April 13, 2014. Submitted to the Annual Review of Linguistics. Comments welcome.
Minor corrections April 30, 2014 and May 18, 2014.

# Bringing machine learning and compositional semantics together

Percy Liang and Christopher Potts

**Abstract**

Computational semantics has long been seen as a field divided between logical and statistical approaches, but this divide is rapidly eroding, with the development of statistical models that learn compositional semantic theories from corpora and databases. This paper presents a simple discriminative learning framework for defining such models and relating them to logical theories. Within this framework, we discuss the task of learning to map utterances to logical forms (semantic parsing) and the task of learning from denotations with logical forms as latent variables. We also consider models that use distributed (e.g., vector) representations rather than logical ones, showing that these can be seen as part of the same overall framework for understanding meaning and structural complexity.

**Keywords** compositionality, logical forms, distributed representations, semantic parsing, discriminative learning, recursive neural networks

## 1 Introduction

Computational semantics often seems like a field divided by methodologies and near-term goals (Cooper 2012). **Logical** approaches rely on techniques from proof theory and model-theoretic semantics, they have strong ties to linguistic semantics, and they are concerned primarily with inference, ambiguity, vagueness, and compositional interpretation of full syntactic parses (Blackburn & Bos 2003, 2005; van Eijck & Unger 2010). In contrast, **statistical** approaches derive their tools from algorithms and optimization, and they tend to focus on word meanings and broad notions of semantic content (Landauer et al. 2007; Turney & Pantel 2010). The two types of approaches share the long-term vision of achieving deep natural language understanding, but their day-to-day differences can make them seem unrelated and even incompatible.

With the present paper, we seek to show that the distinction between logical and statistical approaches is rapidly disappearing, with the development of models that can learn the conventional aspects of natural language meaning from corpora and databases. These models interpret rich linguistic representations in a compositional fashion, and they offer novel perspectives on foundational issues like ambiguity, inference, and grounding. The fundamental question for

these approaches is what kinds of data and models are needed for effective learning. Addressing this question is a prerequisite for implementing robust systems for natural language understanding, and the answers can inform psychological models of language acquisition and language processing.

The leading players in our discussion are compositionality and machine learning. After describing our view of linguistic objects (section 2), we introduce these two players (section 3). Although they come from different scientific worlds, we show that they are deeply united around the concepts of generalization, meaning, and structural complexity. The bulk of the paper is devoted to showing how learning-based theories of semantics bring the two worlds together. Specifically, compositionality characterizes the recursive nature of the linguistic ability required to generalize to a creative capacity, and learning details the conditions under which such an ability can be acquired from data. We substantiate this connection first for models in which the semantic representations are logical forms (section 4) and then for models in which the semantic representations are distributed (e.g., vectors; section 5). Historically, distributional approaches have been more closely associated with learning, but we show, building on much previous literature, that both types of representations can be learned.

Our focus is on learning general theories of semantics, so we develop the ideas using formal tools that are familiar in linguistics, computer science, and engineering, and that are relatively straightforward to present in limited space: context-free grammars, simple logical representations, linear models, and first-order optimization algorithms. This focus means that we largely neglect many important, relevant developments in semantic representation (de Marneffe et al. 2006; MacCartney & Manning 2009; van Eijck & Unger 2010; Palmer et al. 2010), semantic interpretation (Dagan et al. 2006; Saurí & Pustejovsky 2009), and structured prediction (Baklr et al. 2010; Smith 2011). It's our hope, though, that our discussion suggests new perspectives on these efforts. (For more general introductions to data-driven approaches to computational semantics, see Ng & Zelle 1997; Jurafsky & Martin 2009: §IV.)

## 2 Linguistic objects

We model linguistic objects as triples $\langle u, s, d\rangle$, where $u$ is an **utterance**, $s$ is a **semantic representation**, and $d$ is the **denotation** of $s$. We use $\ulcorner u \urcorner$ for the translation of syntactic expression $u$ into its semantic representation, and we use $[\![s]\!]$ for the denotation of semantic representation $s$. The goal of the next few subsections is to more fully articulate what these objects are like and how they relate to each other.

Table 1 describes a simple interpreted grammar for these triples $\langle u, s, d\rangle$, and table 2 provides some examples. This grammar provides a (very limited) theory of basic English arithmetic descriptions like "two times minus one". Though the grammar captures an unusual fragment of language, it is an excellent vehicle for defining semantic theories and exploring learning-based perspectives on them, so we will rely on it for concrete illustrations throughout this paper.

| Syntax | Semantic representation | Denotation |
|---|---|---|
| N → one | `1` | 1 |
| N → two | `2` | 2 |
| ⋮ | ⋮ | ⋮ |
| R → plus | $+$ | the $R$ such that $R(x, y) = x + y$ |
| R → minus | $-$ | the $R$ such that $R(x, y) = x - y$ |
| R → times | $\times$ | the $R$ such that $R(x, y) = x * y$ |
| S → minus | $\neg$ | the $f$ such that $f(x) = -x$ |
| N → S N | $\ulcorner \mathrm{S} \urcorner \ulcorner \mathrm{N} \urcorner$ | $[\![\ulcorner \mathrm{S} \urcorner]\!]([\![\ulcorner \mathrm{N} \urcorner]\!])$ |
| N → $\mathrm{N}_L$ R $\mathrm{N}_R$ | $(\ulcorner \mathrm{R} \urcorner \ulcorner \mathrm{N}_L \urcorner \ulcorner \mathrm{N}_R \urcorner)$ | $[\![\ulcorner \mathrm{R} \urcorner]\!]([\![\ulcorner \mathrm{N}_L \urcorner]\!], [\![\ulcorner \mathrm{N}_R \urcorner]\!])$ |

Table 1: An illustrative grammar. $\ulcorner u \urcorner$ is the translation of syntactic expression $u$, and $[\![s]\!]$ is the denotation of semantic representation $s$. N is the CFG's start symbol. In the final rule, the $L$ and $R$ subscripts are meta-annotations to ensure deterministic translation and interpretation.

| Utterance | Semantic representation | Denotation |
|---|---|---|
| A. seven minus five | `(− 7 5)` | 2 |
| B. minus three plus one | `(+ ¬3 1)` | −2 |
| C. two minus two times two | `(× (− 2 2) 2)` | 0 |
| D. two plus three plus four | `(+ 2 (+ 3 4))` | 9 |

Table 2: Examples derived from the grammar in table 1.

## 2.1 Utterances

We model each utterance as a sequence of strings (words). These can be thought of as derived from the output of the context-free grammar (CFG) given in the left column of table 1. This unstructured starting point helps keep the focus on semantics. However, relatively little hinges on this choice; for example, while using syntactic trees, dependency graphs, or shallow parses would affect the precise mapping to semantic representations and on to denotations, the substantive connections with the models we discuss below would remain the same.

## 2.2 Semantic representations

In linguistics, semantic representations are generally **logical forms**: expressions in a fully specified, unambiguous artificial language. The grammar in table 1 adopts such a view, defining semantic representations with a logical language that has constant symbols for numbers and relations and uses juxtaposition and bracketing to create complex expressions. In the literature, one encounters a variety of different formalisms — for example, lambda calculi (Carpenter 1997)

or first-order fragments thereof (Bird et al. 2009), natural logics (MacCartney & Manning 2009; Moss 2009), diagrammatic languages (Kamp & Reyle 1993), programming languages (Blackburn & Bos 2005), robot controller languages (Matuszek et al. 2012b), and database query languages (Zelle & Mooney 1996).

A given utterance might be consistent with multiple logical forms in our grammar, creating **ambiguity**. For instance, the utterance in line B of table 2 also maps to the logical form $\neg$`(+ 3 1)`, which denotes $-4$. Intuitively, this happens if "minus" is parsed as taking scope over the addition expression to its right. Similarly, utterance C can be construed with "two times two" as a unit, leading to the logical form `(− 2 (× 2 2))`, which denotes $-2$. Utterance D also has an alternative analysis as `(+ (+ 2 3) 4)`, but this ambiguity is spurious in the sense that it has the same denotation as the one in table 1. Our grammar also has one lexical ambiguity — "minus" can pick out a unary or binary relation — but this is immediately resolved in complex structures.

In semantic theories aligned with generative syntax (see Heim & Kratzer 1998), the logical form is an abstract syntactic object that might differ significantly from the surface syntax, especially with regard to the relative scope of semantic operators (Reinhart 1997; Szabolcsi 2009). For example, the phrase "All that glitters is not gold" has two readings. The **surface-scope** reading has logical form $\forall x\,((\texttt{glitter}\,x) \rightarrow \neg(\texttt{gold}\,x))$, which says that no glittering thing is gold. The **inverse-scope** reading reverses the order of negation and the universal quantifier, leading to logical form $\neg\forall x\,((\texttt{glitter}\,x) \rightarrow (\texttt{gold}\,x))$, which makes the weaker statement that not all glittering things are gold. In this case, world knowledge and other pragmatic factors favor the inverse-scope reading, though general interpretive preferences seem to broadly favor surface scope. Our grammar doesn't have the sort of operators that could create these ambiguities, so we set them aside; for discussion of how to characterize and learn scopal preferences from data, see Higgins & Sadock 2003; AnderBois et al. 2012; Liang et al. 2013.

The term 'logical form' is often used as a synonym for 'semantic representation', but we keep them separate. In section 5, we discuss theories in which the semantic representations can be vectors and matrices. These are not obviously 'logical', but they can encode information that is expressed directly by logical forms.

## 2.3 Denotations

Like many scientific fields, semantics does not provide a definitive verdict on the true nature of the objects it is investigating. Most theories follow Lewis's (1970: 22) dictum: "In order to say what a meaning *is*, we may first ask what a meaning *does*, and then find something that does that". For the most part, since Montague (1970), linguists have relied on higher-order functional models to provide denotations for their semantic representations. In an idealized sense, these structures permit a rigorous reconstruction of core semantic concepts like truth and entailment, thereby behaving in some sense like meanings behave, though they immediately raise questions about computational complexity and

mental representation (Partee 1980, 1981; Jackendoff 1992, 1997).

The computational models discussed below are similarly flexible about the nature of denotations. They can come from a database, a corpus, a physical environment, or a cognitive model — whatever is appropriate for the task at hand. For example, the grammar in table 1 says that denotations are numbers, but one can take other perspectives, perhaps viewing logical forms as programs and denotations as the result of executing those programs. In that case, the denotations might be events in the physical world, symbolic representations in a lower-level language, or even symbolic representations in the same language used for logical forms, thereby blurring the distinction between representation and denotation (Katz 1972, 1996; Kripke 1975; Chierchia & Turner 1988).

We assume that each logical form has a unique denotation, so that the interpretation function $[\![\cdot]\!]$ is truly a function from logical forms to denotations. Vagueness complicates this assumption (Kennedy 2011). In the presence of vagueness, a given semantic representation $s$ can be compatible with multiple (perhaps infinitely many) denotations. Vagueness exists in all aspects of the lexicon and projects in semantic composition, creating vague meanings for complex phrases. It is the rule in natural language, not the exception, and it is arguably crucial for the flexible, expressive nature of such languages, allowing fixed expressions to make different distinctions in different contexts and helping people to communicate under uncertainty about the world (Kamp & Partee 1995; Graff 2000). We do not attempt to model any aspects of vagueness in this paper, though we acknowledge that it is a central concern in linguistic meaning and a significant challenge for learning.

# 3 Themes

Our two players, compositionality and learning, were born to different fields and have historically been driven by divergent goals. We now review these two concepts, keeping to a standard treatment, but also noting their complementary perspective on semantics. These perspectives are synthesized in section 4.

## 3.1 Compositionality

The principle of compositionality states that the meaning of a complex syntactic phrase is a function of the meanings of its parts and their mode of combination (Katz & Fodor 1963; Montague 1974; Partee 1984; Janssen 1997; Werning et al. 2012). In this general form, the principle has little power to force theoretical decisions, since it grants near total freedom concerning the syntax, the meanings, and the modes of semantic combination, and so attempts have been made to formalize it precisely and substantively (for discussion, see Janssen 1997; Dowty 2007; Szabó 2012). However, even given informally, the principle provides valuable guidance. And, indeed, compositionality is arguably the central principle of linguistic semantics, shaping all discussions about lexical meaning, the relationship between syntax and semantics, and other foundational concepts.

Intuitively, compositionality outlines a recursive interpretation process in which the lexical items are listed as base cases and the recursive clauses define the modes of combination. Current theories assume that the modes of combination are few and highly general, which places essentially all of the complexity in the lexicon (Klein & Sag 1985). This design is evident in our grammar (table 1): the lexical items take up most of the space and encode, in their logical forms and denotations, the ways in which they can combine with other terms. The final two lines of the grammar define the modes of combination, both of which amount to functional application between denotations. Wider-coverage grammars contain lexical meanings that are more complex, and they might also postulate additional modes of combination, but the guiding ideas are the same.

Compositionality is central to characterizing the ways in which small changes to a syntactic structure can yield profound changes in meaning. For instance, "two minus three" and "three minus two" contain the same words but lead to different denotations. The grammar gives a precise account of how this interpretive consequence follows from the syntax of the two utterances. Similarly, we saw that "minus three plus one" (table 2, line B) is consistent with two parses. Where the immediate parts in the syntax are "minus" and "three plus one", the subconstituent `(+ 3 1)` resolves to the denotation 4, and $[\![\neg]\!]$ takes this as an argument to produce $[\![\neg]\!](4) = -4$. In contrast, where the immediate parts are "minus three", "plus", and "one", $[\![\neg]\!]$ operates on 3, which affects the rest of the computation in predictable ways to deliver the denotation $-2$. This is just a glimpse of the subtlety of natural language, where the superficially least contentful words ("every", "no", "not", "might", "but", etc.) often drive the most dramatic effects depending on where they appear in the constituent structure, which determines what their corresponding semantic arguments are.

Compositionality is often linked to our ability to produce and interpret novel utterances. While it is too strong to say that compositionality is necessary or sufficient for this kind of creative ability, it does help characterize it: once one has acquired the syntax of the language, memorized all the lexical meanings, and mastered the few modes of composition, one can interpret novel combinations of them. This abstract capacity is at the heart of what computational models of semantics would like to learn, so that they too can efficiently interpret novel combinations of words and phrases.

## 3.2 Learning

In statistics and artificial intelligence, machine learning concerns the ability to generalize from a set of past observations or experiences in a way that leads to improved performance on a future task (Mitchell 1997: §1). For example, an artificial agent might be said to learn (even master) the game of checkers by repeatedly playing games in which it chooses moves, observes the effects of those choices on the game's outcome, and adjusts its future choices in a way that favors wins (Samuel 1959, 1967). Or an email server might learn to distinguish genuine email from spam by observing lots of messages of each kind in order to identify criteria for classifying them along this dimension.

| | | Feature representations $\phi(x, y)$ | | |
|---|---|---|---|---|
| | $(x, y)$ | 'empty string' | 'last word' | 'all words' |
| Train | (twenty five, O) | $\epsilon$ | five | [twenty, five] |
| | (thirty one, O) | $\epsilon$ | one ~~eight~~ | [thirty, one] |
| | (forty nine, O) | $\epsilon$ | nine | [forty, nine] |
| | (fifty two, E) | $\epsilon$ | two | [fifty, two] |
| | (eighty two, E) | $\epsilon$ | two | [eighty, two] |
| | (eighty four, E) | $\epsilon$ | four | [eighty, four] |
| | (eighty six, E) | $\epsilon$ | six | [eighty, six] |
| Test | (eighty five, O) | $\epsilon \rightarrow$ E | five $\rightarrow$ O | [eighty, five] $\rightarrow$ E |

Table 3: Tradeoffs in machine learning. If our representation is too coarse ('empty string'), we ignore useful information. If our representation is too detailed ('all words'), we risk overfitting to the training data, especially where there is not much of it. The key is to strike a good balance, as in 'last word'.

We focus on **supervised learning**, where the experiences take the form of pairs $(x, y)$ called **training examples**. Here, $x \in \mathcal{X}$ is the system's input and $y \in \mathcal{Y}$ is the desired output. In computational linguistics, $x$ could be a sentence and $y$ the syntactic parse of $x$; or $x$ could be a Japanese sentence and $y$ its Swahili translation; or $x$ could be a description and $y$ its referent in the world; and so forth. Such systems are evaluated on their ability to generalize. Specifically, given a new (possibly unseen) test example $(x, y)$, the system observes $x$ and makes a prediction $\hat{y}(x)$. This incurs a cost $c(y, \hat{y}(x))$, which is typically 0 if the prediction is correct ($\hat{y}(x) = y$) and 1 otherwise ($\hat{y}(x) \neq y$).

A machine learning system has three integral pieces: (i) a feature representation of the data, (ii) an objective function (which usually corresponds to minimizing error on training examples), and (iii) an algorithm for optimizing the objective function. For concreteness, we now develop each of these pieces within the paradigm of linear classification, which is the bread and butter of statistical natural language processing (Manning & Schütze 1999; Smith 2011).

Arguably the most important part of machine learning is determining a suitable representation of the data. The key idea in linear classification is to map each input–output pair $(x, y)$ to a $d$-dimensional **feature vector** $\phi(x, y) \in \mathbb{R}^d$, where each coordinate $\phi(x, y)_i$ represents something about $y$ or its relationship to $x$. The illustrative example in table 3 helps to convey why features are crucial. The task is to learn whether a string describing a number (e.g., "thirty five") denotes an odd or even number ($\mathcal{Y} = \{\text{E}, \text{O}\}$). The 'empty string' column represents one extreme end of the spectrum, where we trivially represent all inputs as the same. In this case, the learning system can do no better than guessing the majority class label in its training examples, thereby ignoring a lot of useful information. Near the other end of the spectrum is the 'all words' representation, which splits its input into words and uses all of them as features.

This representation does poorly because it is misled by its small set of training examples, in which "eighty" occurs only in `E` examples (although with a larger training set, this representation would be fine). In the happier middle for our problem and available data lies the 'last word' model, which captures the essence of the task and so is able to perform perfectly. Of course, in more realistic settings, finding the right representation is more challenging, requiring machine learning expertise, domain expertise, and gumption (Domingos 2012).

The next piece involves defining the learning goal in terms of an **objective function** and associated **scoring function**. For scoring, we associate each feature $j$ with a real-valued weight $w_j$, intuitively representing how informative (positive or negative) feature $j$ is. The scoring function itself then assigns a value to any $(x, y)$ pair computing a weighted sum over the features — specifically, the inner product of $\phi(x, y)$ and $\mathbf{w}$:

$$\text{Score}_{\mathbf{w}}(x, y) = \mathbf{w} \cdot \phi(x, y) = \sum_{j=1}^{d} w_j \phi(x, y)_j. \tag{1}$$

With this scoring function, given any new input $x$, we simply predict the output with the highest score: $\hat{y}(x) = \arg\max_{y \in \mathcal{Y}} \text{Score}_{\mathbf{w}}(x, y)$.

The goal of training is to take a set of examples $\mathcal{D}$ consisting of input–output pairs $(x, y)$ and optimize the weights $\mathbf{w}$ so that the score of the correct output $y$ is generally larger than the score of incorrect outputs $y' \in \mathcal{Y} \backslash \{y\}$. There are many suitable objective functions for the problem we address here. For concreteness, we use the multiclass hinge loss objective (Taskar et al. 2003):

$$\min_{\mathbf{w} \in \mathbb{R}^d} \sum_{(x,y) \in \mathcal{D}} \max_{y' \in \mathcal{Y}} [\text{Score}_{\mathbf{w}}(x, y') + c(y, y')] - \text{Score}_{\mathbf{w}}(x, y), \tag{2}$$

where $\mathcal{D}$ is a set of $(x, y)$ training examples and $c(y, y')$ is the cost for predicting $y'$ when the correct output is $y$. Usually, $c(y, y') = 0$ if $y = y'$, and 1 otherwise. In words, we are trying to find the best weight vector $\mathbf{w}$ that minimizes the cumulative loss over all training examples $\mathcal{D}$. To understand the loss on a single $(x, y)$ pair, first note that the loss is zero when the score of the correct output ($\text{Score}_{\mathbf{w}}(x, y)$) exceeds that of any incorrect output ($\text{Score}_{\mathbf{w}}(x, y')$ for $y' \neq y$) by at least $c(y, y') = 1$. Otherwise, we pay linearly in the amount by which the former falls short of the latter.

The final piece of learning is optimization (Boyd & Vandenberghe 2004). There are many ways of solving the optimization problem defined by our objective function (2). One simple method is **stochastic gradient descent** (SGD), which iteratively computes the (sub)gradient of a single term of the optimization problem. The full algorithm is stated in figure 1 in terms of our feature representations, scoring function, and objective function. For each training example $(x, y) \in \mathcal{D}$, SGD make a prediction for it (line 4), giving a boost of $c(y, y')$ to incorrect $y'$. Where the prediction is correct (i.e., $\tilde{y} = y$), the update does nothing because the difference in line 5 is 0. Where the prediction is incorrect, the update adjusts the weights in the direction of the correct answer $\phi(x, y)$ and

away from the incorrectly predicted answer $\phi(x, \tilde{y})$. The parameter $\eta$ controls the size of these adjustments. SGD and its variants are widely used in practice because it is guaranteed to converge (under some technical conditions), is straightforward to implement, and scales to large datasets (Bottou 2010).

StochasticGradientDescent($\mathcal{D}, T, \eta$)
  $\mathcal{D}$: a set of training examples $(x, y) \in (\mathcal{X} \times \mathcal{Y})$
  $T$: the number of passes to make through the data
  $\eta > 0$: learning rate (e.g., $\frac{1}{\sqrt{T}}$)
1 Initialize $\mathbf{w} \leftarrow \mathbf{0}$
2 Repeat $T$ times
3  **for** each $(x, y) \in \mathcal{D}$ (in random order)
4  $\tilde{y} \leftarrow \arg\max_{y' \in \mathcal{Y}} \text{Score}_{\mathbf{w}}(x, y') + c(y, y')$
5  $\mathbf{w} \leftarrow \mathbf{w} + \eta(\phi(x, y) - \phi(x, \tilde{y}))$
6 Return $\mathbf{w}$

Figure 1: The stochastic gradient descent (SGD) optimization algorithm.

# 4 Synthesis

Thus far, we have allowed compositionality and learning to each tell its own story of generalization and productivity. We now show that the two are intimately related. Both concern the ability of a system (human or artificial) to generalize from a finite set of experiences to a creative capacity, and to come to grips with new inputs and experiences effectively. From this perspective, compositionality is a claim about the nature of this ability when it comes to linguistic interpretation, and learning theory offers a framework for characterizing the conditions under which a system can attain this ability in principle. Moreover, establishing the relationship between compositionality and learning provides a recipe for synthesis: the principle of compositionality guides researchers on specific model structures, and machine learning provides them with a set of methods for training such models in practice.

More specifically, the claim of compositionality is that being a semantic interpreter for a language $L$ amounts to mastering the syntax of $L$, the lexical meanings of $L$, and the modes of semantic combination for $L$. This also suggests the outlines of a learning task. The theories sketched above suggest a number of ways of refining this task in terms of the triples $\langle u, s, d \rangle$. We discuss two in detail. The pure **semantic parsing** task (section 4.1) is to learn an accurate mapping from utterances $u$ to logical forms $s$. The **interpretation** task (section 4.2) is to learn an accurate mapping from utterances $u$ to denotations $d$ via latent semantic representations, in effect combining semantic parsing and interpretation.

Throughout our review of the two tasks, we rely on the small illustrative

example in figure 2. The figure is based around the utterance "two times two plus three". Candidate semantic representations are given in row (a). The middle candidate $y_2$ contains a lexical pairing that is illicit in the true, target grammar (table 1). This candidate is a glimpse into the unfettered space of logical forms that our learning algorithm needs to explore. Our feature vector counts lexical pairings and inspects the root-level operator, as summarized in the boxes immediately below each candidate. Row (b) of figure 2 describes how the semantic parsing model operates on these candidates, and row (c) does the same for the interpretation model. The next two subsections describe these processes in detail.

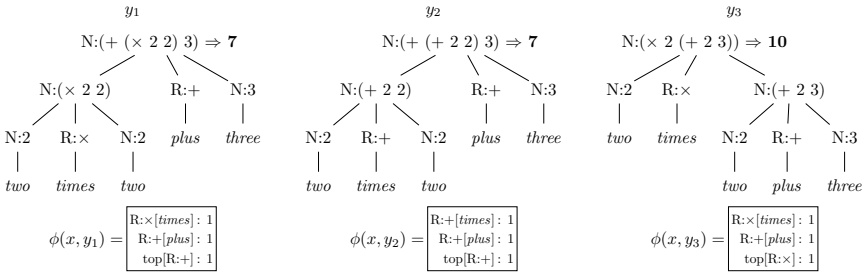
## 4.1 Learning from logical forms

Semantic parsing is the task of mapping from sentences in $\mathcal{X}$ to logical forms in $\mathcal{Y}$. Early work on this task was purely logical (Woods et al. 1972; Warren & Pereira 1982), but statistical methods, pioneered by Zelle & Mooney (1996), Thompson & Mooney (2003), Tang & Mooney (2001), and Zettlemoyer & Collins (2005), have become prevalent over the last twenty years. On our simplified construal, where each logical form is a full tree structure, as in figure 2a, semantic parsing can be treated as an application of the general learning framework we presented in section 3.2. However, it poses special challenges relating to the fact that the space of outputs is an infinite set. Successfully addressing these challenges involves integrating insights about compositionality into the learning model. In particular, we rely on the grammar to define a function $\text{GEN}(x) \subset \mathcal{Y}$ specifying a finite space of outputs for each input utterance $x$.

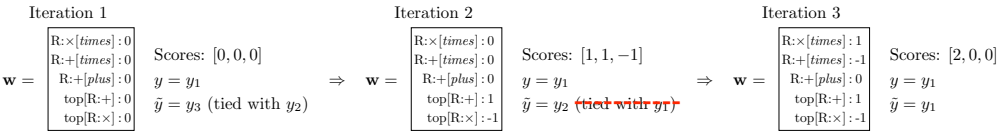
Let us revisit our running arithmetic example. At some level, the goal of learning is to infer the true grammar rules (table 1) from data. To keep things manageable, we assume that the two modes of combination (given in the final two rows of table 1) are already known and that the basic format of the lexical rules is given as '$a \rightarrow b : c$', where $a$ is a pre-terminal symbol, $b$ is a word compatible with $a$, and $c$ is a logical form. This facilitates simultaneous learning of the syntactic category of $b$ and (more importantly for us) the lexical translation relation between $b$ and $c$. In essence, with these two assumptions, the central challenge of learning is lexical in nature. This coincides with the assumption, discussed in section 3.1, that the heart of a grammar is in its lexicon, though, of course, in the end we obtain not a set of rules but a weight vector that encodes lexical, grammatical, and even pragmatic preferences in a soft way.

At the start, we create a crude grammar that *overgenerates*. What is important at this point is that (i) the grammar is extremely simple to specify manually and (ii) the set of candidates $\text{GEN}(x)$ is likely to contain the correct logical form $y$. To meet these two conditions, we add rules of the form '$\text{N} \rightarrow b : c$' for all choices of $b$ and $c$. Some of these will be correct (e.g., '$\text{N} \rightarrow \text{one} : \texttt{1}$') but many more will be incorrect (e.g., '$\text{N} \rightarrow \text{one} : \texttt{2}$'). The absurdity of this grammar may be evident to us as competent users of language, but such judgments are precisely the knowledge we want our system to learn from data. Next, we define a feature vector $\phi(x, y)$. The hope is that, upon examining the training data, the

**(a) Candidates** GEN($x$) for utterance $x =$ *two times two plus three*




**(b) Learning from logical forms (Section 4.1)**




**(c) Learning from denotations (Section 4.2)**

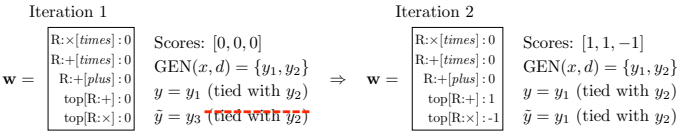



Figure 2: Learning algorithms applied to one example. The utterance is "two times two plus three". In (b), we are given the target semantic representation `(+ (× 2 2) 3)`. In (c), we are given the target denotation 7. In (a), we show the three candidates GEN($x$) that the learning algorithm must decide among, along with their features $\phi$. For example, the first feature in $y_1$ ('R:×[*times*]') has value 1, and says that *times* was mapped to R:×; the last feature 'top[R:+]' says that the topmost R is a +. The feature weights start at all zero; we compute the scores for each of the three candidates ($\text{Score}_{\mathbf{w}}(x, y_i) = \mathbf{w} \cdot \phi(x, y_i)$). Either the target $y$ is provided or, if only the denotation $d$ is given, we choose the highest scoring $y$ that has denotation $d$: $y = \arg\max_{y' \in \text{GEN}(x,d)} \text{Score}_{\mathbf{w}}(x, y')$. The prediction $\tilde{y} = \arg\max_{y' \in \text{GEN}(x)} \text{Score}_{\mathbf{w}}(x, y') + c(y, y')$ is the highest score (augmented with the cost, which is 1 for $y' \neq y$ and 0 otherwise). Finally, a weight update is made: $\mathbf{w} \leftarrow \mathbf{w} + \eta(\phi(x, y) - \phi(x, \tilde{y}))$ with $\eta = 1$. The algorithm terminates in this example when $y = \tilde{y}$. From logical forms, we eventually predict the correct answer $y_1$. From denotations, we end up predicting either $y_1$ or $y_2$, which both have denotation $d$. With more diverse examples, we would be able to favor $y_1$.

bad features $\phi(x, y)_i$ will receive low weight, which suppresses some candidates and, in turn, refines GEN($x$) in a soft way. To achieve this within our learning framework (section 3.2), we modify the objective function (2) by replacing all possible logical forms $\mathcal{Y}$ with those generated by the crude grammar GEN($x$). (Other methods are based in log-linear models; for simplicity, we stick to our earlier paradigm.)

Figure 2 shows how learning proceeds in concrete terms, using a small example from our grammar. The input $x$ is "two times two plus three", and our target logical form is $y_1$. Due to space constraints, we define the candidate set to be GEN($x$) = $\{y_1, y_2, y_3\}$; in reality, even our simple grammar generates more possibilities. The feature vector for each candidate logical form is given in a box immediately below it. The features are mostly lexical ones like 'R:×[*times*]', which count the number of times that the rule was used to construct the logical form, but there is also a class of structural features 'top[R:$r$]', which has value 1 if there is a topmost relation and it has logical form $r$, otherwise 0. Figure 2b depicts two iterations of SGD (figure 1). After one iteration, $y_3$ has fallen behind because of its root-level relation symbol, leaving $y_1$ and $y_2$ tied. After two iterations, our target $y_1$ has pulled ahead. By that stage, the incorrect lexical rule '$R \rightarrow$ times : ~~×~~ +' used to construct $y_2$ has been assigned negative weight, pulling $y_2$ down.

From the perspective of linguistic theory, the learning framework above represents an important shift in philosophy. Whereas full-fledged grammars like table 1 depend on the analyst to precisely define the right set of mappings from syntax to logical form, learning shifts some of this burden to the data and feature representations. In this setting, given enough representative training data, the analyst would need to define only the structural aspects of the grammar, leaving the lexical aspects to learning. In practice, the distinction is more blurred. For the most part, learning-based theories have assumed that lexical items like interrogative pronouns, quantificational determiners, and logical connectives are fixed, so that learning focuses on open-class lexical items. The long-term trends for this division of labor are hard to forecast, but we expect to increasingly see models that are able to learn these more abstract corners of the lexicon as well, since defining richer views of the lexicon has been a major focus of recent work (Wong & Mooney 2007; Kwiatkowski et al. 2011).

For the sake of concreteness, we incorporated a number of simplifying assumptions into the example represented by figure 2. The models presented in the literature tend to be considerably more complex along a number of dimensions. We mention two here. First, in our model, we observe the entire logical form tree structure. In contrast, for Zettlemoyer & Collins (2005) and related approaches, only the final logical expression on the root node of those trees is observed, with the tree structure (the derivational path to the root) treated as a latent variable. This corresponds to a reduction in the amount of information available to the model, thereby deepening the learning challenge. Second, in our model, we seek to learn only the lexicon. However, the lexicon is not all there is to compositionality. Though we might seek to minimize the complexity of the modes of combination, they will always be needed for the structures attested in

language. More ambitious versions of the above model do indeed seek to learn these modes of combination, using largely the same technique as we employed above for the lexicon: hypothesize a wide range of potential modes of combination and rely on the data to learn preferences regarding which modes to retain in the learned grammar (Zettlemoyer & Collins 2007).

While the crude grammar and data provide an appealing division of labor between manual effort and learning, the approach does present significant computational challenges, stemming from the fact that the number of candidate logical forms $\text{GEN}(x)$ is in general exponential in the length of the sentence. For example, in our simple grammar, even with just nine atomic numerical expressions to consider, the number of logical forms for the short utterance "two times two plus tree" is $9^3 \times 3^2 \times 2 = 36,450$. The issue can be mitigated somewhat by dynamic programming algorithms for parsing. In syntactic parsing, for example, such algorithms can find the highest scoring parse for each contiguous span of the utterance given a syntactic category, building on previously optimal steps as they go and thus never exploring large numbers of suboptimal choices. In semantic parsing, however, each subproblem needs to be parametrized, not just by the syntactic category, but also by the entire logical form. To address this, beam search approximations are typically employed, where only $k$ parses are stored for each span across all logical forms. Zettlemoyer & Collins (2005, 2007) take further steps towards reducing the size of the candidate space by generating the crude grammar on the fly: for each example $(x, y)$, their algorithm maps each word in the utterance $x$ to each predicate in the logical form $y$. This keeps the grammar small, which in turn helps tame $\text{GEN}(x)$. More could be said on the complexity issues surrounding these approaches; the above remarks are meant only to emphasize that complexity can often be a formidable barrier that requires algorithmic ingenuity and solid engineering to break down.

## 4.2 Learning from denotations

Arguably the major shortcoming of semantic parsing is that it leaves the task of interpretation — of associating utterances with their denotations — outside of the learning model itself. We turn now to the interpretation task, where we learn from denotations directly, using logical forms only as intermediate, hidden representations along the way. This is a more realistic portrayal of human language acquisition and is also advantageous from the point of view of developing artificial systems.

In the interpretation task, the training instances are utterance–denotation pairs $(x, d)$, and the logical forms $y$ are not observed. This requires a reformulation of our objective function (2). We use a latent support vector machine objective (Yu & Joachims 2009):

$$\min_{\mathbf{w}\in\mathbb{R}^d} \sum_{(x,d)\in\mathcal{D}} \max_{y'\in\text{GEN}(x)} [\text{Score}_{\mathbf{w}}(x, y') + c(y, y')] - \max_{y\in\text{GEN}(x,d)} \text{Score}_{\mathbf{w}}(x, y), \quad (3)$$

where $\text{GEN}(x, d) = \{y \in \text{GEN}(x) : [\![y]\!] = d\}$ is the set of logical forms that evaluate to denotation $d$. The only change from (2) is that the score of the correct

logical form $y$ has been replaced with the maximum score over all logical forms $y \in \text{GEN}(x, d)$ that have the correct denotation.

As before, we can apply SGD to this objective. The only modification from figure 1 is that the inner **for**-loop is now over pairs $(x, d)$, and thus the correct logical form $y$ is no longer given. Rather, $y$ is chosen to be the member of $\text{GEN}(x, d)$ with the highest score (thus, $y$ might change as learning progresses). Figure 2c walks through the first steps of this optimization process. The input is $x =$ "two times two plus three", and the output is $d = 7$. Thus, $\text{GEN}(x, d) = \{y_1, y_2\}$; the logical form $y_3$ has an incompatible denotation. In this simple example, the model cannot distinguish between $y_1$ and $y_2$, since both have identical denotations. Of course, with additional diverse examples, we could resolve this uncertainty; the purpose of this illustration is to highlight the loss of information due to learning directly from denotations.

Figure 2 also highlights the subservient role that logical forms play in the interpretation task. We do not observe them in training nor do we evaluate the model in terms of their accuracy. As far as the interpretation task is concerned, the model in figure 2 performs perfectly on this example, even if it is for the "wrong reason". If we only ever saw "plus" in the context of "two" and "two", then we would be forever blissfully indifferent to the choice of $y_1$ and $y_2$. More realistically, if we later received examples with other arguments, then we would be able to break the symmetry and favor $y_1$.

Learning from denotations is more difficult than learning from logical forms, not only information-theoretically, but also computationally. When learning from logical forms, we could constrain the lexicon by including a word–predicate pair only if it occurs in our training data. Without logical forms, naively, we would have to pay the full the price of having a completely uninformed lexicon that includes all possible word–predicate pairs. Computationally, this is only feasible in the simplest scenarios. To manage the computational complexity, we combine two key ideas: (i) using type information and (ii) controlling the order in which logical forms are built (Liang et al. 2011, 2013; Berant & Liang 2014). We assume that we have the lexical entries for a set of "easy" words (e.g., "one" $\rightarrow$ 1), but we do not have lexical entries for others (e.g., "plus"). As a simple example, consider the sentence "one plus two". Rather than guessing all possible predicates for "plus", we can first parse "one" into 1 and "two" into 2 (this part could be non-deterministic). Now, we can restrict the search to relational predicates. On the toy arithmetic domain, the gains would be minimal, but in problems involving large-scale question-answering databases and complex language, it can facilitate vast reductions in the candidate space.

On the positive side, there is one sense in which denotations provide more information than logical forms alone: we can define features on denotations. For example, suppose that, at test time, we are given the phrase "the father of Daniel Bernoulli" and the two candidate logical forms `father(DanielBernoulli)` and `daughter(DanielBernoulli)`. When we interpret each in our model, the first yields {Jakob Bernoulli} and the second yields the empty set. We could define a feature and learn its associated weight to favor logical forms in which the complement to "the" has denotation with cardinality 1, thus favoring the first

choice in this case.

These ideas are developed in detail by Liang et al. (2011, 2013) using dependency representations that can contain a wide range of quantificational operators and can even model scope ambiguities of the sort we discussed briefly in section 2.2. These models match the accuracies obtained by learning from logical forms. What's more, since denotations (or good proxies for them) are abundant in naturally occurring data in a way that logical forms are not, these models hold the promise of being applicable in a wide range of domains with little need for laborious annotation. This promise is beginning to be realized in practice (Berant et al. 2013; Berant & Liang 2014; Kwiatkowski et al. 2013).

## 4.3 Extensions

We presented only skeletal versions of the above theories, but we sought to capture the essential spirit of the synthesis between learning and compositionality. Over the last decade, the computational community has been expounding on variants of these ideas and actively applying them to a number of different areas. In this brief section, we hope to convey the richness and diversity of this work and to provide references for the interested reader.

One of the great strengths of learning-based approaches is how easily they can be adapted to new settings — both in terms of languages and in terms of discourse domains. The high-level paradigm is as follows: the feature representations and learning algorithm are language- and domain-independent, creating a vessel. The crude grammar and data fill the vessel with language- and domain-dependent information. This paradigm has been realized successfully in practice. Much early statistical work in semantic parsing (Zettlemoyer & Collins 2005, 2007; Ge & Mooney 2005; Kate et al. 2005; Kate & Mooney 2006; Wong & Mooney 2006, 2007; Lu et al. 2008) focused on mapping questions to structured database queries in the domains of US geography (Geo880; Zelle & Mooney 1996), job postings (Jobs640; Tang & Mooney 2001), and ATIS air travel planning (Zettlemoyer & Collins 2007). Matuszek et al. (2012b) show that essentially the same model can learn to map instructions to a robot's programming language, thereby allowing a user to guide the robot around a building. Cai & Yates (2013) learn models for question-answering on Freebase (Bollacker et al. 2008) across 81 different domains. Finally, Kwiatkowksi et al. (2010) and Kwiatkowski et al. (2011) learn semantic parsers across four languages (English, Spanish, Turkish, and Japanese).

Another major theme that arose from section 4.2 is the importance of learning from denotations — more generally, connecting language with the world. Many authors have considered this setting using a variety of intermediate semantic representations. Applications include querying databases (Clarke et al. 2010; Liang et al. 2011; Berant et al. 2013; Kwiatkowski et al. 2013), interpreting natural language for performing programming tasks (Kushman & Barzilay 2013; Lei et al. 2013), playing computer games (Branavan et al. 2010, 2011), following navigational instructions (Vogel & Jurafsky 2010; Chen 2012; Artzi & Zettlemoyer 2013), implementing dialogue systems (Artzi & Zettlemoyer 2011),

and interacting in the real world via perception (Matuszek et al. 2012a; Tellex et al. 2011; Krishnamurthy & Kollar 2013). The trend clearly points toward incorporating more compositionality into these applications.

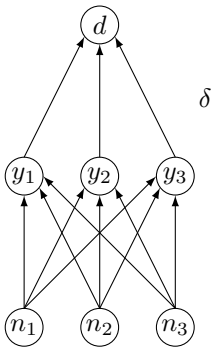
# 5 Distributed representations

We have so far concentrated on semantic representations that are logical forms. We now introduce an alternative perspective on which semantic representations are **distributed representations** — vectors and matrices. Their real-valued nature provides a foundation for representing shades of meaning and, with work, they can encode the same kinds of information as logical forms. Due to space limitations, our discussion is brief, meant only to convey the general ideas and provide connections with the vibrant literature.

For concreteness, we structure our discussion around a small example involving adjectival modification (Mitchell & Lapata 2010), adapted from propositional logic cases analyzed by Rumelhart et al. (1986a). Table 4 presents the grammar. The semantic representations for nouns are three-dimensional column vectors. (We write them as transposed row vectors to save space; $a^\top$ is the transpose of $a$.) As an informal aid to understanding, one can think of the first dimension as encoding concreteness and the second as encoding interactivity. The third is just a bias term that is always 1. Our single adjective "unpredictable" is represented as a $3 \times 3$ matrix, in analogy with adjectives denoting functions on noun-type meanings in logical approaches.

| Syntax | Representation |
|---|---|
| N → rollercoaster | $[1.0\ 1.0\ 1.0]^\top$ |
| N → airplane | $[1.0\ 0.0\ 1.0]^\top$ |
| N → website | $[0.0\ 1.0\ 1.0]^\top$ |
| N → movie | $[0.0\ 0.0\ 1.0]^\top$ |
| A → unpredictable | $\begin{bmatrix} -6.3 & -6.3 & 2.5 \\ -4.4 & -4.4 & 6.5 \\ 0.0 & 0.0 & 1.0 \end{bmatrix}$ |
| N → A N | $\sigma(\ulcorner \mathrm{A} \urcorner \ulcorner \mathrm{N} \urcorner)$ |

(a) Grammar.




$d = [\![y]\!] = \sigma(\delta y)$

$\delta = [9.0\ -8.8\ 5.6]$

$y = \sigma(\ulcorner \mathrm{A} \urcorner \ulcorner \mathrm{N} \urcorner)$

$\ulcorner \mathrm{A} \urcorner$

$\ulcorner \mathrm{N} \urcorner$

(b) Composition via a neural network.

Table 4: Grammar for distributed representations.

The entire compositional computation of the denotation (table 4b) can be described as a two-layer **neural network**. The inputs to the network are the noun vectors. This vector is multiplied by the adjective representation $\ulcorner \mathrm{A} \urcorner$; the arrows represent the dense connections between the coordinates of the input and

output vectors. The result is another vector representation ($\ulcorner \mathrm{A} \urcorner \ulcorner \mathrm{N} \urcorner$), which is passed to the sigmoid function ($\sigma(z) = (1+e^{-z})^{-1}$), which monotonically transforms each coordinate into a value in the interval $[0, 1]$. (The non-linear nature of this function is also key to the model's representational power.) The result is a modified-noun representation $y$ in the same space (of the same semantic type) as the input noun, so it could be further modified, just as one would expect from a compositional theory.

The second layer of the neural network maps the modified-noun representation to its denotation $d = [\![y]\!]$, which is just a single value in $[0, 1]$. The weights on this layer are $\delta$, which parametrize the interpretation function according to $[\![y]\!] = \sigma(\delta y)$. In our example, this can be thought of as an evaluative denotation, with values near 1 being positive and values near 0 being negative. This is a very limited denotation, but it suffices for illustrative purposes, and networks like these can have multi-dimensional output labels encoding lots of different kinds of information; in terms of the network structure, this just means changing $\delta$ into a matrix.

Our guiding semantic intuition is that the evaluative sense of an adjective like "unpredictable" is influenced by the noun it modifies (Partee 1995). An "unpredictable rollercoaster" and "unpredictable movie" are evaluatively positive (denotation near 1), whereas "unpredictable airplane" and "unpredictable website" are evaluatively negative (denotation near 0). Figure 3 summarizes how modification works in this theory to capture these intuitions. The lexical noun vectors are given in the two-dimensional space (the third dimension is always 1). The arrows depict the action of our composition function, which maps the noun meanings into different parts of the space as a result of their adjectival modification. In this case, the network brought the negative phrases to the same vector and more or less swapped the positions of the two positive phrases. (Other vectors of the adjective will have different effects but lead to similar groupings.) The colors reflect the final output denotation, with blue indicating a denotation near 1 and red indicating a denotation near 0.

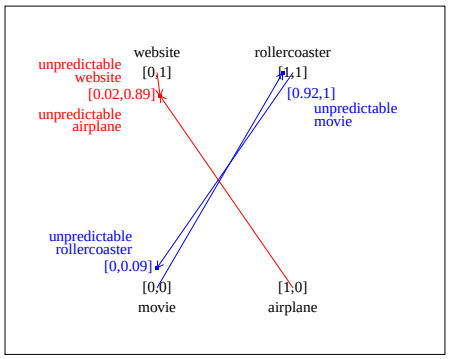



Figure 3: Modification by "unpredictable" using the grammar in table 4. Blue indicates positive denotations, red negative ones.

The goal of learning is to obtain the parameters, which consist of the interpretation vector $\delta$ and the semantic representations in table 4a. We limit attention here to learning the matrix representation of the adjective $\ulcorner$unpredictable$\urcorner$, taking the noun vectors as given. Table 5 shows our training examples $(x, d)$, where each $x$ decomposes into an adjective–noun pair $(a, n)$. We can apply the standard supervised learning task (section 3.2). Given parameters, our predicted denotation is $\tilde{d} = \sigma(\delta\ \sigma(\ulcorner a \urcorner \ulcorner n \urcorner))$. The objective function is to choose parameters that minimize the cross-entropy of the observed denotation $d$ and the predicted denotation $\tilde{d}$:

$$\min_{\delta \in \mathbb{R}^3, \ulcorner \text{unpredictable} \urcorner \in \mathbb{R}^{3\times 3}} \sum_{(x,d)\in\mathcal{D}} -\left(d \log \tilde{d} + (1-d)\log(1-\tilde{d})\right). \tag{4}$$

To optimize this objective, we apply stochastic gradient descent as before. For neural networks, computing the gradients involves repeatedly applying the chain rule, and the classic **backpropagation algorithm** provides a way to organize this computation (Rumelhart et al. 1986a,b). We initialize $\ulcorner$unpredictable$\urcorner$ and $\delta$ with small, normally distributed random values. For each given input $((a, n), d)$, where $(a, n)$ is a phrase and $d$ is one of the scalar sentiment denotations in table 5, the algorithm computes the representation $\tilde{y}$ and denotation $\tilde{d}$ (forward propagation). This prediction is compared with the actual value $d$, and the resulting error signal is sent back through the network, which modifies $\delta$ and $\ulcorner$unpredictable$\urcorner$ to try to make the network's predictions more accurate (backward propagation).

| Phrase $x = (a, n)$ | Denotation $d$ |
|---|---|
| unpredictable rollercoaster | 1 |
| unpredictable airplane | 0 |
| unpredictable website | 0 |
| unpredictable movie | 1 |

Table 5: Training data used to learn the parameters $\ulcorner$unpredictable$\urcorner$ and $\delta$.

Vector-based representations trace historically to early distributional approaches to language (Firth 1935; Harris 1954), on which meanings (and other linguistic properties) are characterized entirely by co-occurrence statistics in corpora. However, as our example shows, modern versions of this idea allow that the counts might represent considerably more abstract information relating not only to co-occurrence at different levels of analysis but also associations with language-external denotations and even logical relationships familiar from work involving logical forms (Clark et al. 2011). In more complex models, words are represented not just by vectors but also by matrices and higher-dimensional tensors, which creates the possibility for richer modes of semantic combination. For a review of proposals that can be cast as elaborations of the above, see Socher et al. 2013b: §4.

In practice, distributed representations typically encode not just semantic information, but also a mix of morphological, syntactic, and semantic associations (Turney & Pantel 2010; Grefenstette et al. 2011; Lewis & Steedman 2013). Supervised training of distributed representations has, in turn, proven helpful for a variety of problems, including syntactic parsing (Socher et al. 2013a), part-of-speech tagging, named entity recognition, noun phrase chunking, and semantic role labeling (Collobert & Weston 2008; Collobert et al. 2011).

To date, these models have been applied only to narrow aspects of the full task of semantic interpretation, with the bulk of the work going to sentiment analysis, where the denotations can modeled by a small set of categories or values in a predefined numerical range, as we did above (Socher et al. 2011b, 2012, 2013b). Bowman (2014) reports on experiments assessing how well these models do at capturing core semantic properties related to entailment and contradiction, and Socher et al. (2011a) obtain successful results on the related task of paraphrase detection (a kind of synonymy); see also Baroni et al. 2012 for work on entailment and compositionality with distributional representations. At present, one of the most exciting open questions in computational semantics is what kind of data and models will be needed to operate on richer denotation domains.

# 6 Conclusion

Within computational semantics, logical and statistical approaches are often regarded as separate. Luckily, this seems not to stem from the kind of rough and tumble history that Pereira (2000) describes for syntax, but rather from a perception that the two groups are working on different things. With this paper, we sought unity around the notion of **generalization** as it pertains to meaning and structural complexity. Intuitively, the compositionality hypothesis defines a learning task in which the goal is to acquire mappings between forms and meanings that will be effective with new, arbitrarily complex structures. A great deal of recent work has identified methods for solving this problem. We presented a simple framework, combining discriminative learning with formal grammar, but this is just a glimpse into this thriving area of investigation. For linguists, we hope this review provides a fruitful new perspective on the nature and acquisition of grammar and meaning; for computer scientists, we hope it reveals new ways in which linguistic insights and concepts can support more robust natural language technologies and define richer learning problems.

# References

AnderBois S, Brasoveanu A, Henderson R. 2012. The pragmatics of quantifier scope: A corpus study. In *Proceedings of Sinn und Bedeutung 16*, eds. A Aguilar-Guevara, A Chernilovskaya, R Nouwen, vol. 1 of *MIT Working Papers in Linguistics*. Cambridge, MA: MIT Linguistics

Artzi Y, Zettlemoyer LS. 2011. Bootstrapping semantic parsers from conversations. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Edinburgh: ACL

Artzi Y, Zettlemoyer LS. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics* 1:49–62

Baklr G, Hofmann T, Schölkopf B, Smola AJ, Taskar B, eds. 2010. *Predicting Structured Data*. Cambridge, MA: MIT Press

Baroni M, Bernardi R, Do NQ, Shan Cc. 2012. Entailment above the word level in distributional semantics. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*. Avignon, France: ACL

Berant J, Chou A, Frostig R, Liang P. 2013. Semantic parsing on Freebase from question–answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle: ACL

Berant J, Liang P. 2014. Semantic parsing via paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Baltimore: ACL

Bird S, Klein E, Loper E. 2009. *Natural Language Processing with Python*. Sebastopol, CA: O'Reilly Media

Blackburn P, Bos J. 2003. Computational semantics. *Theoria* 18:27–45

Blackburn P, Bos J. 2005. *Representation and Inference for Natural Language: A First Course in Computational Semantics*. Stanford, CA: CSLI

Bollacker K, Evans C, Paritosh P, Sturge T, Taylor J. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *International Conference on Management of Data*. Vancouver: ACM

Bottou L. 2010. Large-scale machine learning with stochastic gradient descent. In *Proceedings of the 19th International Conference on Computational Statistics*, eds. Y Lechevallier, G Saporta. Berlin: Springer, 177–186

Bowman SR. 2014. Can recursive neural tensor networks learn logical reasoning? In *Proceedings of the 2nd International Conference on Learning Representations*. Banff, Canada

Boyd S, Vandenberghe L. 2004. *Convex Optimization*. Cambridge: Cambridge University Press

Branavan S, Silver D, Barzilay R. 2011. Learning to win by reading manuals in a monte-carlo framework. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: ACL

Branavan S, Zettlemoyer L, Barzilay R. 2010. Reading between the lines: Learning to map high-level instructions to commands. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Uppsala, Sweden: ACL

Cai Q, Yates A. 2013. Large-scale semantic parsing via schema matching and lexicon extension. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Sofia, Bulgaria: ACL

Carpenter B. 1997. *Type-Logical Semantics*. Cambridge, MA: MIT Press

Chen D. 2012. Fast online lexicon learning for grounded language acquisition. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Jeju Island, Korea: ACL

Chierchia G, Turner R. 1988. Semantics and property theory. *Linguistics and Philosophy* 11:261–302

Clark S, Coecke B, Sadrzadeh M. 2011. Mathematical foundations for a compositional distributed model of meaning. *Linguistic Analysis* 36:345–384

Clarke J, Goldwasser D, Chang MW, Roth D. 2010. Driving semantic parsing from the world's response. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*. Uppsala, Sweden: ACL

Collobert R, Weston J. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine learning*, ICML '08. Helsinki, Finland: ACM

Collobert R, Weston J, Bottou L, Karlen M, Kavukcuoglu K, Kuksa P. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12:2493–2537

Cooper R. 2012. Book review: Computational semantics with functional programming Jan van Eijck and Christina Unger. *Computational Linguistics* 38:447–449

Dagan I, Glickman O, Magnini B. 2006. The PASCAL recognising textual entailment challenge. In *Machine Learning Challenges, Lecture Notes in Computer Science*, eds. J Quinonero-Candela, I Dagan, B Magnini, F d'Alché Buc, vol. 3944. Springer-Verlag

de Marneffe MC, MacCartney B, Manning CD. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation*. ACL

Domingos P. 2012. A few useful things to know about machine learning. *Communications of ACM* 55:78–87

Dowty D. 2007. Compositionality as an empirical problem. In *Direct Compositionality*, eds. C Barker, P Jacobson. Oxford: Oxford University Press, 23–101

van Eijck J, Unger C. 2010. *Computational Semantics with Functional Programming*. Cambridge: Cambridge University Press

Firth JR. 1935. The technique of semantics. *Transactions of the Philological Society* 34:36–73

Ge R, Mooney R. 2005. A statistical semantic parser that integrates syntax and semantics. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*. Ann Arbor, MI: ACL

Graff D. 2000. Shifting sands: An interest-relative theory of vagueness. *Philosophical Topics* 28:45–81

Grefenstette E, Sadrzadeh M, Clark S, Coecke B, Pulman S. 2011. Concrete sentence spaces for compositional distributional models of meaning. In *Proceedings of the 9th International Conference on Computational Semantics (IWCS 2011)*. Portland, OR: ACL

Harris Z. 1954. Distributional structure. *Word* 10:146–162

Heim I, Kratzer A. 1998. *Semantics in Generative Grammar*. Oxford: Blackwell

Higgins D, Sadock JM. 2003. A machine learning approach to modeling scope preferences. *Computational Linguistics* 29:73–96

Jackendoff RS. 1992. *Languages of the Mind*. Cambridge, MA: MIT Press

Jackendoff RS. 1997. *The Architecture of the Language Faculty*. Cambridge, Massachusetts: The MIT Press

Janssen TMV. 1997. Compositionality. In *Handbook of Logic and Language*, eds. J van Benthem, A ter Meulen. Cambridge, MA and Amsterdam: MIT Press and North-Holland, 417–473

Jurafsky D, Martin JH. 2009. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Englewood Cliffs, NJ: Prentice-Hall, 2nd ed.

Kamp H, Partee BH. 1995. Prototype theory and compositionality. *Cognition* 57:129–191

Kamp H, Reyle U. 1993. *From Discourse to Logic: Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory*. Dordrecht: Kluwer

Kate RJ, Mooney RJ. 2006. Using string-kernels for learning semantic parsers. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*. Sydney, Australia: ACL

Kate RJ, Wong YW, Mooney RJ. 2005. Learning to transform natural to formal languages. In *Proceedings of the 20th National Conference on Artificial Intelligence - Volume 3*. Pittsburgh, PA: AAAI Press

Katz JJ. 1972. *Semantic Theory*. New York: Harper & Row

Katz JJ. 1996. Semantics in linguistics and philosophy: An intensionalist perspective. In Lappin (1996), 599–616

Katz JJ, Fodor JA. 1963. The structure of semantic theory. *Language* 39:170–210

Kennedy C. 2011. Ambiguity and vagueness. In *Semantics: An International Handbook of Natural Language Meaning*, eds. C Maienborn, K von Heusinger, P Portner, vol. 1. Berlin: Mouton de Gruyter, 507–535

Klein E, Sag IA. 1985. Type-driven translation. *Linguistics and Philosophy* 8:163–201

Kripke S. 1975. Outline of a theory of truth. *Journal of Philosophy* 72:690–716

Krishnamurthy J, Kollar T. 2013. Jointly learning to parse and perceive: Connecting natural language to the physical world. *Transactions of the Association for Computational Linguistics* 1:193–206

Kushman N, Barzilay R. 2013. Using semantic unification to generate regular expressions from natural language. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Atlanta, Georgia: ACL

Kwiatkowksi T, Zettlemoyer L, Goldwater S, Steedman M. 2010. Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. Cambridge, MA: ACL

Kwiatkowski T, Choi E, Artzi Y, Zettlemoyer L. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, Washington: ACL

Kwiatkowski T, Zettlemoyer LS, Goldwater S, Steedman M. 2011. Lexical generalization in CCG grammar induction for semantic parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Edinburgh: ACL

Landauer TK, McNamara DS, Dennis S, Kintsch W, eds. 2007. *Handbook of Latent Semantic Analysis*. Hillsdale, NJ: Lawrence Erlbaum Associates

Lappin S, ed. 1996. *The Handbook of Contemporary Semantic Theory*. Oxford: Blackwell Publishers

Lei T, Long F, Barzilay R, Rinard M. 2013. From natural language specifications to program input parsers. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Sofia, Bulgaria: ACL

Lewis D. 1970. General semantics. *Synthese* 22:18–67

Lewis M, Steedman M. 2013. Combined distributional and logical semantics. *Transactions of the Association for Computational Linguistics* 1:179–192

Liang P, Jordan M, Klein D. 2011. Learning dependency-based compositional semantics. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: ACL

Liang P, Jordan MI, Klein D. 2013. Learning dependency-based compositional semantics. *Computational Linguistics* 39:389–446

Lu W, Ng HT, Lee WS, Zettlemoyer LS. 2008. A generative model for parsing natural language to meaning representations. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*. Honolulu, HI: ACL

MacCartney B, Manning CD. 2009. An extended model of natural logic. In *Proceedings of the Eighth International Conference on Computational Semantics*. Tilburg, The Netherlands: ACL

Manning CD, Schütze H. 1999. *Foundations of Statistical Natural Language Processing*. Cambridge, MA: MIT Press

Matuszek C, FitzGerald N, Zettlemoyer LS, Bo L, Fox D. 2012a. A joint model of language and perception for grounded attribute learning. In *Proceedings of the 29th International Conference on Machine Learning*, eds. J Langford, J Pineau. Omnipress

Matuszek C, Herbst E, Zettlemoyer LS, Fox D. 2012b. Learning to parse natural language commands to a robot control system. In *Proceedings of the 13th International Symposium on Experimental Robotics*

Mitchell J, Lapata M. 2010. Composition in distributional models of semantics. *Cognitive Science* 34:1388–1429

Mitchell TM. 1997. *Machine Learning*. New York: McGraw Hill

Montague R. 1970. Universal grammar. *Theoria* 36:373–398. Reprinted in Montague (1974), 222–246

Montague R. 1974. *Formal Philosophy: Selected Papers of Richard Montague*. New Haven, CT: Yale University Press

Moss LS. 2009. Natural logic and semantics. In *Proceedings of the 18th Amsterdam Colloquium: Revised Selected Papers*, eds. M Aloni, H Bastiaanse, T de Jager, P van Ormondt, K Schulz. Berlin: Springer

Ng HT, Zelle J. 1997. Corpus-based approaches to semantic interpretation in natural language processing. *AI Magazine* 18:45–64

Palmer M, Gildea D, Xue N. 2010. *Semantic Role Labeling*. San Rafael, CA: Morgan & Claypool

Partee BH. 1980. Semantics – mathematics or psychology? In *Semantics from Different Points of View*, eds. E Bäuerle, A von Stechow. Berlin: Springer-Verlag

Partee BH. 1981. Montague grammar, mental representations, and reality. In *Philosophy and Grammar*, eds. S Kanger, S Öhman. Dordrecht: D. Reidel, 59–78

Partee BH. 1984. Compositionality. In *Varieties of Formal Semantics*, eds. F Landman, F Veltman. Dordrecht: Foris, 281–311. Reprinted in Barbara H. Partee (2004) *Compositionality in formal semantics*, Oxford: Blackwell 153–181. Page references to the reprinting.

Partee BH. 1995. Lexical semantics and compositionality. In *Invitation to Cognitive Science*, eds. LR Gleitman, M Liberman, vol. 1. Cambridge, MA: MIT Press, 311–360

Pereira FCN. 2000. Formal grammar and information theory: Together again? *Philosophical Transactions of the Royal Society* 358:1239–1253

Reinhart T. 1997. Quantifier scope: How labor is divided between QR and choice functions. *Linguistics and Philosophy* 20:335–397

Rumelhart DE, Hinton GE, Williams RJ. 1986a. Learning internal representations by error propagation. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, eds. DE Rumelhart, JL McClelland, vol. 1: Foundations. Cambridge, MA: MIT Press, 318–362

Rumelhart DE, Hinton GE, Williams RJ. 1986b. Learning representations by back-propagating errors. *Nature* 323:533–536

Samuel AL. 1959. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development* 3:211–229

Samuel AL. 1967. Some studies in machine learning using the game of checkers. II — recent progress. *IBM Journal of Research and Development* 11:601–617

Saurí R, Pustejovsky J. 2009. FactBank: A corpus annotated with event factuality. *Language Resources and Evaluation* 43:227–268

Smith NA. 2011. *Linguistic Structure Prediction*. San Rafael, CA: Morgan & Claypool

Socher R, Bauer J, Manning CD, Andrew Y. N. 2013a. Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, vol. 1: Long Papers. Sofia, Bulgaria: ACL

Socher R, Huang EH, Pennin J, Manning CD, Ng AY. 2011a. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems 24*, eds. J Shawe-Taylor, RS Zemel, PL Bartlett, FCN Pereira, KQ Weinberger

Socher R, Huval B, Manning CD, Ng AY. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Conference on Empirical Methods in Natural Language Processing*. Jeju Island, Korea

Socher R, Pennington J, Huang EH, Ng AY, Manning CD. 2011b. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Edinburgh, Scotland, UK.: ACL

Socher R, Perelygin A, Wu J, Chuang J, Manning CD, et al. 2013b. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle: ACL

Szabó ZG. 2012. Compositionality. In *The Stanford Encyclopedia of Philosophy*, ed. EN Zalta. CSLI, winter 2012 ed.

Szabolcsi A. 2009. *Quantification*. Cambridge: Cambridge University Press

Tang LR, Mooney RJ. 2001. Using multiple clause constructors in inductive logic programming for semantic parsing. In *Proceedings of the 12th European Conference on Machine Learning*. London, UK, UK: Springer-Verlag

Taskar B, Guestrin C, Koller D. 2003. Max-margin Markov networks. In *Advances in Neural Information Processing Systems 17*, eds. S Thrun, LK Saul, B Schölkopf. Cambridge, MA: MIT Press

Tellex S, Kollar T, Dickerson S, Walter MR, Banerjee AG, et al. 2011. Understanding natural language commands for robotic navigation and mobile manipulation. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*. San Francisco: AAAI Press

Thompson CA, Mooney RJ. 2003. Acquiring word-meaning mappings for natural language interfaces. *Journal of Artificial Intelligence Research* 18:1–44

Turney PD, Pantel P. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research* 37:141–188

Vogel A, Jurafsky D. 2010. Learning to follow navigational directions. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Uppsala, Sweden: ACL

Warren DHD, Pereira FCN. 1982. An efficient easily adaptable system for interpreting natural language queries. *American Journal of Computational Linguistics* 8:110–122

Werning M, Hinzen W, Machery E. 2012. *The Oxford Handbook of Compositionality*. Oxford: Oxford University Press

Wong YW, Mooney R. 2006. Learning for semantic parsing with statistical machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*. New York City, USA: ACL

Wong YW, Mooney R. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*. Prague, Czech Republic: ACL

Woods WA, Kaplan RM, Nash-Webber B. 1972. The lunar sciences natural language information system: Final report. Tech. rep., BBN Report 2378, Bolt Beranek and Newman Inc.

Yu CNJ, Joachims T. 2009. Learning structural SVMs with latent variables. In *Proceedings of the 26th International Conference on Machine Learning*, eds. L Bottou, ML Littman. Madison, WI: Omnipress

Zelle JM, Mooney RJ. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence – Volume 2*. AAAI Press

Zettlemoyer LS, Collins M. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the Twenty First Conference on Uncertainty in Artificial Intelligence*

Zettlemoyer LS, Collins M. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. Prague, Czech Republic: ACL