical form then executing it against the context.

Our model follows an encoder-decoder architecture, using recurrent neural networks with Long Short Term Memory (LSTM) cells (Hochreiter and Schmidhuber, 1997). The input question and table entities are first encoded as vectors that are then decoded into a logical form (Figure 6.2). We make two significant additions to the standard encoder-decoder architecture. First,

[1]Note that this idea is very similar to the one we used to obtain ontology grounded token representations in Chapter 3

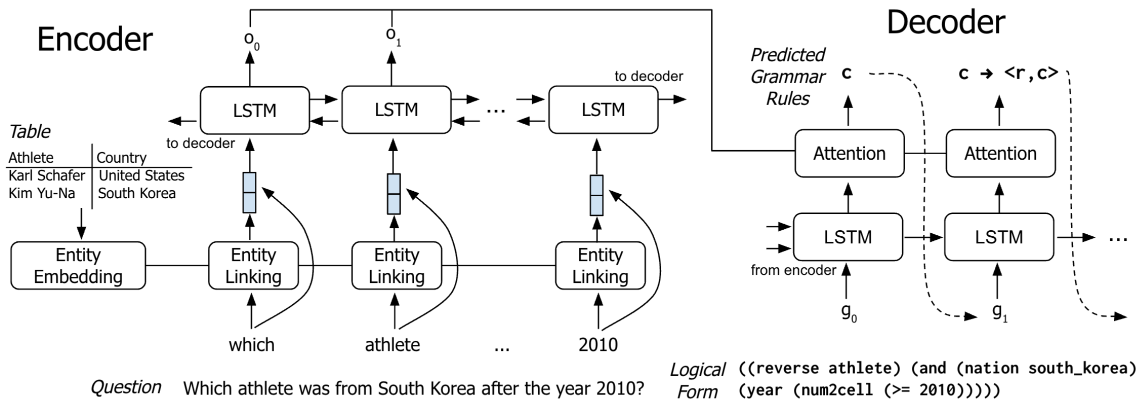

Encoder
Decoder
Table
Athlete | Country
Karl Schafer | United States
Kim Yu-Na | South Korea
Entity Embedding
Entity Linking
LSTM
to decoder
from encoder
Attention
Predicted Grammar Rules
c
c → <r,c>
which
athlete
2010
Question Which athlete was from South Korea after the year 2010?
Logical Form ((reverse athlete) (and (nation south_korea) (year (num2cell (>= 2010)))))

Figure 6.2: Overview of our type-constrained neural semantic parsing model

the encoder includes a special entity embedding and linking module that produces a *link embedding* for each question token that represents the table entities it links to (Section 6.3.1). These link embeddings are concatenated with word embeddings for each question token, then encoded with a bidirectional LSTM. Second, the action space of the decoder is defined by a *type-constrained grammar* which guarantees that generated logical forms satisfy type constraints (Section 6.3.2). The decoder architecture is simply an LSTM with attention that predicts a sequence of generation actions within this grammar.

We train the parser using question-answer pairs as supervision, using an objective based on enumerating logical forms via dynamic programming on denotations (Pasupat and Liang, 2016) (Section 6.4.2). This objective function makes it possible to train neural models with question-answer supervision, which is otherwise difficult for efficiency and gradient variance reasons.

6.3.1 Encoder

The encoder network is a standard bidirectional LSTM augmented with an entity embedding and linking module.

Notation Throughout this section, we denote entities as e, and their corresponding types as $\tau(e)$. The i^{th} token in a question is denoted q_i. We use v_w to denote a learned vector representation (embedding) of word w, e.g., v_{q_i} denotes the vector representation of the ith question token. Finally, we denote the set of all entities as E, and all entities belonging to a type τ as E_τ. The entities E include all of the entities from the table, as well as numeric entities detected in the question by NER.

Entity Embedding The encoder first constructs an embedding for each entity in the knowledge graph given its type and position in the graph. Let $W(e)$ denote the set of words in the name of entity e and $N(e)$ the neighbors of entity e in the knowledge graph. Each entity's embedding r_e

is a nonlinear projection of a type vector $v_{\tau(e)}$ and a neighbor vector $v_{N(e)}$:

$$v_{N(e)} = \sum_{e' \in N(e)} \sum_{w \in W(e')} v_w \tag{6.1}$$

$$r_e = \tanh\left(P_\tau v_{\tau(e)} + P_N v_{N(e)}\right) \tag{6.2}$$

The type vector $v_{\tau(e)}$ is a one-hot vector for $\tau(e)$, with dimension equal to the number of entity types in the grammar. The neighbor vector $v_{N(e)}$ is simply an average of the word vectors in the names of e's neighbors. P_τ and P_N are learned parameter matrices for combining these two vectors.

Entity Linking This module generates a *link embedding* l_i for each question token representing the entities it links to. The first part of this module generates an entity linking score $s(e, i)$ for each entity e and token index i:

$$s(e, i) = \max_{w \in W(e)} v_w^\intercal v_{q_i} + \psi^\intercal \phi(e, i) \tag{6.3}$$

This score has two terms. The first represents similarity in word embedding space between the token and entity name, computed as function of the embeddings of words in $W(e)$ and the word embedding of the ith token, v_{q_i}. The second represents a linear classifier with parameters ψ on features $\phi(e, i)$. The feature function ϕ contains only a few features: exact token match, lemma match, edit distance, an NER indicator feature, and a bias feature. It also includes token and lemma features coming from the neighbors of the node that originally matches the token. We found that features were an effective way to address sparsity in the entity name tokens, many of which appear too infrequently to learn embeddings for.

Finally, the entity embeddings and linking scores are combined to produce a link embedding for each token. The scores $s(e, i)$ are then fed into a softmax layer over all entities e of the same type, and the link embedding l_i is an average of entity vectors r_e weighted by the resulting distribution. We include a null entity, $\varnothing$, in each softmax layer to permit the model to identify tokens that do not refer to an entity. The null entity's embedding is the all-zero vector and its score $s(\varnothing, \cdot) = 0$.

$$p(e|i, \tau) = \frac{\exp s(e, i)}{\sum_{e' \in E_\tau \cup \{\varnothing\}} \exp s(e', i)} \tag{6.4}$$

$$l_i = \sum_{\tau} \sum_{e \in E_\tau} r_e p(e|i, \tau) \tag{6.5}$$

Bidirectional LSTM We concatenate the link embedding l_i and the word embedding v_{q_i} of each token in the question, and feed them into a bidirectional LSTM:

$$x_i = \begin{bmatrix} l_i \\ v_{q_i} \end{bmatrix} \tag{6.6}$$

$$(o_i^f, f_i) = \text{LSTM}(f_{i-1}, x_i) \tag{6.7}$$

$$(o_i^b, b_i) = \text{LSTM}(b_{i+1}, x_i) \tag{6.8}$$

$$o_i = \begin{bmatrix} o_i^f \\ o_i^b \end{bmatrix} \tag{6.9}$$

This process produces an encoded vector representation of each token o_i. The final LSTM hidden states f_n and b_{-1} are concatenated and used to initialize the decoder.

6.3.2 Decoder

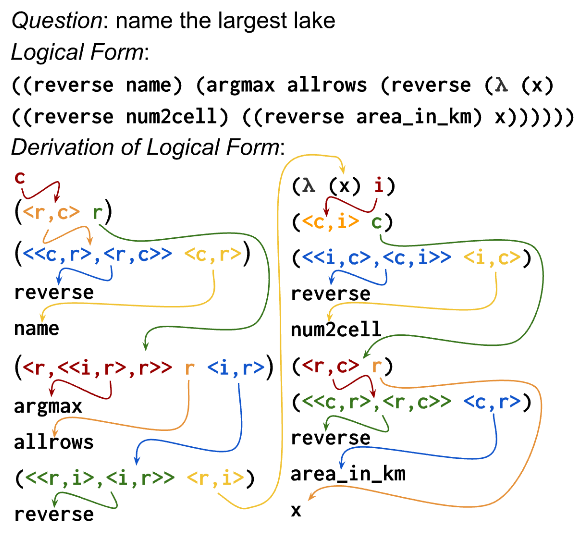
The decoder is an LSTM with attention that selects parsing actions from a grammar over well-typed logical forms.

Type-Constrained Grammar The parser maintains a state at each step of decoding that consists of a logical form with typed *holes*. A hole is a tuple $[\tau, \Gamma]$ of a type τ and a *scope* Γ that contains typed variable bindings, $(x : \alpha) \in \Gamma$. The scope is used to store and generate the arguments of lambda expressions. The grammar consists of a collection of four kinds of production rules on holes:

1. **Application** $[\tau, \Gamma] \to ([\langle \beta, \tau \rangle, \Gamma]\ [\beta, \Gamma])$ rewrites a hole of type τ by applying a function from β to τ to an argument of type β. We also permit applications with more than one argument.
2. **Constant** $[\tau, \Gamma] \to \texttt{const}$ where constant `const` has type τ. This rule generates both context-independent operations such as `argmax` and context-specific entities such as `united_states`.
3. **Lambda** $[\langle \alpha, \tau \rangle, \Gamma] \to\ \lambda x.\ [\tau, \Gamma \cup \{(x : \alpha)\}]$ generates a lambda expression where the argument has type α. x is a fresh variable name.
4. **Variable** $[\tau, \Gamma] \to\ x$ where $(x : \tau) \in \Gamma$. This rule generates a variable bound in a previously-generated lambda expression that is currently in scope.

We instantiate each of the four rules above by replacing the type variables τ, α, β with concrete types, producing, e.g., $[\texttt{c}, \Gamma] \to ([\langle \texttt{r}, \texttt{c} \rangle, \Gamma]\ [\texttt{r}, \Gamma])$ from the application rule. The set of instantiated rules is automatically derived from a corpus of logical forms, which we in turn produce by running dynamic programming on denotations (see Section 6.4.2). Every logical form can be derived in exactly one way using the four kinds of rules above; this derivation is combined with the (automatically-assigned) type of each of the logical form's subexpressions to instantiate the type variables in each rule. We then filter out constant rules that generate context-specific entities (which are handled specially by the decoder) to produce a context-independent grammar.

The first action of the parser is to predict a root type for the logical form, and then decoding proceeds according to the production rules above. Each time step of decoding fills the leftmost


Question: name the largest lake

Logical Form:

`((reverse name) (argmax allrows (reverse (λ (x)`
`((reverse num2cell) ((reverse area_in_km) x))))))`

Derivation of Logical Form:

Figure 6.3: The derivation of a logical form using the type-constrained grammar. The holes in the left column have empty scope, while holes in the right column have scope $\Gamma = \{(x : \texttt{r})\}$

hole in the logical form, and decoding terminates when no holes remain. Figure 6.3 shows the sequence of decoder actions used to generate a logical form.

Network Architecture The decoder is an LSTM that outputs a distribution over grammar actions using an attention mechanism over the encoded question tokens. The decoder also uses a copy-like mechanism on the entity linking scores to generate entities. Say that, during the jth time step, the current hole has type τ. The decoder generates a score for each grammar action whose left-hand side is τ using the following equations:

$$\begin{aligned}
(y_j, h_j) &= \text{LSTM}(h_{j-1}, \begin{bmatrix} g_{j-1} \\ o_{j-1} \end{bmatrix}) && (6.10)\\
a_j &= \text{softmax}(OW^a y_j) && (6.11)\\
o_j &= (a_j)^T O && (6.12)\\
s_j &= W^2_\tau \text{ReLU}(W^1 \begin{bmatrix} y_j \\ o_j \end{bmatrix} + b^1) + b^2_\tau && (6.13)\\
s_j(e_k) &= \sum_i s(e_k, i) a_{ji} && (6.14)\\
p_j &= \text{softmax}(\begin{bmatrix} s_j \\ s_j(e_1) \\ s_j(e_2) \\ \dots \end{bmatrix}) && (6.15)
\end{aligned}$$

The input to the LSTM g_{j-1} is a grammar action embedding for the action chosen in previous time step. g_0 is a learned parameter vector, and h_0 is the concatenated hidden states of the encoder LSTMs. The matrix O contains the encoded token vectors $o_1, \ldots, , o_n$ from the encoder. Equations (6.10) to (6.12) above perform a softmax attention over O using a learned parameter matrix W^a. Equation 6.13 generates scores s for the context-independent grammar rules applicable to type τ using a multilayer perceptron with weights W^1,b^1,W^2_τ,b^2_τ. Equation 6.14 generates a score for each entity e with type τ by averaging the entity linking scores with the current attention a_j. Finally, the context-independent and -dependent scores are concatenated and softmaxed to produce a probability distribution p_j over grammar actions in Equation 6.15. If a context-independent action is chosen, g_j is a learned parameter vector for that action. Otherwise $g_j = g^\tau$, which is a learned parameter representing the selection of an entity with type τ.

6.4 Experiments with WIKITABLEQUESTIONS

We now describe an application of our framework to WIKITABLEQUESTIONS, a challenging dataset for question answering against semi-structured Wikipedia tables (Pasupat and Liang, 2015). An example question from this dataset is shown in Figure 6.1.

6.4.1 Context and Logical Form Representation

We follow (Pasupat and Liang, 2015) in using the same table structure representation and λ-DCS language for expressing logical forms. In this representation, tables are expressed as knowledge graphs over 6 types of entities: cells, cell parts, rows, columns, numbers and dates. Each entity also has a name, which is typically a string value in the table. Our parser uses both the entity names and the knowledge graph structure to construct embeddings for each entity. Specifically, the neighbors of a column are the cells it contains, and the neighbors of a cell are the columns it belongs to.

The logical form language consists of a collection of named sets and entities, along with operators on them. The named sets are used to select table cells, e.g., `united_states` is the set of cells that contain the text "united states". The operators include functions from sets to sets, e.g., the `next` operator maps a row to the next row. Columns are treated as functions from cells to their rows, e.g., `(country united_states)` generates the rows whose `country` column contains "united states". Other operators include reversing relations (e.g., in order to map rows to cells in a certain column), relations that interpret cells as numbers and dates, and set and arithmetic operations. The language also includes aggregation and quantification operations such as `count` and `argmax`, along with λ abstractions that can be used to join binary relations.

Our parser also assigns a type to every λ-DCS expression, which is used to enforce type constraints on generated logical forms. The base types are cells $\texttt{c}$, parts $\texttt{p}$, rows $\texttt{r}$, numbers $\texttt{i}$, and dates $\texttt{d}$. Columns such as `country` have the functional type $\langle \texttt{c}, \texttt{r} \rangle$, representing functions from cells $\texttt{c}$ to rows $\texttt{r}$. Other operations have more complex functional types, e.g., `reverse` has type $\langle \langle \texttt{c}, \texttt{r} \rangle, \langle \texttt{r}, \texttt{c} \rangle \rangle$, which enables us to write `(reverse country)`.[2] The parser as-

[2]Technically, `reverse` has the parametric polymorphic type $\langle \langle \alpha, \beta \rangle, \langle \beta, \alpha \rangle \rangle$, where α and β are *type variables* that can be any type. This type allows `reverse` to reverse any function. However, this is a detail that can largely be ignored. We only use parametric polymorphism when typing logical forms to generate the type-constrained grammar; the grammar itself does not have type variables, but rather a fixed number of concrete instances – such as

signs every λ-DCS constant a type, then applies standard programming language type inference algorithms (Pierce, 2002) to automatically assign types to larger expressions.

6.4.2 Training with Offline Search

Our parser is trained from question-answer pairs, treating logical forms as a latent variable. We use a new loglikelihood objective function for this process that first automatically enumerates a set of correct logical forms for each example, then trains on these logical forms. This objective simplifies the search problem during training and is well-suited to training our neural model.

The training data consists of a collection of n question-answer-table triples, $\{(q^i, a^i, T^i)\}_{i=1}^n$. We first run dynamic programming on denotations (Pasupat and Liang, 2016) on each table T^i and answer a^i to generate a set of logical forms $\ell \in \mathcal{L}^i$ that execute to the correct answer.

Dynamic programming on denotations DPD is an automatic procedure for enumerating logical forms that execute to produce a particular value; it leverages the observation that there are fewer denotations than logical forms to enumerate this set relatively efficiently. It is a kind of chart parsing algorithm where partial parses are grouped together in cells by their output category, size, and the denotation. This is an improvement over the previous work by Pasupat and Liang (2015), where the logical forms were grouped based on only the output category and size. Given this chart, DPD performs two forward passes, In the first pass, the algorithm finds all the cells that lead to the correct denotation. In the second pass, all the rule combinations which lead to the cell corresponding to the correct denotation are listed, and logical forms are generated from those cells, using only the rule combinations that are known to lead to the final cells.

Followed by this process, Pasupat and Liang (2016) also prune the resulting logical forms to remove spurious ones. Recall that as described in Section 5.3, spurious logical forms are those that yield the correct answer, but only coincidentally so, since they are not true translations of the original questions. The insight behind the pruning process is that a correct logical form will produce the correct logical form even if the contents of the table are are shuffled around, whereas a spurious one may not (see the original paper for more details). We use the filtered set of logical forms, and train the model with the following Static MML objective:

$$\mathcal{O}(\theta) = \sum_{i=1}^{n} \log \sum_{\ell \in \mathcal{L}^i} P(\ell | q^i, T^i; \theta) \tag{6.16}$$

We optimize this objective function using stochastic gradient descent. If $|\mathcal{L}^i|$ is small, e.g., 5-10, the gradient of the ith example can be computed exactly by simply replicating the parser's network architecture $|\mathcal{L}^i|$ times, once per logical form. (Note that this takes advantage of the property that each logical form has a unique derivation in the decoder's grammar.) However, $|\mathcal{L}^i|$ often contains many thousands of logical forms, which makes the above computation infeasible. We address this problem by truncating $\mathcal{L}^i$ to the $m = 100$ shortest logical forms, then using a beam search with a beam of $k = 5$ to approximate the sum.

$\langle\langle \mathtt{c}, \mathtt{r}\rangle, \langle \mathtt{r}, \mathtt{c}\rangle\rangle$ – of the above polymorphic type.

We briefly contrast this objective function with two other commonly-used approaches. The first approach is commonly used in prior semantic parsing work with loglinear models (Liang et al., 2011a; Pasupat and Liang, 2015) and uses a similar loglikelihood objective. The gradient computation for this objective requires running a wide beam search, generating, e.g., 300 logical forms, executing each one to identify which are correct, then backpropagating through a term for each. This process would be very expensive with a neural model due to the cost of each backpropagation pass. Another approach is to train the network with REINFORCE (Williams, 1992), which essentially samples a logical form instead of using beam search. This approach is known to be difficult to apply when the space of latent variables is large and the reward signal sparse, as it is in semantic parsing. Our objective improves on these by precomputing correct logical forms in order to avoid searching for them during the gradient computation.

6.4.3 Experimental Setup

We used the standard train/test splits of WIKITABLEQUESTIONS. The training set consists of 14,152 examples and the test set consists of 4,344 examples. The training set comes divided into 5 cross-validation folds for development using an 80/20 split. All data sets are constructed so that the development and test tables are not present in the training set. We report question answering accuracy measured using the official evaluation script, which performs some simple normalization of numbers, dates, and strings before comparing predictions and answers. When generating answers from a model's predictions, we skip logical forms that do not execute (which may occur for some baseline models) or answer with the empty string (which is never correct). All reported accuracy numbers are an average of 5 parsers, each trained on one training fold, using the respective development set to perform early stopping.

We trained our parser with 20 epochs of stochastic gradient descent. We used 200-dimensional word embeddings for the question and entity tokens, mapping all tokens that occurred < 3 times in the training questions to `UNK`. (We tried using a larger vocabulary that included frequent tokens in tables, but this caused the parser to seriously overfit the training examples.) The hidden and output dimensions of the forward/backward encoder LSTMs were set to 100, such that the concatenated representations were also 200-dimensional. The decoder LSTM uses 100-dimensional action embeddings and has a 200-dimensional hidden state and output. The action selection MLP has a hidden layer dimension of 100. We used a dropout probability of 0.5 on the output of both the encoder and decoder LSTMs, as well as on the hidden layer of the action selection MLP. At test time, we decode with a beam size of 10.

6.4.4 Results

Table 6.1 compares the accuracy of our semantic parser to prior work on WIKITABLEQUESTIONS. We distinguish between single models and ensembles, as we expect ensembling to improve accuracy, but not all prior work has used it. Prior work on this data set includes a loglinear semantic parser (Pasupat and Liang, 2015), that same parser with a neural, paraphrase-based reranker (Haug et al., 2017), and a neural programmer that answers questions by predicting a sequence of table operations (Neelakantan et al., 2016). We find that our parser outperforms the best prior result on this data set by 4.6%, despite that prior result using a 15-model ensemble. An

Model	Ensemble Size	Dev.	Test
Neelakantan et al. (2016)	1	34.1	34.2
Haug et al. (2017)	1	-	34.8
Pasupat and Liang (2015)	1	37.0	37.1
Neelakantan et al. (2016)	15	37.5	37.7
Haug et al. (2017)	15	-	38.7
Our Parser	1	42.7	43.3
Our Parser	5	-	**45.9**

Table 6.1: Development and test set accuracy of our semantic parser compared to prior work on WIKITABLEQUESTIONS

ensemble of 5 parsers, one per training fold, improves accuracy by an additional 2.6% for a total improvement of 7.2%.

This ensemble was constructed by averaging the logical form probabilities of parsers trained on each of the 5 cross-validation folds. Note that this ensemble is trained on the entire training set — the development data from one fold is training data for the others — so we therefore cannot report its development accuracy. We investigate the sources of this accuracy improvement in the remainder of this section via ablation experiments.

6.4.5 Type Constraints

Our second experiment measures the importance of type constraints on the decoder by comparing it to sequence-to-sequence (seq2seq) and sequence-to-tree (seq2tree) models. The seq2seq model generates the logical form a token at a time, e.g., [(, (,`reverse`, . . .], and has been used in several recent neural semantic parsers (Jia and Liang, 2016a; Dong and Lapata, 2016b). The seq2tree model improves on the seq2seq model by including an action for generating matched parentheses, then recursively generating the subtree within (Dong and Lapata, 2016b). These baseline models use the same network architecture (including entity embedding and linking) and training regime as our parser, but assign every constant the same type and have a different grammar in the decoder. These models were implemented by preprocessing logical forms and applying a different type system.

Table 6.2 compares the accuracy of our parser to both the seq2seq and seq2tree baselines. Both of these models perform considerably worse than our parser, demonstrating the importance of type constraints during decoding. Interestingly, we found that both baselines typically generate well-formed logical forms: only 7.4% of seq2seq and 6.6% of seq2tree's predicted logical forms failed to execute. Type constraints prevent these errors from occurring in our parser, though the relatively small number of such errors does not does not seem to fully explain the 9% accuracy improvement. We hypothesize that the additional improvement occurs because type constraints also increase the effective capacity of the model, as both the seq2seq and seq2tree models must use some of their capacity to learn the type constraints on logical forms.

Model	**Dev. Accuracy**
seq2seq	31.3
seq2tree	31.6
Our Parser	42.7

Table 6.2: Development accuracy of our semantic parser compared to sequence-to-sequence and sequence-to-tree models.

Model	**Dev. Accuracy**
Full model	42.7
w/o entity embeddings	41.8
token features, no similarity	28.1
all features, no similarity	37.8
similarity only, no features	27.5

Table 6.3: Development accuracy of ablated parser variants trained without parts of the entity linking module.

6.4.6 Entity Embedding and Linking

Our next experiment measures the contribution of the entity embedding and linking module. We trained several ablated versions of our parser, removing both the embedding similarity and featurized classifier from the entity linking module. Table 6.3 shows the accuracy of the resulting models. The results demonstrate that the entity linking features are important, particularly the more complex features beyond simple token matching. In our experience, the “related column” features are especially important for this data set, as columns that appear in the logical form are often not mentioned in the text, but rather implied by a mention of a cell from the column. Embedding similarity alone is not very effective, but it does improve accuracy when combined with the featurized classifier. We found that word embeddings enabled the parser to overfit, which may be due to the relatively small size of the training set, or because we did not use pretrained embeddings. Incorporating pretrained embeddings is an area for future work.

We also examined the effect of the entity embeddings computed using each entity’s knowledge graph context by replacing them with one-hot vectors for the entity’s type. The accuracy of this parser dropped from 42.7% to 41.8%, demonstrating that the knowledge graph embeddings help.

6.4.7 DPD Training

Our final experiment examines the impact on accuracy of varying the number of logical forms m used when training with dynamic programming on denotations. Table 6.4 shows the development accuracy of several parsers trained with varying m. These results demonstrate that using more logical forms generally leads to higher accuracy.

# of logical forms	1	5	10	50	100
Dev. Accuracy	39.7	41.9	41.6	43.1	42.7

Table 6.4: Development accuracy of our semantic parser when trained with varying numbers of logical forms produced by dynamic programming on denotations.

6.4.8 Error Analysis

To better understand the mistakes made by our system, we analyzed a randomly selected set of 100 questions that were answered incorrectly. We identified three major classes of error:

Parser errors (41%): These are examples where a correct logical form is available, but the parser does not select it. A large number of these errors (15%) occur on questions that require selecting an answer from a given list of options, as in *Who had more silvers, Colombia or The Bahamas?* In such cases, the type of the predicted answer is often wrong. Another common subclass is entity linking errors due to missing background knowledge (13%), e.g., understanding that *largest* implicitly refers to the *Area* column.

Representation failures (25%): The knowledge graph representation makes certain assumptions about the table structure and cell values which are sometimes wrong. One common problem is that the graph lacks some cell parts necessary to answer the question (15%). For example, answering a question asking for a state may require splitting cell values in the *Location* column into city and state names. Another common problem is unusual table structures (10%), such as a table listing the number of Olympic medals won by each country that has a final row for the totals. These structures often cause quantifiers such as `argmax` to select the wrong row.

Unsupported operations (11%): These are examples where the logical form language lacks a necessary function. Examples of missing functions are finding consecutive sets of values, computing percentages and performing string operations on cell values.

6.5 Conclusion

We introduced in this chapter, two ways to incorporate contextual knowledge into building neural semantic parsers: grammar constrained decoding, and context linking. As evidenced by the fact that our grammar-constrained model out performs seq2seq and seq2tree models trained on the same data, the constraints provide a useful inductive bias. Moreover, that the most of the logical forms produced by trained seq2seq and seq2tree models are also executable, shows that grammar constraints free our model from having to distinguish valid logical forms from invalid ones, and instead let it learn deeper semantics. Our joint entity embedding and linking module provides an effective way to sidestep the issue of having to learn representations for previously unseen entities.

While our model is weakly supervised, it still assumes a good set of offline searched logical forms, and we relied on the output from DPD, pruned using human annotations. In Chapter 7, we further relax this assumption, and in doing so leverage more contextual knowledge.

Chapter 7

Training Semantic Parsers using Iterative Coverage-Guided Search

We looked at the one instance of knowledge-guided reasoning as semantic parsing in Chapter 6. There we exploited the knowledge of the syntactic and semantic properties of the target language to define a grammar that can constrain the output space in an encoder-decoder model, thereby improving its performance at question-answering on a hard reasoning task. While the training setup we used there was weakly-supervised, and did not assume availability of annotated logical forms, we still had to rely on a *good* set of approximate logical fnorms. In this chapter, we further relax the reliance on supervision, and leverage more contextual knowledge to build a semantic parser from question-answer pairs alone.

As we explained in Chapter 5, training semantic parsers from question-answer pairs typically involves searching over an exponentially large space of logical forms. Neural semantic parsers do not include lexicons to provide guidance during search like their traditional variants did, and thus have no way of differentiating paths leading to correct logical forms from those that lead to spurious ones coincidentally evaluating to the correct answers. To deal with this issue, we propose a search process in this chapter that maximizes the relevance of the retrieved logical forms to the input utterances through a learned lexical mapping, and a novel EM-like iterative training algorithm that alternates between searching for consistent logical forms and maximizing the marginal likelihood of the retrieved ones, with the parameters from the maximization step being used to warm-start the search step in the next iteration.

We demonstrate the effectiveness of these two techniques on two difficult reasoning tasks: WIKITABLEQUESTIONS(WTQ) (Pasupat and Liang, 2015), an open domain task with significant lexical variation, and Cornell Natural Language Visual Reasoning (NLVR) (Suhr et al., 2017), a closed domain task with binary denotations, and thus far less supervision. We show that: 1) interleaving online search and MML over retrieved logical forms (Section 7.2) is a more effective training algorithm than each of those objectives alone; 2) coverage guidance during search (Section 7.1) is helpful for dealing with weak supervision, more so in the case of NLVR where the supervision is weaker; 3) a combination of the two techniques yields 44.3% test accuracy on WTQ, outperforming the previous best single model in a comparable setting, and 82.9% test accuracy on NLVR, outperforming the best prior model, which also relies on greater supervision.

7.1 Coverage-guided search

Weakly-supervised training of semantic parsers relies heavily on lexical cues to guide the initial stages of learning to good logical forms. Traditionally, these lexical cues were provided in the parser's lexicon. Neural semantic parsers remove the lexicon, however, and so need another mechanism for obtaining these lexical cues. In this section we introduce the use of coverage to inject lexicon-like information into neural semantic parsers.

Coverage is a measure of relevance of the candidate logical form y_i to the input x_i, in terms of how well the productions in y_i map to parts of x_i. We use a small manually specified lexicon as a mapping from source language to the target language productions, and define coverage of y_i as the number of productions triggered by the input utterance, according to the lexicon, that are included in y_i.

We use this measure of coverage to augment our loss function, and train using an MBR based algorithm as follows. We use beam search to train a model to minimize the expected value of a cost function $\mathcal{C}$:

$$\min_{\theta} \sum_{i=1}^{N} \mathbb{E}_{\tilde{p}(y_i|x_i;\theta)} \mathcal{C}(x_i, y_i, w_i, d_i) \tag{7.1}$$

where $\tilde{p}$ is a *re-normalization*[1] of the probabilities assigned to all logical forms on the beam.

We define the cost function $\mathcal{C}$ as:

$$\mathcal{C}(x_i, y_i, w_i, d_i) = \lambda \mathcal{S}(y_i, x_i) + (1 - \lambda)\mathcal{T}(y_i, w_i, d_i) \tag{7.2}$$

where the function $\mathcal{S}$ measures the number of items that y_i is missing from the actions (or grammar production rules) triggered by the input utterance x_i given the lexicon; and the function $\mathcal{T}$ measures the consistency of the evaluation of y_i in w_i, meaning that it is 0 if $[\![y_i]\!]^{w_i} = d_i$, or a value e otherwise. We set e as the maximum possible value of the coverage cost for the corresponding instance, to make the two costs comparable in magnitude. λ is a hyperparameter that gives the relative weight of the coverage cost.

7.2 Iterative search

In this section we describe the iterative technique for refining the set of candidate logical forms associated with each training instance.

As discussed in Section 5.3 in Chapter 5, most prior work on weakly-supervised training of semantic parsers uses dynamic MML. This is particularly problematic in domains like NLVR, where the supervision signal is binary—it is very hard for dynamic MML to bootstrap its way to finding good logical forms. To solve this problem, we interleave static MML, which has a consistent supervision signal from the start of training, with the coverage-augmented MBR algorithm described in Section 7.1.

In order to use static MML, we need an initial set of candidate logical forms. We obtain this candidate set using a bounded-length exhaustive search, filtered using heuristics based on

[1]Note that without this re-normalization, and with a -1/0 cost function based on denotation accuracy, MBR will maximize the likelihood of correct logical forms on the beam, which is equivalent to dynamic MML.

the same lexical mapping used for coverage in Section 7.1. A bounded-length search will not find logical forms for the entire training data, so we can only use a subset of the data for initial training. We train a model to convergence using static MML on these logical forms, then use that model to initialize coverage-augmented MBR training. This gives the model a good starting place for the dynamic learning algorithm, and the search at training time can look for logical forms that are longer than could be found with the bounded-length exhaustive search. We train MBR to convergence, then use beam search on the MBR model to find a new set of candidate logical forms for static MML on the training data. This set of logical forms can have a greater length than those in the initial set, because this search uses model scores to not exhaustively explore all possible paths, and thus will likely cover more of the training data. In this way, we can iteratively improve the candidate logical forms used for static training, which in turn improves the starting place for the online search algorithm.

```
Input  : Dataset $\mathcal{D} = \{X, W, D\}$; and
         seed set $\mathcal{D}^0 = \{X^0, Y^0\}$ such that
         $X^0 \subset X$ and $\mathcal{C}(x_i^0, y_i^0, W_i, D_i) = 0$
Output: Model parameters $\theta^{\text{MBR}}$
Initialize dataset $\mathcal{D}^{\text{MML}} = \mathcal{D}^0$;
while $Acc(\mathcal{D}_{dev})$ is increasing do
    $\theta^{\text{MML}} = \texttt{MML}(\mathcal{D}^{\text{MML}})$;
    Initialize $\theta^{\text{MBR}} = \theta^{\text{MML}}$;
    Update $\theta^{\text{MBR}} = \texttt{MBR}(\mathcal{D}; \theta^{\text{MBR}})$;
    Update $\mathcal{D}^{\text{MML}} = \texttt{Decode}(\mathcal{D}; \theta^{\text{MBR}})$;
end
```

Algorithm 1: Iterative coverage-guided search

Algorithm 1 concretely describes this process. `Decode` in the algorithm refers to running a beam search decoder that returns a set of consistent logical forms (i.e. $\mathcal{T} = 0$) for each of the input utterances. We start off with a seed dataset $\mathcal{D}^0$ for which consistent logical forms are available.

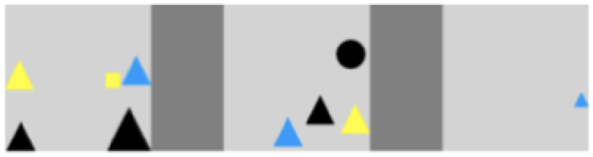
7.3 Task Details

We already described WIKITABLEQUESTIONS in Chapter 6. We will now describe the NLVR dataset in greater detail.

Cornell NLVR is a language-grounding dataset containing natural language sentences provided along with synthetically generated visual contexts, and a label for each sentence-image pair indicating whether the sentence is true or false in the given context. Figure 7.1 shows two example sentence-image pairs from the dataset (with the same sentence). The dataset also comes with structured representations of images, indicating the color, shape, size, and x- and y-coordinates of each of the objects in the image. While we show images in Figure 7.1 for ease of exposition, we use the structured representations in this work.

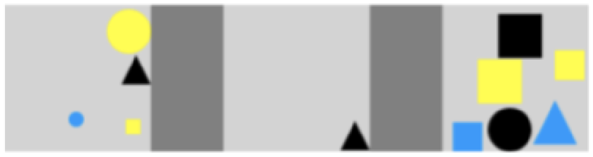
Utterance (x): There is a box with only one item that is blue.

World w_1:



Denotation (d_1): True

World w_2:



Denotation (d_2): False

Logical form (y):
```
(box_exists
  (member_color_any_equals
      (member_count_equals all_boxes 1)
   color_blue))
```

Lexicon triggered actions: `box_exists, 1, color_blue`

Figure 7.1: Example from NLVR dataset showing one sentence associated with two worlds and corresponding binary labels, and translation of the sentence above in our logical form language. Also shown are the actions triggered by the lexicon from the utterance

Recall the formal notation for semantic parsing we introduced in Chapter 5, Section 5.2.1. There we identified that the dataset during training in a weakly supervised setup is typically $\{x_i, w_i, d_i\}_{i=1}^N$, where each utterance x_i comes with a world w_i, and the corresponding denotation d_i. A special property of the NLVR dataset is that the same sentence occurs with multiple worlds. While searching for logical forms that execute to a particular denotation, either during training or otherwise, this property can be exploited as follows: We can define a stricter objective that a logical form for a given utterance must evaluate to the correct denotation (either *True* or *False*) in *all* the worlds it occurs in. We do this, as described in Section 7.1. Accordingly, we tweak the general notation for semantic parsing problems as follows: We define the dataset as $\{x_i, W_i, D_i\}_{i=1}^N$, where $W_i = \{w_i^j\}_{j=1}^M$ is the set of all the worlds that x_i occurs in, and $D_i = \{d_i^j\}_{j=1}^M$ is the set of corresponding denotations.

Going back to Figure 7.1, x_i is *There is a box with only one item that is blue*, the structured representations associated with the two images shown are two of the worlds (w_i^1 and w_i^2), in which a translation of the utterance in some logical form language could be evaluated. The corresponding labels, *True* and *False* respectively, are the denotations d_i^1 and d_i^2 that the translation is supposed to produce, when executed in the two worlds respectively. Figure 7.1 also shows such a translated logical form, written in the language we will describe next.

7.3.1 Logical form languages

NLVR

For NLVR, we define a typed variable-free functional query language, inspired by the GeoQuery language (Zelle and Mooney, 1996c). Our language contains six basic types: `box` (referring to one of the three gray areas in Figure 7.1), `object` (referring to the circles, triangles and squares in Figure 7.1), `shape`, `color`, `number` and `boolean`. The constants in our language are color and shape names, the set of all boxes in an image, and the set of all objects in an image. The functions in our language include those for filtering objects and boxes, and making assertions, a higher order function for handling negations, and a function for querying objects in boxes. This type specification of constants and functions gives us a grammar with 115 productions, of which 101 are terminal productions. (see Figure 7.2 for the complete set of rules in our grammar). Figure 7.1 shows an example of a complete logical form in our language.

WIKITABLEQUESTIONS

For WIKITABLEQUESTIONS we use the language presented in Liang et al. (2018). This is a variable-free language as well. We use this language instead of λ-DCS, like we did in Chapter 6, because we found that the logical forms produced by the variable free language are on average shorter than the λ-DCS ones, which makes the former a good choice for our learning algorithm in this chapter. The lack of variables makes this new language less expressive than λ-DCS, but we actually found that to be helpful in avoiding spurious logical forms to some extent. Figure 7.3 shows an example of a context and a question from WIKITABLEQUESTIONS, along with an associated logical form from the language defined by Liang et al. (2018).

7.3.2 Lexicons for coverage

The lexicon we use for the coverage measure described in Section 7.1 contains under 40 rules for each logical form language. They mainly map words and phrases to constants and unary functions in the target language. The complete lexicons are shown in Tables 7.1 and 7.2. Table 7.1 uses the following placeholders:

$$\begin{aligned}
&\texttt{color} \in \{\texttt{yellow}, \texttt{blue}, \texttt{black}\}\\
&\texttt{shape} \in \{\texttt{square}, \texttt{triangle}, \texttt{circle}\}\\
&\texttt{size} \in \{\texttt{big}, \texttt{medium}, \texttt{small}\}\\
&\texttt{location} \in \{\texttt{above}, \texttt{below}, \texttt{top}, \texttt{left},\\
&\quad \texttt{right}, \texttt{bottom}, \texttt{corner}, \texttt{wall}\}\\
&\texttt{number} \in \{1...9\}
\end{aligned}$$

Figures 7.1 and 7.3 also show the actions triggered by the corresponding lexicons for the utterances shown. We find that small but precise lexicons are sufficient to guide the search process away from spurious logical forms. Moreover, as shown empirically in Section 7.4.4, the model for NLVR does not learn much without this simple but crucial guidance.

Basic types: bool (t), box (b), object (o), shape (s), color (c), number (e)

Constants

b → all_boxes
c → color_black, color_blue, color_yellow
c → color_blue
c → color_yellow
e → 1, 2, 3, 4, 5, 6, 7, 8, 9
o → all_objects
s → shape_circle, shape_square, shape_triangle

Nonterminal Productions

@start@ → t
t → [<b,<e,t>>, b, e], [<b,t>, b], [<o,<c,t>>, o, c]
    [<o,<e,t>>, o, e], [<o,<s,t>>, o, s], [<o,t>, o]
b → [<b,<c,b>>, b, c], [<b,<e,b>>, b, e]
    [<b,<s,b>>, b, s], [<b,b>, b]
o → [<b,o>, b], [<o,o>, o]
<o,o> → [<<o,o>,<o,o>>, <o,o>]

Assertion functions

<b,<e,t>> → box_count_equals,
box_count_greater
box_count_greater_equals
box_count_lesser
box_count_lesser_equals
box_count_not_equals
<b,t> → box_exists
<o,<c,t>> → object_color_all_equals
object_color_any_equals
object_color_none_equals
<o,<e,t>> → object_color_count_equals
object_color_count_greater
object_color_count_greater_equals
object_color_count_lesser
object_color_count_lesser_equals
object_color_count_not_equals
object_count_equals
object_count_greater
object_count_greater_equals
object_count_lesser
object_count_lesser_equals
object_count_not_equals
object_shape_count_equals
object_shape_count_greater
object_shape_count_greater_equals
object_shape_count_lesser
object_shape_count_lesser_equals
object_shape_count_not_equals
<o,<s,t>> → object_shape_all_equals
object_shape_any_equals
object_shape_none_equals
<o,t> → object_exists

Box filtering functions

<b,<s,b>> → member_shape_all_equals,
member_shape_any_equals,
member_shape_none_equals,
<b,<c,b>> → member_color_all_equals,
member_color_any_equals,
member_color_none_equals,
member_color_count_equals,
<b,<e,b>> → member_color_count_greater,
member_color_count_greater_equals,
member_color_count_lesser,
member_color_count_lesser_equals,
member_color_count_not_equals,
member_count_equals,
member_count_greater,
member_count_greater_equals,
member_count_lesser,
member_count_lesser_equals,
member_count_not_equals,
member_shape_count_equals,
member_shape_count_greater,
member_shape_count_greater_equals,
member_shape_count_lesser,
member_shape_count_lesser_equals,
member_shape_count_not_equals,
<b,b> → member_color_different,
member_color_same,
member_shape_different,
member_shape_same

Box membership function

<b,o> → object_in_box

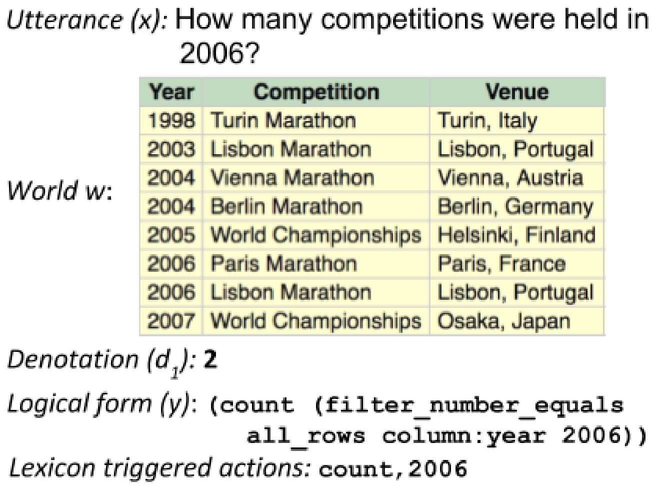
Object filtering functions

<o,o> → above, below, big, black, blue, bottom,
circle, medium, same_color,
same_shape, small, square, top,
touch_bottom, touch_corner, touch_left
touch_object, touch_right, touch_top,
touch_wall, triangle, yellow

Negation (Higher order function)

<<o,o>, <o,o>> → negate_filter

Figure 7.2: Complete grammar used by our parser for the NLVR domain



Utterance (x): How many competitions were held in 2006?

World w:

Year	Competition	Venue
1998	Turin Marathon	Turin, Italy
2003	Lisbon Marathon	Lisbon, Portugal
2004	Vienna Marathon	Vienna, Austria
2004	Berlin Marathon	Berlin, Germany
2005	World Championships	Helsinki, Finland
2006	Paris Marathon	Paris, France
2006	Lisbon Marathon	Lisbon, Portugal
2007	World Championships	Osaka, Japan

Denotation (d_1): **2**

Logical form (y):
```
(count (filter_number_equals
     all_rows column:year 2006))
```
Lexicon triggered actions: `count,2006`

Figure 7.3: Example from WIKITABLEQUESTIONS dataset showing an utterance, a world, associated denotation, corresponding logical form, and actions triggered by the lexicon.

there is a box → `box_exists`
there is a [other] → `object_exists`
box . . . [color] → `color_[color]`
box . . . [shape] → `shape_[shape]`
not → `negate_filter`
contains → `object_in_box`
touch . . . [location] → `touch_[location]`
[location] → `[location]`
[shape] → `[shape]`
[color] → `[color]`
[size] → `[size]`
[number] → `[number]`

Table 7.1: Coverage Lexicon for NLVR

at least $\rightarrow$ `filter`$_{\geq}$
[greater|larger|more] than $\rightarrow$ `filter`$_{\geq}$
at most $\rightarrow$ `filter`$_{\leq}$
no [greater|larger|more] than $\rightarrow$ `filter`$_{\leq}$
[next|below|after] $\rightarrow$ `next`
[previous|above|before] $\rightarrow$ `previous`
[first|top] $\rightarrow$ `top`
[last|bottom] $\rightarrow$ `bottom`
same $\rightarrow$ `same_as`
total $\rightarrow$ `sum`
difference $\rightarrow$ `diff`
average $\rightarrow$ `average`
[least|smallest|lowest|smallest] $\rightarrow$ `argmin`
[most|longest|highest|largest] $\rightarrow$ `argmax`
[what|when] ... [last|least] $\rightarrow$ `min`
[what|when] ... [first|most] $\rightarrow$ `max`
how many $\rightarrow$ `count`

Table 7.2: Coverage Lexicon for WIKITABLEQUESTIONS

7.4 Experiments

We evaluate both our contributions on NLVR and WIKITABLEQUESTIONS.

7.4.1 Model

In this work, we use a grammar-constrained encoder-decoder neural semantic parser for our experiments. Of the many variants of this basic architecture, all of which are essentially seq2seq models with constrained outputs and/or re-parameterizations, we choose to use the parser of Krishnamurthy et al. (2017), as it is particularly well-suited to the WIKITABLEQUESTIONS dataset, which we evaluate on.

The encoder in the model is a bi-directional recurrent neural network with Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) cells, and the decoder is a grammar-constrained decoder also with LSTM cells. Instead of directly outputting tokens in the logical form, the decoder outputs *production rules* from a CFG-like grammar. These production rules sequentially build up an abstract syntax tree, which determines the logical form. The model also has an entity linking component for producing table entities in the logical forms; this component is only applicable to WIKITABLEQUESTIONS, and we remove it when running experiments on NLVR. The particulars of the model are not the focus of this work, so we refer the reader to the original paper for more details.

In addition, we slightly modify the constrained decoding architecture from Krishnamurthy et al. (2017) to bias the predicted actions towards those that would decrease the value of $\mathcal{S}(y_i, x_i)$. This is done using a coverage vector, $v_i^{\mathcal{S}}$ for each training instance that keeps track of the production rules triggered by x_i, and gets updated whenever one of those desired productions is produced by the decoder. That is, v_i^S is a vector of 1s and 0s, with 1s indicating the triggered productions that are yet to be produced by the decoder. This is similar to the idea of checklists used by Kiddon et al. (2016). The decoder in the original architecture scores output actions at each time step by computing a dot product of the predicted action representation with the embeddings of each of the actions. We add a weighted sum of all the actions that are yet to produced:

$$s_i^a = e^a.(p_i + \gamma * v_i^S.E) \tag{7.3}$$

where s_i^a is the score of action a at time step i, e^a is the embedding of that action, p_i is the predicted action representation, E is the set of embeddings of all the actions, and γ is a learned parameter for regularizing the bias towards yet-to-be produced triggered actions.

7.4.2 Experimental setup

NLVR We use the standard train-dev-test split for NLVR, containing 12409, 988 and 989 sentence-image pairs respectively. NLVR contains most of the sentences occurring in multiple worlds (with an average of 3.9 worlds per sentence). We set the word embedding and action embedding sizes to 50, and the hidden layer size of both the encoder and the decoder to 30. We initialized all the parameters, including the word and action embeddings using Glorot uniform initialization (Glorot and Bengio, 2010). We found that using pretrained word representations did not help. We added a dropout (Srivastava et al., 2014) of 0.2 on the outputs of the encoder and the decoder and before predicting the next action, set the beam size to 10 both during training and at test time, and trained the model using ADAM (Kingma and Ba, 2014) with a learning rate of 0.001. All the hyper-parameters are tuned on the validation set.

WIKITABLEQUESTIONS This dataset comes with five different cross-validation folds of training data, each containing a different 80/20 split for training and development. We first show results aggregated from all five folds in Section 7.4.3, and then show results from controlled experiments on fold 1. We replicated the model presented in Krishnamurthy et al. (2017), and only changed the training algorithm and the language used. We used a beam size of 20 for MBR during training and decoding, and 10 for MML during decoding, and trained the model using Stochastic Gradient Descent (Kiefer et al., 1952) with a learning rate of 0.1, all of which are tuned on the validation sets.

Specifics of iterative search For our iterative search algorithm, we obtain an initial set of candidate logical forms in both domains by exhaustively searching to a depth of 10[2]. During search we retrieve the logical forms that lead to the correct denotations in all the corresponding worlds, and sort them based on their coverage cost using the coverage lexicon described in Section 7.3.2, and choose the top-k[3]. At each iteration of the search step in our iterative training algorithm, we increase the maximum depth of our search with a step-size of 2, finding more complex logical

[2]It was prohibitively expensive to search beyond depth of 10.

[3]k is a hyperparameter that is chosen on the dev set at each iteration in iterative search, and is typically 10 or 20

Approach	**Dev**	**Test**
Pasupat and Liang (2015)	37.0	37.1
Neelakantan et al. (2017)	34.1	34.2
Haug et al. (2018)	-	34.8
Zhang et al. (2017)	40.4	43.7
Liang et al. (2018) (MAPO) (mean $\pm$ std.)	42.3 $\pm$ 0.3	43.1 $\pm$ 0.5
Liang et al. (2018) (MAPO) (best)	42.7	43.8
Iterative Search (mean $\pm$ std.)	42.1 $\pm$ 1.1	43.9 $\pm$ 0.3
Iterative Search (best)	**43.1**	**44.3**

Table 7.3: Comparison of single model performances of Iterative Search with previously reported single model performances

forms and covering a larger proportion of the training data. While exhaustive search is prohibitively expensive beyond a fixed number of steps, our training process that uses beam search based approximation can go deeper.

Implementation We implemented our model and training algorithms within the AllenNLP (Gardner et al., 2018) toolkit. The code and models are publicly available at `https://github.com/allenai/iterative-search-semparse`.

7.4.3 Main results

WikiTableQuestions Table 7.3 compares the performance of a single model trained using Iterative Search, with that of previously published single models. We excluded ensemble models since there are differences in the way ensembles are built for this task in previous work, either in terms of size or how the individual models were chosen. We show both best and average (over 5 folds) single model performance from Liang et al. (2018) (Memory Augmented Policy Optimization). The best model was chosen based on performance on the development set. Our single model performances are computed in the same way. Note that Liang et al. (2018) also use a lexicon similar to ours to prune the seed set of logical forms used to initialize their memory buffer.

In Table 7.4, we compare the performance of our iterative search algorithm with three baselines: 1) Static MML, as described in Section 5.3.1 trained on the candidate set of logical forms obtained through the heuristic search technique described in Section 7.4.2; 2) Iterative MML, also an iterative technique but unlike iterative search, we skip MBR and iteratively train static MML models while increasing the number of decoding steps; and 3) MAPO (Liang et al., 2018), the current best published system on WTQ. All four algorithms are trained and evaluated on the first fold, use the same language, and the bottom three use the same model and the same set of logical forms used to train static MML.

NLVR In Table 7.5, we show a comparison of the performance of our *iterative coverage-guided search* algorithm with the previously published approaches for NLVR. The first two rows corre-

Algorithm	Dev acc.	Test acc.
MAPO	42.1	42.7
Static MML	40.0	42.2
Iterative MML	42.5	43.1
Iterative Search	**43.0**	**43.8**

Table 7.4: Comparison of iterative search with static MML, iterative MML, and the previous best result from Liang et al. (2018), all trained on the official split 1 of WIKITABLEQUESTIONS and tested on the official test set.

	Dev.		Test-P		Test-H	
Approach	Acc.	Cons.	Acc.	Cons.	Acc.	Cons.
MaxEnt (Suhr et al., 2017)	68.0	-	67.7	-	67.8	-
BiATT-Pointer (Tan and Bansal, 2018)	74.6	-	73.9	-	71.8	-
Abs. Sup. (Goldman et al., 2018)	84.3	66.3	81.7	60.1	-	-
Abs. Sup. + ReRank (Goldman et al., 2018)	85.7	67.4	84.0	65.0	82.5	63.9
Iterative Search	85.4	64.8	82.4	61.3	**82.9**	**64.3**

Table 7.5: Comparison of our approach with previously published approaches. We show accuracy and consistency on the development set, and public (Test-P) and hidden (Test-H) test sets.

spond to models that are not semantic parsers. This shows that semantic parsing is a promising direction for this task. The closest work to ours is the weakly supervised parser built by Goldman et al. (2018). They build a lexicon similar to ours for mapping surface forms in input sentences to abstract clusters. But in addition to defining a lexicon, they also manually annotate complete sentences in this abstract space, and use those annotations to perform data augmentation for training a supervised parser, which is then used to initialize a weakly supervised parser. They also explicitly use the abstractions to augment the beam during decoding using caching, and a separately-trained discriminative re-ranker to re-order the logical forms on the beam. As a discriminative re-ranker is orthogonal to our contributions, we show their results with and without it, with "Abs. Sup." being more comparable to our work. Our model, which uses no data augmentation, no caching during decoding, and no discriminative re-ranker, outperforms their variant without reranking on the public test set, and outperforms their best model on the hidden test set, achieving a new state-of-the-art result on this dataset.

7.4.4 Effect of coverage-guided search

To evaluate the contribution of coverage-guided search, we compare the the performance of the NLVR parser in two different settings: with and without coverage guidance in the cost function. We also compare the performance of the parser in the two settings, when initialized with parameters from an MML model trained to maximize the likelihood of the set of logical forms obtained from exhaustive search. Table 7.6 shows the results of this comparison. We measure

	No coverage		+ coverage	
	Acc.	Cons.	Acc.	Cons.
No init.	56.4	12.0	73.9	43.6
MML init.	77.7	51.1	80.7	56.4

Table 7.6: Effect of coverage guidance on NLVR parsers trained with and without initialization from an MML model. Metrics shown are accuracy and consistency on the public test set.

Iter.	Length	% cov.	Step	Dev. Acc
0	10	51	M	64.0
1	12	65	S	81.6
			M	76.5
2	14	65	S	82.7
			M	81.8
3	16	73	S	85.4
			M	83.1
4	18	75	S	84.7
			M	81.2

Table 7.7: Effect of iterative search (S) and maximization (M) on NLVR. % cov. is the percentage of training data for which the S step retrieves consistent logical forms.

accuracy and consistency of all four models on the publicly available test set, using the official evaluation script. Consistency here refers to the percentage of logical forms that produce the correct denotation in all the corresponding worlds, and is hence a stricter metric than accuracy. The cost weight (λ in Equation 7.2) was tuned based on validation set performance for the runs with coverage, and we found that $\lambda = 0.4$ worked best.

It can be seen that both with and without initialization, coverage guidance helps by a big margin, with the gap being even more prominent in the case where there is no initialization. When there is neither coverage guidance nor a good initialization, the model does not learn much from unguided search and get a test accuracy not much higher than the majority baseline of 56.2%.

We found that coverage guidance was not as useful for WTQ. The average value of the best performing λ was around 0.2, and higher values neither helped nor hurt performance.

7.4.5 Effect of iterative search

To evaluate the effect of *iterative search*, we present the accuracy numbers from the search (S) and maximization (M) steps from different iterations in Tables 7.7 and 7.8, showing results on NLVR and WTQ, respectively. Additionally, we also show number of decoding steps used at each iterations, and the percentage of sentences in the training data for which we were able to obtain consistent logical forms from the S step, the set that was used in the M step of the same iteration.

Iter.	Length	% cov.	Step	Dev. Acc
0	10	83.3	M	40.0
1	12	70.2	S	42.5
			M	42.5
2	14	71.3	S	43.1
			M	42.7
3	16	71.0	S	42.8
			M	42.5
4	18	71.0	S	43.0
			M	42.7

Table 7.8: Iterative search on WIKITABLEQUESTIONS. M and S refer to Maximization and Search steps.

0	*There is a tower with four blocks* `(box_exists (member_count_equals all_boxes 4))`
1	*Atleast one black triangle is not touching the edge* `(object_exists (black (triangle ((negate_filter touch_wall) all_objects))))`
2	*There is a yellow block as the top of a tower with exactly three blocks.* `(object_exists (yellow (top (object_in_box (member_count_equals all_boxes 3)))))`
3	*The tower with three blocks has a yellow block over a black block* `(object_count_greater_equals (yellow (above (black (object_in_box (member_count_equals all_boxes 3))))) 1)`

Table 7.9: Complexity of logical forms produced at different iterations, from iteration 0 to iteration 3; each logical form could not be produced at the previous iterations

It can be seen in both tables that a better MML model gives a better initialization for MBR, and a better MBR model results in a larger set of utterances for which we can retrieve consistent logical forms, thus improving the subsequent MML model. The improvement for NLVR is more pronounced (a gain of 21% absolute) than for WTQ (a gain of 3% absolute), likely because the initial exhaustive search provides a much higher percentage of spurious logical forms for NLVR, and thus the starting place is relatively worse.

Complexity of Logical Forms We analyzed the logical forms produced by our iterative search algorithm at different iterations to see how they differ. As expected, for NLVR, allowing greater depths lets the parser explore more complex logical forms. Table 7.9 shows examples from the validation set that indicate this trend.

7.5 Related Work

The main contributions of this work deal with training semantic parsers with weak supervision, and we gave a detailed discussion of related training methods in Section 5.3 in Chapter 5.

We evaluate our contributions on the NLVR and WIKITABLEQUESTIONS datasets. Other work that evaluates on on these datasets include Goldman et al. (2018), Tan and Bansal (2018), Neelakantan et al. (2017), Krishnamurthy et al. (2017), Haug et al. (2018), and Liang et al. (2018). These prior works generally present modeling contributions that are orthogonal (and in some cases complementary) to the contributions of this paper. There has also been a lot of recent work on neural semantic parsing, most of which is also orthogonal to (and could probably benefit from) our contributions Dong and Lapata (2016c); Jia and Liang (2016b); Yin and Neubig (2017a); Krishnamurthy et al. (2017); Rabinovich et al. (2017b). Recent attempts at dealing with the problem of spuriousness include Misra et al. (2018) and Guu et al. (2017).

Coverage has recently been used in machine translation Tu et al. (2016) and summarization (See et al., 2017). There have also been many methods that use coverage-like mechanisms to give lexical cues to semantic parsers. Goldman et al. (2018)'s abstract examples is the most recent and related work, but the idea is also related to lexicons in pre-neural semantic parsers (Kwiatkowski et al., 2011).

7.6 Conclusion

Summary We have presented a new technique for training semantic parsers with weak supervision. Our key insights are that lexical cues are crucial for guiding search during the early stages of training, and that the particulars of the approximate marginalization in maximum marginal likelihood have a large impact on performance. To address the first issue, we used a simple coverage mechanism for including lexicon-like information in neural semantic parsers that don't have lexicons. For the second issue, we developed an iterative procedure that alternates between statically-computed and dynamically-computed training signals. Together these two contributions greatly improve semantic parsing performance, leading to a new state-of-the-art result on NLVR. As these contributions are to the learning algorithm, they are broadly applicable to many models trained with weak supervision, and we demonstrate this with a significant gain to a baseline parser on WIKITABLEQUESTIONS.

Future Work In this chapter, we dealt with the issue of lack of direct supervision from logical forms by relying on indirect supervision from minimal lexicons and automatically generated approximate sets of logical forms. Alternative approaches for dealing with this issue include annotating complete logical forms either manually or semi-automatically (Goldman et al., 2018) or using human annotations to filter automatically produced logical forms (Pasupat and Liang, 2016), and using them as training data. While we show that the methods presented in this chapter can outperform those approaches (Tables 7.5 and 7.4), it has to be noted that our methods are also orthogonal to those approaches. One potential direction for future work would be to analyze how coverage from minimal lexicons and iterative search interact with complete logical form annotations. For example, given a coverage lexicon, it would be interesting to analyze how many utterances need to be annotated before the performance on a test set plateaus. It would also be interesting to have a small set of questions annotated, and automatically learn lexical anchoring rules by from those logical forms, and use them for coverage-guidance, in a directly supervised model trained on those logical forms.

Chapter 8

Conclusion

8.1 Summary

In this thesis we identified two classes of knowledge relevant to language understanding: *background* and *contextual*.

We defined background knowledge as the implicit shared human knowledge that is often omitted in human generated text or speech. We presented two ways to incorporate this kind of knowledge in NLU systems. In Chapter 3, we explored how knowledge stated in an explicit symbolic knowledge base can be incorporated in an NLU system to produce better encoded representations. Particularly, we linked words to an ontology (WordNet), and built an ontology-aware encoder that exploited the WordNet hierarchies to learn context-sensitive token representations of words. We showed the proposed model helps textual entailment and prepositional phrase attachment tasks. By giving the encoder access to the hypernym paths of relevant concepts we showed that we can learn useful task-dependent generalizations using an attention mechanism. Then in Chapter 4, we noted that not all background knowledge can be stated symbolically, and focused on the kind of background knowledge that is implicit in relations between linguistic elements. Concretely, we showed how selectional preferences in structures relevant to the task at hand can be leveraged to encode background knowledge. We showed that this is indeed more effective than encoding whole sentences using comparable neural network architectures.

We defined contextual knowledge as the explicit additional information that reading comprehension systems need to ground reasoning in the respective contexts. We focused on a subclass of reasoning tasks where reasoning can be broken down into a sequence of discrete operations over structured contexts. We viewed such problems as semantic parsing into domain-specific languages. Given this framework, we pursued four research goals related to incorporating contextual knowledge into semantic parsers, the first two related to the model architecture while building transition based semantic parsers using neural encoder-decoder models, and the remaining two related to the learning algorithms used when only weak supervision is available. First, in Chapter 6, we investigated how the knowledge of syntactic and semantic properties of the domain specific language can be exploited to restrict the search space of the decoder in an encoder-decoder model built for semantic parsing. Second, also in Chapter 6, we dealt with the issue of reasoning over previously unseen entities by using an entity linking module to score entity productions. Then, in Chapter 7 we dealt with the issue of spuriousness while training weakly supervised semantic parsers. We incorporated minimal lexicons into the parsers to define a measure of coverage, thereby ensuring that the semantic parsers are penalized for producing logical forms that are not relevant to the input utterances. Finally, we introduced a novel training scheme that alternates between exploring the search space for new logical forms, and maximizing the likelihood of the retrieved ones. We showed that this scheme effectively exploits the compositionality of the logical form language to bias the model towards good paths. Using these techniques, we built parsers for two hard tasks: WIKITABLEQUESTIONS and NLVR, and showed state-of-the-art results on both tasks.

8.2 Future work in Knowledge-Aware Encoding

Following are some potential directions for future work that can be built on top of the research ideas in Chapters 3 and 4.

We explored the use of a kind of symbolic knowledge, coming from an ontology, in an end-to-end NLU system in Chapter 3. This idea can be extended to other NLU tasks while using appropriate knowledge sources. For example, factual knowledge from Freebase may be helpful while building open domain question answering systems, or knowledge from a protein interaction database could be useful in building models that understand biomedical texts. An important issue one needs to solve before doing so is linking. Mapping spans in text to entities in a knowledge base, or entity linking, is very much an unsolved problem. Similarly relation extraction, or identifying whether a given span of text provides evidence for the existence of a specific relation that could potentially exist between a pair of entities, is a very challenging research problem. We did not have to deal with these issues completely in Chapter 3, because our entities there were synsets in WordNet, and mapping them to relevant spans of text was relatively straightforward. However, if we have to build similar systems with other knowledge sources, we will have to build joint models for entity linking, relation extraction, and reasoning, which is an exciting research direction.

In terms of incorporating implicit background into NLU systems, the ideas in Chapter 3 form the basis for modeling only a specific kind of commonsense: the eventive kind. More recent work has identified other kinds of commonsense information: Examples include Forbes and Choi (2017), who extract commonsense information related to physical properties of real world objects, and Tandon et al. (2018) who encode commonsense information related to state changes and procedures. In fact, identifying the kinds of commonsense knowledge that is not readily available, encoded in some form, but those that NLU systems need for effective reasoning is a valuable exercise in itself, and could pave way for solving several hard NLU tasks.

8.3 Future work in Knowledge-Aware Reasoning

In Chapters 6 and 7 we focused on a specific subclass of reasoning tasks: those that are grounded in structured contexts, and the reasoning can be expressed as a sequence of discrete operations. There is a lot of ongoing work in tasks where the grounding is in unstructured contexts such as paragraphs and documents (Hill et al., 2015; Seo et al., 2016; Dhingra et al., 2016; Xiong et al., 2016; Yu et al., 2018, among others). To the best of our knowledge, all of that work performs reasoning with continuous operations. That may very well be because all these reasoning systems are built for datasets like (Richardson et al., 2013; Rajpurkar et al., 2016; Joshi et al., 2017, among others) which require selecting spans in passages given as context.

In contrast, the recently introduced DROP dataset (Dua et al., 2019) requires discrete reasoning over paragraphs. The biggest hurdle in taking the route of semantic parsing for this task is the lack of high-quality information extraction. Given that it is already challenging to extract the kinds of information required by the questions above, it would be even more difficult to build a semantic parser that relies on the output of an IE system. One way to deal with this issue is to train joint models for information extraction over paragraphs, and semantic parsing over questions. This opens up a new class of NLP tasks where formalism used for information extraction is determined by the end-task of question answering.

Bibliography

Naoki Abe and Hang Li. Learning word association norms using tree cut pair models. *arXiv preprint cmp-lg/9605029*, 1996. 2.3

Eneko Agirre. Improving parsing and pp attachment performance with sense information. In *ACL*. Citeseer, 2008. 2.4

Jacob Andreas, Andreas Vlachos, and Stephen Clark. Semantic parsing as machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, 2013. 5.2.2

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Learning to compose neural networks for question answering. In *HLT-NAACL*, 2016a. 1.3.2, 5.3.2

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 39–48, 2016b. 5.1

Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. Linear algebraic structure of word senses, with applications to polysemy. *arXiv preprint arXiv:1601.03764*, 2016. 2.2.1

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. Unsupervised neural machine translation. *arXiv preprint arXiv:1710.11041*, 2017. 6.1

Yoav Artzi and Luke Zettlemoyer. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*, 1(1): 49–62, 2013. 5.1, 5.2.2

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014a. 6.1

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014b. 1.1.2, 3.3.2, 5.1

Marco Baroni and Alessandro Lenci. Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics*, 36(4):673–721, 2010. 2.3

David Belanger and Sham M. Kakade. A linear dynamical system model for text. In *ICML*, 2015. 2.2.1

Yonatan Belinkov, Tao Lei, Regina Barzilay, and Amir Globerson. Exploring compositional architectures and word vector representations for prepositional phrase attachment. *Transactions of the Association for Computational Linguistics*, 2:561–572, 2014. 3.4, 3.4.1, 3.4.2, 3.4.3, 3.4.3, 4, 3.4.3, 1, 3.4.3

Jonathan Berant and Percy Liang. Semantic parsing via paraphrasing. In *Proceedings of ACL*, volume 7, page 92, 2014. 6.1

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 2013a. 5.2.2, 5.2.2, 5.3

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, 2013b. 1.3.2, 5.1, 5.3, 5.3.1

Olivier Bodenreider. The unified medical language system (umls): integrating biomedical terminology. *Nucleic acids research*, 32 Database issue:D267–70, 2004. 1.2.1

Antoine Bordes, Jason Weston, Ronan Collobert, Yoshua Bengio, et al. Learning structured embeddings of knowledge bases. In *AAAI*, 2011. 4.5

Johan Bos and Katja Markert. Recognising textual entailment with logical inference. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 628–635. Association for Computational Linguistics, 2005. 1.1.2

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *EMNLP*, 2015. 3.5

Samuel R Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D Manning, and Christopher Potts. A fast unified model for parsing and sentence understanding. *arXiv preprint arXiv:1603.06021*, 2016. 1.1.2, 3.5.3, 3.5.4

Eric Breck, Marc Light, Gideon S Mann, Ellen Riloff, Brianne Brown, Pranav Anand, Mats Rooth, and Michael Thelen. Looking under the hood: Tools for diagnosing your question answering engine. In *Proceedings of the workshop on Open-domain question answering-Volume 12*, pages 1–8. Association for Computational Linguistics, 2001. 1.2.2

Eric Brill and Philip Resnik. A rule-based approach to prepositional phrase attachment disambiguation. In *Proceedings of the 15th conference on Computational linguistics-Volume 2*, pages 1198–1204. Association for Computational Linguistics, 1994. 2.4

Qingqing Cai and Alexander Yates. Large-scale semantic parsing via schema matching and lexicon extension. In *In Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2013. 5.2.2

A. Carlson, C. Cumby, J. Rosen, and D. Roth. The snow learning architecture. (UIUCDCS-R-99-2101), 5 1999. URL `http://cogcomp.cs.illinois.edu/papers/CCRR99.pdf`. 1.3.1

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*, 2013. 3

Danqi Chen and Christopher D Manning. A fast and accurate dependency parser using neural networks. In *EMNLP*, pages 740–750, 2014. 3.2

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. Enhanced lstm for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1657–1668, 2017. 1.1.1

Xinxiong Chen, Zhiyuan Liu, and Maosong Sun. A unified model for word sense representation and disambiguation. In *EMNLP*, pages 1025–1035, 2014. 2.2.1

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*, 2014. 6.1

François Chollet. Keras. `https://github.com/fchollet/keras`, 2015. 4.2.1

Kenneth Ward Church and Patrick Hanks. Word association norms, mutual information, and lexicography. *Computational linguistics*, 16(1):22–29, 1990. 4.4

Massimiliano Ciaramita and Mark Johnson. Explaining away ambiguity: Learning verb selectional preference with bayesian networks. In *Proceedings of the 18th conference on Computational linguistics-Volume 1*, pages 187–193. Association for Computational Linguistics, 2000. 2.3

Peter Clark, Philip Harrison, and Niranjan Balasubramanian. A study of the knowledge base requirements for passing an elementary science test. In *AKBC@CIKM*, 2013. 1.2.1

Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008. 2.1.1, 2.2.3

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537, 2011. 4.3

Courtney Corley and Rada Mihalcea. Measuring the semantic similarity of texts. In *Proceedings of the ACL workshop on empirical modeling of semantic equivalence and entailment*, pages 13–18. Association for Computational Linguistics, 2005. 1.1.2

Deborah A. Dahl, Madeleine Bates, Michael Brown, William Fisher, Kate Hunicke-Smith, David Pallett, Christine Pao, Alexander Rudnicky, and Elizabeth Shriberg. Expanding the scope of the ATIS task: The ATIS-3 corpus. In *Proceedings of the Workshop on Human Language Technology*, 1994. 5.2.2, 6.1

Dipanjan Das and Noah A Smith. Paraphrase identification as probabilistic quasi-synchronous recognition. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 468–476. Association for Computational Linguistics, 2009. 1.1.1

Pradeep Dasigi and Eduard Hovy. Modeling newswire events using neural networks for anomaly detection. In *COLING 2014*, volume 25, pages 1414–1422. Dublin City University and Association for Computational Linguistics, 2014. 4.2.1

Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American society for information* *science*, 41(6):391–407, 1990. 2.1.1

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 2.2.3, 4.3.2

Bhuwan Dhingra, Hanxiao Liu, William W Cohen, and Ruslan Salakhutdinov. Gated-attention readers for text comprehension. *arXiv preprint arXiv:1606.01549*, 2016. 8.3

Li Dong and Mirella Lapata. Language to logical form with neural attention. *CoRR*, abs/1601.01280, 2016a. 1.3.2

Li Dong and Mirella Lapata. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016b. 5.2.2, 6, 6.4.5

Li Dong and Mirella Lapata. Language to logical form with neural attention. In *ACL'16*, 2016c. 7.5

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. 2019. 8.3

Katrin Erk. A simple, similarity-based model for selectional preferences. *ANNUAL MEETING-ASSOCIATION FOR COMPUTATIONAL LINGUISTICS*, 45(1):216, 2007. 2.3

Katrin Erk, Sebastian Padó, and Ulrike Padó. A flexible, corpus-driven model of regular and inverse selectional preferences. *Computational Linguistics*, 36(4):723–763, 2010. 2.3

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard H. Hovy, and Noah A. Smith. Retrofitting word vectors to semantic lexicons. In *NAACL*, 2015. 2.2.2, 2, 3.4.3

Jessica Ficler and Yoav Goldberg. Controlling linguistic style aspects in neural language generation. *arXiv preprint arXiv:1707.02633*, 2017. 6.1

J. Firth. A synopsis of linguistic theory 1930-1955. In *Studies in Linguistic Analysis*. Philological Society, Oxford, 1957. reprinted in Palmer, F. (ed. 1968) Selected Papers of J. R. Firth, Longman, Harlow. 2.1.1

Maxwell Forbes and Yejin Choi. Verb physics: Relative physical knowledge of actions and objects. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 266–276, 2017. 4.5, 8.2

Matt Gardner, Partha Pratim Talukdar, Jayant Krishnamurthy, and Tom Mitchell. Incorporating vector space similarity in random walk inference over knowledge bases. 2014. 4.5

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew E. Peters, Michael Schmitz, and Luke S. Zettlemoyer. Allennlp: A deep semantic natural language processing platform. *CoRR*, abs/1803.07640, 2018. 7.4.2

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010. 7.4.2

Vaibhava Goel and William J Byrne. Minimum bayes-risk automatic speech recognition. *Com-*

puter Speech & Language, 14(2):115–135, 2000. 5.3.3

Omer Goldman, Veronica Latcinnik, Udi Naveh, Amir Globerson, and Jonathan Berant. Weakly-supervised semantic parsing with abstract examples. In *ACL*, 2018. 5.3.1, 7.4.3, 7.5, 7.6

Joshua Goodman. Parsing algorithms and metrics. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 177–183. Association for Computational Linguistics, 1996. 5.3.3

Kelvin Guu, Panupong Pasupat, Evan Zheran Liu, and Percy Liang. From language to programs: Bridging reinforcement learning and maximum marginal likelihood. In *Association for Computational Linguistics (ACL)*, 2017. 5.3.2, 5.3.3, 5.3.4, 7.5

Till Haug, Octavian-Eugen Ganea, and Paulina Grnarova. Neural multi-step reasoning for question answering on semi-structured tables, 2017. 6.4.4

Till Haug, Octavian-Eugen Ganea, and Paulina Grnarova. Neural multi-step reasoning for question answering on semi-structured tables. In *ECIR*, 2018. 7.4.3, 7.5

Marti A Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics-Volume 2*, pages 539–545. Association for Computational Linguistics, 1992. 1.1.1

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701, 2015. 1.1.2

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. The goldilocks principle: Reading children's books with explicit memory representations. *arXiv preprint arXiv:1511.02301*, 2015. 1.2.2, 8.3

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997. 1.4, 4.2, 6.3, 7.4.1

Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard Hovy. Learning whom to trust with mace. In *Proceedings of NAACL-HLT*, pages 1120–1130, 2013. 4.3.1

Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 873–882. Association for Computational Linguistics, 2012. 2.2.1

Drew A Hudson and Christopher D Manning. Compositional attention networks for machine reasoning. *arXiv preprint arXiv:1803.03067*, 2018. 5.1

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. Learning a neural semantic parser from user feedback. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 963–973, 2017. 6.1

Mohit Iyyer, Wen tau Yih, and Ming-Wei Chang. Search-based neural structured learning for sequential question answering. In *Association for Computational Linguistics*, 2017. 5.3.3, 5.3.4

Sujay Kumar Jauhar, Chris Dyer, and Eduard H. Hovy. Ontologically grounded multi-sense representation learning for semantic vector space models. In *NAACL*, 2015. 2.2.1, 2.2.2

Robin Jia and Percy Liang. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016a. 5.2.2, 6, 6.4.5

Robin Jia and Percy Liang. Data recombination for neural semantic parsing. In *ACL'16*, 2016b. 7.5

Richard Johansson and Luis Nieto Pina. Embedding a semantic network in a word space. In *In Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics–Human Language Technologies*. Citeseer, 2015. 2.2.2

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. 2017. 8.3

Rohit J Kate, Yuk Wah Wong, and Raymond J Mooney. Learning to transform natural to formal languages. In *Proceedings of the 20th national conference on Artificial intelligence-Volume 3*, pages 1062–1068. AAAI Press, 2005. 6.1

Jerrold J Katz and Jerry A Fodor. The structure of a semantic theory. *language*, 39(2):170–210, 1963. 1.3.1

Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. Globally coherent text generation with neural checklist models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 329–339, 2016. 7.4.1

Jack Kiefer, Jacob Wolfowitz, et al. Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, 23(3):462–466, 1952. 7.4.2

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 3.5.3, 4.2.1, 7.4.2

Karin Kipper, Anna Korhonen, Neville Ryant, and Martha Palmer. A large-scale classification of english verbs. *Language Resources and Evaluation*, 42(1):21–40, 2008. 4

Klaus Krippendorff. *Content analysis: An introduction to its methodology*. Sage Publications (Beverly Hills), 1980. 4.3.1

Saisuresh Krishnakumaran and Xiaojin Zhu. Hunting elusive metaphors using lexical resources. In *Proceedings of the Workshop on Computational approaches to Figurative Language*, pages 13–20. Association for Computational Linguistics, 2007. 2.3

Jayant Krishnamurthy and Tom M. Mitchell. Weakly supervised training of semantic parsers. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 2012. 5.1, 5.2.2

Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. Neural semantic parsing with type constraints for semi-structured tables. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1516–1526, 2017. 1.3.2, 5.1, 7.4.1, 7.4.2, 7.5

Yuval Krymolowski and Dan Roth. Incorporating knowledge in natural language learning: A case study. *Urbana*, 51:61801. 1.3.1

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. Learning to automatically solve algebra word problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 271–281, Baltimore, Maryland, June 2014. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P14-1026`. 5.1

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. Lexical generalization in CCG grammar induction for semantic parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2011. 5.2.2, 6.1, 7.5

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 2013. 5.2.2

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. In *NAACL*, 2016. 3.2

Tao Lei, Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. Low-rank tensors for scoring dependency structures. In *ACL*. Association for Computational Linguistics, 2014. 3.4.3

Alessandro Lenci. Composing and updating verb argument expectations: A distributional semantic model. In *Proceedings of the 2nd Workshop on Cognitive Modeling and Computational Linguistics*, pages 58–66. Association for Computational Linguistics, 2011. 2.3

Hector J Levesque, Ernest Davis, and Leora Morgenstern. The winograd schema challenge. In *Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning*, pages 552–561. AAAI Press, 2012. 1.1.1

Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Dan Jurafsky. Adversarial learning for neural dialogue generation. *arXiv preprint arXiv:1701.06547*, 2017. 6.1

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D. Forbus, and Ni Lao. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. *CoRR*, abs/1611.00020, 2016. 1.3.2, 5.3.2

Chen Liang, Mohammad Norouzi, Jonathan Berant, Quoc Le, and Ni Lao. Memory augmented policy optimization for program synthesis with generalization. *arXiv preprint arXiv:1807.02322*, 2018. (document), 5.3.2, 5.3.2, 7.3.1, 7.4.3, 7.4, 7.5

Percy Liang. Learning executable semantic parsers for natural language understanding. *Communications of the ACM*, 59(9):68–76, 2016. 5.1

Percy Liang, Michael I. Jordan, and Dan Klein. Learning dependency-based compositional semantics. In *ACL*, 2011a. 6.4.2

Percy Liang, Michael I Jordan, and Dan Klein. Learning dependency-based compositional semantics. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, 2011b. 5.2.2, 5.3, 5.3.1, 6.1

Wang Ling, Phil Blunsom, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský, Fumin Wang, and Andrew Senior. Latent predictor networks for code generation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long*

Papers), 2016. 5.2.2

Nicholas Locascio, Karthik Narasimhan, Eduardo De Leon, Nate Kushman, and Regina Barzilay. Neural generation of regular expressions from natural language with minimal domain knowledge. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016. 5.2.2

Kevin Lund and Curt Burgess. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior research methods, instruments, & computers*, 28(2):203–208, 1996. 2.1.1

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *Proceedings of the International Conference on Learning Representations*, 2013a. 2.1.1

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546, 2013b. 2.2.2

George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11): 39–41, 1995. 1.2.1, 2.2.3, 3.1, 3.2

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics, 2009. 4.5

Dipendra Misra, Ming-Wei Chang, Xiaodong He, and Wen tau Yih. Policy shaping and generalized update equations for semantic parsing from denotations. In *EMNLP*, 2018. 7.5

Dan I Moldovan and Vasile Rus. Logic form transformation of wordnet and its applicability to question answering. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 402–409. Association for Computational Linguistics, 2001. 1.1.2, 1.3.1

Lili Mou, Men Rui, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. Recognizing entailment and contradiction by tree-based convolution. In *Proc. of ACL*, 2016. 3.5.3

Ramesh Nallapati, Bowen Zhou, Cícero Nogueira dos Santos, Çaglar Gülçehre, and Bing Xiang. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *CoNLL*, 2016. 6.1

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. Efficient non-parametric estimation of multiple embeddings per word in vector space. *arXiv preprint arXiv:1504.06654*, 2015. 2.2.1

Arvind Neelakantan, Quoc V. Le, Martín Abadi, Andrew McCallum, and Dario Amodei. Learning a natural language interface with neural programmer. *CoRR*, abs/1611.08945, 2016. 1.3.2, 6.4.4

Arvind Neelakantan, Quoc V Le, Martin Abadi, Andrew McCallum, and Dario Amodei. Learning a natural language interface with neural programmer. In *ICLR*, 2017. 7.4.3, 7.5

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 809–816, 2011. 4.5

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(02):95–135, 2007. 4.4

Martha Palmer, Daniel Gildea, and Paul Kingsbury. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106, 2005. 4.2

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics, 2002. 1.1.2

Panupong Pasupat and Percy Liang. Compositional semantic parsing on semi-structured tables. *arXiv preprint arXiv:1508.00305*, 2015. (document), 1.2.2, 1.3.2, 5.2.2, 5.3, 6.1, 6.1, 6.4, 6.4.1, 6.4.2, 6.4.2, 6.4.4, 7, 7.4.3

Panupong Pasupat and Percy Liang. Inferring logical forms from denotations. *arXiv preprint arXiv:1606.06900*, 2016. 5.3, 5.3.1, 6.2, 6.3, 6.4.2, 6.4.2, 7.6

Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization. *CoRR*, abs/1705.04304, 2017. 6.1

Anselmo Peñas, Eduard Hovy, Pamela Forner, Álvaro Rodrigo, Richard Sutcliffe, and Roser Morante. Qa4mre 2011-2013: Overview of question answering for machine reading evaluation. In *International Conference of the Cross-Language Evaluation Forum for European Languages*, pages 303–320. Springer, 2013. 1.2.2

Nanyun Peng, Hoifung Poon, Chris Quirk, Kristina Toutanova, and Wen-tau Yih. Cross-sentence n-ary relation extraction with graph lstms. *arXiv preprint arXiv:1708.03743*, 2017. 1.1.1

Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, 2014. 3.4.3

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proc. of NAACL*, 2018. 2.2.3, 3

Benjamin C. Pierce. Types and programming languages. 2002. 6.4.1

Maxim Rabinovich, Mitchell Stern, and Dan Klein. Abstract syntax networks for code generation and semantic parsing. In *The 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2017a. 5.2.2, 6.1

Maxim Rabinovich, Mitchell Stern, and Dan Klein. Abstract syntax networks for code generation and semantic parsing. In *ACL*, 2017b. 7.5

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2.2.3

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *EMNLP*, 2016. 5.1, 8.3

Adwait Ratnaparkhi, Jeff Reynar, and Salim Roukos. A maximum entropy model for prepositional phrase attachment. In *Proceedings of the workshop on Human Language Technology*, 1994. 3.4.3

Joseph Reisinger and Raymond J Mooney. Multi-prototype vector-space models of word meaning. In *HLT-ACL*, 2010. 2.2.1

Philip Resnik. Semantic classes and syntactic ambiguity. In *Proceedings of the workshop on Human Language Technology*. Association for Computational Linguistics, 1993. 2.4, 3.3.1, 3.4

Philip Resnik. Selectional constraints: An information-theoretic model and its computational realization. *Cognition*, 61(1):127–159, 1996. 2.3

Philip Resnik. Selectional preference and sense disambiguation. In *Proceedings of the ACL SIGLEX Workshop on Tagging Text with Lexical Semantics: Why, What, and How*, pages 52–57. Washington, DC, 1997. 1.3.1, 2.3

Matthew Richardson, Christopher JC Burges, and Erin Renshaw. Mctest: A challenge dataset for the open-domain machine comprehension of text. 1.2.2

Matthew Richardson, Christopher J. C. Burges, and Erin Renshaw. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *EMNLP*, 2013. 8.3

Mats Rooth, Stefan Riezler, Detlef Prescher, Glenn Carroll, and Franz Beil. Inducing a semantically annotated lexicon via em-based clustering. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 104–111. Association for Computational Linguistics, 1999. 2.3

Sascha Rothe and Hinrich Schütze. Autoextend: Extending word embeddings to embeddings for synsets and lexemes. In *ACL*, 2015. 2.2.1, 3.3.2, 3.4.3, 3.4.3

Diarmuid O Séaghdha. Latent variable models of selectional preference. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 435–444. Association for Computational Linguistics, 2010. 2.3

Abigail See, Peter J Liu, and Christopher D Manning. Get to the point: Summarization with pointer-generator networks. *ACL*, 2017. 7.5

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *CoRR*, abs/1611.01603, 2016. 5.1, 8.3

Ekaterina Shutova, Simone Teufel, and Anna Korhonen. Statistical metaphor processing. *Computational Linguistics*, 39(2):301–353, 2013. 1.3.1, 2.3

David A Smith and Jason Eisner. Minimum risk annealing for training log-linear models. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 787–794. Association for Computational Linguistics, 2006. 5.3.3

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. Reasoning with neural tensor networks for knowledge base completion. In *Advances in neural information processing systems*, pages 926–934, 2013. 4.5

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014. 7.4.2

Alane Suhr, Mike Lewis, James Yeh, and Yoav Artzi. A corpus of natural language for visual reasoning. 2017. 1.3.2, 5.3, 7, 7.4.3

Alane Suhr, Srinivasan Iyer, and Yoav Artzi. Learning to map context-dependent sentences to executable formal queries. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 2238–2249, 2018. 6.1

Ilya Sutskever, Joshua B Tenenbaum, and Ruslan R Salakhutdinov. Modelling relational data using bayesian clustered tensor factorization. In *Advances in neural information processing systems*, pages 1821–1828, 2009. 4.5

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014. 6.1

Hao Tan and Mohit Bansal. Object ordering with bidirectional matchings for visual reasoning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2018. 7.4.3, 7.5

Niket Tandon, Bhavana Dalvi, Joel Grus, Wen tau Yih, Antoine Bosselut, and Peter Clark. Reasoning about actions and state changes by injecting commonsense knowledge. In *EMNLP*, 2018. 4.5, 8.2

Marta Tatu and Dan Moldovan. A semantic approach to recognizing textual entailment. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 371–378. Association for Computational Linguistics, 2005. 1.1.2

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics, 2003a. 1.1.1

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *NAACL*, 2003b. 3.5.1

Andrew Trask, Felix Hill, Scott Reed, Jack Rae, Chris Dyer, and Phil Blunsom. Neural arithmetic logic units. *arXiv preprint arXiv:1808.00508*, 2018. 5.1

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. Modeling coverage for neural machine translation. *ACL*, 2016. 7.5

Tim Van de Cruys. A non-negative tensor factorization model for selectional preference induction. *GEMS: GEometrical Models of Natural Language Semantics*, page 83, 2009. 2.3

Tim Van de Cruys. A neural network approach to selectional preference acquisition. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 26–35, 2014. 2.3

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017. 2.2.3

Luke Vilnis and Andrew McCallum. Word representations via gaussian embedding. In *ICLR*, 2015. 2.2.1

Peilu Wang, Yao Qian, Frank K Soong, Lei He, and Hai Zhao. Part-of-speech tagging with bidirectional long short-term memory recurrent neural network. *arXiv preprint arXiv:1510.06168*, 2015a. 1.1.1

Yushi Wang, Jonathan Berant, and Percy Liang. Building a semantic parser overnight. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2015b. 6.1

Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. *arXiv preprint arXiv:1410.3916*, 2014. 1.1.2

Yorick Wilks. Preference semantics. Technical report, STANFORD UNIV CA DEPT OF COMPUTER SCIENCE, 1973. 2.3, 4

Yorick Wilks. A preferential, pattern-seeking, semantics for natural language inference. *Artificial intelligence*, 6(1):53–74, 1975. 1.3.1

Yorick Wilks. Making preferences more active. In *Words and Intelligence I*, pages 141–166. Springer, 2007a. 2.3

Yorick Wilks. A preferential, pattern-seeking, semantics for natural language inference. In *Words and Intelligence I*, pages 83–102. Springer, 2007b. 2.3

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992. 6.4.2

Sam Wiseman and Alexander M Rush. Sequence-to-sequence learning as beam-search optimization. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1296–1306, 2016. 6.1

Yuk Wah Wong and Raymond J. Mooney. Learning for semantic parsing with statistical machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL*, 2006. 5.2.2

Yuk Wah Wong and Raymond J. Mooney. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, 2007. 5.2.2

Chunyang Xiao, Marc Dymetman, and Claire Gardent. Sequence-based structured prediction for semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1341–1350, 2016. 1.3.2

Caiming Xiong, Stephen Merity, and Richard Socher. Dynamic memory networks for visual and textual question answering. In *ICML*, 2016. 1.1.2, 8.3

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016. 1.1.2

Scott Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the Joint Conference of the 53rd Annual Meeting of the ACL and the 7th International Joint Conference on Natural Language Processing of the AFNLP*, 2015. 5.2.2, 6

Wen-tau Yih, Matthew Richardson, Christopher Meek, Ming-Wei Chang, and Jina Suh. The value of semantic parse labeling for knowledge base question answering. In *ACL*, 2016a. 5.3

Wen-tau Yih, Matthew Richardson, Christopher Meek, Ming-Wei Chang, and Jina Suh. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, 2016b. 5.3

Pengcheng Yin and Graham Neubig. A syntactic neural model for general-purpose code generation. In *ACL'17*, 2017a. 7.5

Pengcheng Yin and Graham Neubig. A syntactic neural model for general-purpose code generation. In *The 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2017b. 5.2.2, 6.1

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. Qanet: Combining local convolution with global self-attention for reading comprehension. *arXiv preprint arXiv:1804.09541*, 2018. 5.1, 8.3

Mo Yu and Mark Dredze. Improving lexical embeddings with semantic knowledge. In *ACL*, 2014. 2.2.2

Benat Zapirain, Eneko Agirre, Lluis Marquez, and Mihai Surdeanu. Selectional preferences for semantic role classification. *Computational Linguistics*, 2013. 2.4

John M. Zelle and Raymond J. Mooney. Learning to parse database queries using inductive logic programming. In *AAAI/IAAI, Vol. 2*, 1996a. 1.3.2, 6.1

John M. Zelle and Raymond J. Mooney. Learning to parse database queries using inductive logic programming. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, 1996b. 5.2.2, 5.3

John M Zelle and Raymond J Mooney. Learning to parse database queries using inductive logic programming. In *Proceedings of the thirteenth national conference on Artificial intelligence-Volume 2*, pages 1050–1055. AAAI Press, 1996c. 5.3, 7.3.1

Luke S. Zettlemoyer and Michael Collins. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI*, 2005a. 1.1.2, 1.3.2, 5.3, 6.1

Luke S. Zettlemoyer and Michael Collins. Learning to map sentences to logical form: structured classification with probabilistic categorial grammars. In *UAI '05, Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence*, 2005b. 5.2.2, 5.3

Luke S Zettlemoyer and Michael Collins. Online learning of relaxed ccg grammars for parsing to logical form. In *EMNLP-CoNLL*, pages 678–687, 2007. 1.3.2, 5.2.2, 6.1

Yuchen Zhang, Panupong Pasupat, and Percy Liang. Macro grammars and holistic triggering for efficient semantic parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1214–1223, 2017. 7.4.3

Kai Zhao and Liang Huang. Type-driven incremental semantic parsing with polymorphism. In *HLT-NAACL*, 2015. 5.2.2