Learning Distributions over Logical Forms for Referring Expression Generation

Nicholas FitzGerald Yoav Artzi Luke Zettlemoyer
Computer Science & Engineering
University of Washington
Seattle, WA 98195
{nfitz,yoav,lsz}@cs.washington.edu

Abstract

We present a new approach to referring expression generation, casting it as a density estimation problem where the goal is to learn distributions over logical expressions identifying sets of objects in the world. Despite an extremely large space of possible expressions, we demonstrate effective learning of a globally normalized log-linear distribution. This learning is enabled by a new, multi-stage approximate inference technique that uses a pruning model to construct only the most likely logical forms. We train and evaluate the approach on a new corpus of references to sets of visual objects. Experiments show the approach is able to learn accurate models, which generate over 87% of the expressions people used. Additionally, on the previously studied special case of single object reference, we show a 35% relative error reduction over previous state of the art.

1 Introduction

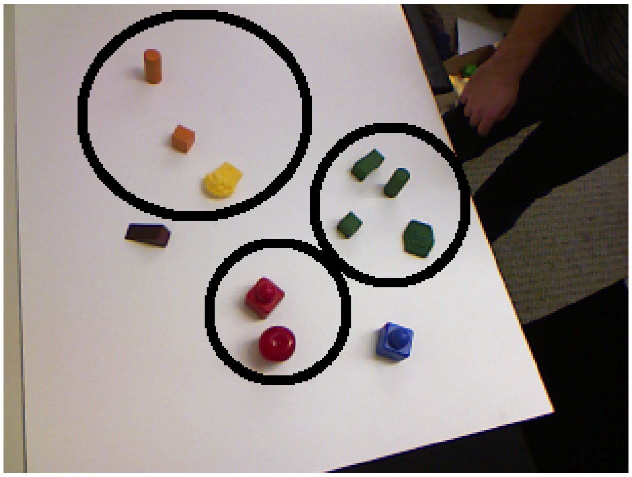
Understanding and generating natural language requires reasoning over a large space of possible meanings; while many statements might achieve the same goal in a certain situation, some are more likely to be used than others. In this paper, we model these preferences by learning distributions over situated meaning use.

We focus on the task of referring expression generation (REG), where the goal is to produce an expression which uniquely identifies a pre-defined object or set of objects in an environment. In practice, many such expressions can be produced. Figure 1 shows referring expressions provided by human subjects for a set of objects (Figure 1a), demonstrating variation in utterances (Figure 1b) and their corresponding meaning representations (Figure 1c). Although nearly a third of the people simply listed the colors of the desired objects, many other strategies were also used and no single option dominated. Learning to model such variation would enable systems to better anticipate what people are likely to say and avoid repetition during generation, by producing appropriately varied utterances themselves. With these goals in mind, we cast REG as a density estimation problem, where the goal is to learn a distribution over logical forms.

Learning such distributions is challenging. For a target set of objects, the number of logical forms that can be used to describe it grows combinatorially with the number of observable properties, such as color and shape. However, only a tiny fraction of these possibilities are ever actually used by people. We must learn to efficiently find these few, and accurately estimate their associated likelihoods.

We demonstrate effective learning of a globally normalized log-linear distribution with features to account for context dependence and communicative goals. We use a stochastic gradient descent algorithm, where the key challenge is the need to compute feature expectations over all possible logical forms. For that purpose, we present a multi-stage inference algorithm, which progressively constructs meaning representations with increasing complexity, and learns a pruning model to retain only those that are likely to lead to high probability expressions. This approach allows us to consider a large

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1914–1925,
Seattle, Washington, USA, 18-21 October 2013. ©2013 Association for Computational Linguistics



(a)

The green, red, orange and yellow toys. (1)
The green, red, yellow, and orange objects. (1)
The red, green, yellow and orange toys. (1)
The red, yellow, orange and green objects. (1)
All the green, red, yellow and orange toys. (1)
All the yellow, orange, red and green objects. (1)
All the pieces that are not blue or brown. (2)
All items that are not brown or blue. (2)
All items that are not brown or blue. (2)
Everything that is not brown or blue. (3)
Everything that is not purple or blue. (3)
All but the black and blue ones. (4)
Any toy but the blue and brown toys. (4)
Everything that is green, red, orange or yellow. (5)
All objects that are not triangular or blue. (6)
Everything that is not blue or a wedge. (7)
Everything that is not a brown or blue toy. (8)
All but the blue piece and brown wedge. (9)
Everything except the brown wedge and the blue object. (10)
All pieces but the blue piece and brown triangle shape. (11)

(b)

$\hat{P}(z \mid S,G)$	z	
0.30	$\iota(\lambda x.(yellow(x) \lor orange(x) \lor red(x) \lor green(x)) \land object(x) \land plu(x))$	(1)
0.15	$\iota(\lambda x.\neg(brown(x) \lor blue(x)) \land object(x) \land plu(x))$	(2)
0.10	$Every(\lambda x.\neg(brown(x) \lor blue(x)) \land object(x) \land sg(x))$	(3)
0.10	$Every(\lambda x.object(x) \land sg(x)) \setminus [\iota(\lambda x.(blue(x) \lor brown(x)) \land object(x) \land plu(x)]$	(4)
0.05	$Every(\lambda x.(yellow(x) \lor orange(x) \lor red(x) \lor green(x)) \land object(x) \land sg(x))$	(5)
0.05	$\iota(\lambda x.(triangle(x) \lor blue(x)) \land object(x) \land plu(x))$	(6)
0.05	$Every(\lambda x.object(x) \land sg(x)\neg(blue(x) \lor equal(x, \mathcal{A}(\lambda y.triangle(y) \land sg(y)))))$	(7)
0.05	$Every(\lambda x.object(x) \land sg(x) \land \neg equal(x, \mathcal{A}(\lambda y.(brown(y) \lor blue(y)) \land object(y) \land sg(y))))$	(8)
0.05	$Every(\lambda x.object(x) \land sg(x)) \setminus [\iota(\lambda x.(blue(x) \land object(x) \land sg(x)) \lor (brown(x) \land triangle(x) \land sg(x))]$	(9)
0.05	$Every(\lambda x.object(x) \land sg(x)) \setminus [\iota(\lambda x.brown(x) \land triangle(x) \land sg(x)) \cup \iota(\lambda y.blue(y) \land object(y) \land sg(x))]$	(10)
0.05	$\iota(\lambda x.object(x) \land plu(x)) \setminus [\iota(\lambda x.(blue(x) \land object(x) \land sg(x)) \lor (brown(x) \land triangle(x) \land object(x) \land sg(x))]$	(11)

(c)

Figure 1: An example scene from our object selection dataset. Figure 1a shows the image shown to subjects on Amazon Mechanical Turk. The target set G is the circled objects. Figure 1b shows the 20 sentences provided as responses. Figure 1c shows the empirical distribution $\hat{P}(z|G,S)$ for this scene, estimated by labeling the sentences in Figure 1b. The correspondence between a sentence in 1b and its labeled logical expression in 1c is indicated by the number in parentheses. Section 5.1 presents a discussion of the space of possible logical forms.

set of possible meanings, while maintaining computational tractability.

To represent meaning we build on previous approaches that use lambda calculus (Carpenter, 1997; Zettlemoyer and Collins, 2005; Artzi and Zettlemoyer, 2013b). We extend these techniques by modeling the types of plurality and coordination that are prominent in expressions which refer to sets.

We also present a new corpus for the task of referring expression generation.[1] While most previous REG data focused on naming single objects, to the best of our knowledge, this is the first corpus with sufficient coverage for learning to name sets of objects. Experiments demonstrate highly accurate learned models, able to generate over 87% of the expressions people used. On the previously studied special case of single object reference, we achieve state-of-the-art performance, with over 35% relative error reduction over previous state of the art (Mitchell et al., 2013).

2 Related Work

Referring expression generation has been extensively studied in the natural language generation

[1] The corpus was collected using Amazon Mechanical Turk and is available on the authors' websites.

community, dating as far back as SHRDLU (Winograd, 1972). Most work has built on variations of the Incremental Algorithm (Dale and Reiter, 1995), a deterministic algorithm for naming single objects that constructs conjunctive logical expressions. REG systems are used in generation pipelines (Dale and Reiter, 2000) and are also commonly designed to be cognitively plausible, for example by following Gricean maxims (Grice, 1975). Krahmer and van Deemter (2012) and van Deemter et al. (2012a) survey recent literature on REG.

Different approaches have been proposed for generating referring expressions for sets of objects. Van Deemter (2002) extended the Incremental Algorithm to allow disjunction and negation, enabling reference to sets. Further work attempted to resolve the unnaturally long expressions which could be generated by this approach (Gardent, 2002; Horacek, 2004; Gatt and van Deemter, 2007). Later, description logic was used to name sets (Areces et al., 2008; Ren et al., 2010). All of these algorithms are manually engineered and deterministic.

In practice, human utterances are surprisingly varied, loosely following the Gricean ideals (van Deemter et al., 2012b). Much recent work in REG has identified the importance of modeling the variation observed in human-generated referring expressions (Viethen and Dale, 2010; Viethen et al., 2013; van Deemter et al., 2012b; Mitchell et al., 2013), and some approaches have applied machine-learning techniques to single-object references (Viethen and Dale, 2010; Mitchell et al., 2011a,b). Recently, Mitchell et al. (2013) introduced a probabilistic approach for conjunctive descriptions of single objects, which will provide a comparison baseline for experiments in Section 8. To the best of our knowledge, this paper presents the first learned probabilistic model for referring expressions defining sets, and is the first effort to treat REG as a density estimation problem.

REG is related to *content selection*, which has been studied for generating text from databases (Konstas and Lapata, 2012), event streams (Chen et al., 2010), images (Berg et al., 2012; Zitnick and Parikh, 2013), and text (Barzilay and Lapata, 2005; Carenini et al., 2006). However, most approaches to this problem output bags of concepts, while we construct full logical expressions, allowing our approach to capture complex relations between attributes.

Finally, our approach to modeling meaning using lambda calculus is related to a number of approaches that used similar logical representation in various domains, including database query interfaces (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005, 2007), natural language instructions (Chen and Mooney, 2011; Matuszek et al., 2012b; Kim and Mooney, 2012; Artzi and Zettlemoyer, 2013b), event streams (Liang et al., 2009; Chen et al., 2010), and visual descriptions (Matuszek et al., 2012a; Krishnamurthy and Kollar, 2013). Our use of logical forms follows this line of work, while extending it to handle plurality and coordination, as described in Section 4.1. In addition, lambda calculus was shown to enable effective natural language generation from logical forms (White and Rajkumar, 2009; Lu and Ng, 2011). If combined with these approaches, our approach would allow the creation of a complete REG pipeline.

3 Technical Overview

Task Let $\mathcal{Z}$ be a set of logical expressions that select a target set of objects G in a world state S, as formally defined in Section 5.1. We aim to learn a probability distribution $P(z \mid S, G)$, with $z \in \mathcal{Z}$.

For example, in the referring expressions domain we work with, the state $S = \{o_1, \ldots, o_n\}$ is a set of n objects o_i. Each o_i has three properties: color, shape and type. The target set $G \subseteq S$ is the subset of objects to be described. Figure 1a shows an example scene. The world state S includes the 11 objects in the image, where each object is assigned color (*yellow, green* ...), shape (*cube, cylinder* ...) and type (*broccoli, apple* ...). The target set G contains the circled objects. Our task is to predict a distribution which closely matches the empirical distribution in Figure 1c.

Model and Inference We model $P(z|S, G)$ as a globally normalized log-linear model, using features of the logical form z, and its execution with respect to S and G. Since enumerating all $z \in \mathcal{Z}$ is intractable, we develop an approximate inference algorithm which constructs a high quality candidate set, using a learned pruning model. Section 5.2 describes the globally scored log-linear model. Section 5.3 presents a detailed description of the inference procedure.

Learning We use stochastic gradient descent to learn both the global scoring model and the explicit pruning model, as described in section 6. Our data consists of human-generated referring expressions, gathered from Amazon Mechanical Turk. These sentences are automatically labelled with logical forms with a learned semantic parser, providing a stand-in for manually labeled data (see Section 7).

Evaluation Our goal is to output a distribution that closely matches the distribution that would be produced by humans. We therefore evaluate our model with gold standard labeling of crowd-sourced referring expressions, which are treated as samples from the implicit distribution we are trying to model. The data and evaluation procedure are described in Section 7. The results are presented in Section 8.

4 Modeling Referring Expressions

4.1 Semantic Modeling

Our semantic modeling approach uses simply-typed lambda-calculus following previous work (Carpenter, 1997; Zettlemoyer and Collins, 2005; Artzi and Zettlemoyer, 2013b), extending it in one important way: we treat *sets* of objects as a primitive type, rather than individuals. This allows us to model plurality, cardinality, and coordination for the language observed in our data, and is further motivated by recent cognitive science evidence that sets and their properties are represented as single units in human cognition (Scontras et al., 2012).

Plurals Traditionally, noun phrases are identified with the entity-type $\mathtt{e}$ and pick out individual objects (Carpenter, 1997). This makes it difficult to interpret plural noun-phrases which pick out a set of objects, like "The red cubes". Previous approaches would map this sentence to the same logical expression as the singular "The red cube", ignoring the semantic distinction encoded by the plural.

Instead, we define the primitive entity $\mathtt{e}$ to range over sets of objects. $\langle \mathtt{e}, \mathtt{t} \rangle$-type expressions are therefore functions from sets to a truth-value. These are used in two ways, modeling both *distributive* and *collective* predicates (cf. Stone, 2000):

1. *Distributive* predicates are $\langle \mathtt{e}, \mathtt{t} \rangle$-type expressions which will return $true$ if every individual in the set has a given property. For example, the expression $\lambda x.red(x)$ will be true for all sets which contain only objects for which the value red is *true*.
2. *Collective* predicates are $\langle \mathtt{e}, \mathtt{t} \rangle$-type expressions which indicate a property of the set itself. For example, in the phrase "the two cubes", "two" corresponds to the expression $\lambda x.cardinality_2(x)$ which will return $true$ only for sets which have exactly two members.

We define semantic plurality in terms of two special *collective* predicates: sg for singular and plu for plural. For examples, "cube" is interpreted as $\lambda x.cube(x) \wedge sg(x)$, whereas "cubes" is interpreted as $\lambda x.cube(x) \wedge plu(x)$. The sg predicate returns $true$ only for singleton sets. The plu predicate returns $true$ for sets that contain two or more objects.

We also model three kinds of determiners, functional-type $\langle \langle \mathtt{e}, \mathtt{t} \rangle, \mathtt{e} \rangle$-type expressions which select a single set from the power-set represented by their $\langle \mathtt{e}, \mathtt{t} \rangle$-type argument. The definite determiner "the" is modeled with the predicate ι, which resolves to the maximal set amongst those licensed by its argument. The determiner $Every$ only accepts $\langle \mathtt{e}, \mathtt{t} \rangle$-type arguments that define singleton sets (i.e. the argument includes the sg predicate) and returns a set containing the union of these singletons. For example, although "red cube" is a singular expression, "Every red cube" refers to a set. Finally, the indefinite determiner "a" is modeled with the logical constant $\mathcal{A}$, which picks a singleton set by implicitly introducing an existential quantifier (Artzi and Zettlemoyer, 2013b).[2]

Coordination Two types of coordination are prominent in set descriptions. The first is attribute coordination, which is typically modeled with the boolean operators: $\wedge$ for conjunction and $\vee$ for disjunction. For example, the phrase "the red cubes and green rectangle" involves a disjunction that joins two conjunctive expressions, both within the scope of the definite determiner: $\iota(\lambda x.(red(x) \wedge cube(x) \wedge plu(x)) \vee (green(x) \wedge rectangle(x) \wedge sg(x)))$.

[2] This treatment of the indefinite determiner is related to generalized skolem terms as described by Steedman (2011).

The second kind of coordination, a new addition of this work, occurs when two sets are coordinated. This can either be set union ($\cup$) as in the phrase "The cubes and the rectangle" ($\iota(\lambda x.cube(x) \wedge plu(x)) \cup \iota(\lambda x.rectangle(x) \wedge sg(x)))$), or set difference ($\setminus$) as in the phrase "All blocks except the green cube": ($\iota(\lambda x.object(x) \wedge plu(x)) \setminus \iota(\lambda x.green(x) \wedge cube(x) \wedge sg(x)))$.

4.2 Visual Domain

Objects in our scenes are labeled with attribute values for four attribute types: `color` (7 values, such as *red*, *green*), `shape` (9 values, such as *cube*, *sphere*), `type` (16 values, such as *broccoli*, *apple*) and a special `object` property, which is *true* for all objects. The special `object` property captures the role of descriptions that are true for all objects, such as "toy" or "item". Each of these 33 attribute values corresponds to an $\langle \mathtt{e}, \mathtt{t}\rangle$-type predicate.

5 Model and Inference

In this section, we describe our approach to modeling the probability $P(z \mid S, G)$ of a logical form $z \in \mathcal{Z}$ that names a set of objects G in a world S, as defined in Section 3. We first define $\mathcal{Z}$ (Section 5.1), and then present the distribution (Section 5.2) and an approximate inference approach that makes use of a learned pruning model (Section 5.3).

5.1 Space of Possible Meanings

The set $\mathcal{Z}$ defines logical expressions that we will consider for picking the target set G in state S. In general, we can construct infinitely many such expressions. For example, every $z \in \mathcal{Z}$ can be trivially extended to form a new candidate z' for $\mathcal{Z}$ by adding a true clause to any conjunct it contains. However, the vast majority of such expressions are overly complex and redundant, and would never be used in practice as a referring expression.

To avoid this explosion, we limit the type and complexity of the logical expressions that are included in $\mathcal{Z}$. We consider only $\mathtt{e}$-type expressions, since they name sets, and furthermore only include expressions that name the desired target set G.[3] We also limit the overall complexity of each $z \in \mathcal{Z}$, to contain not more than M logical constants.

- $p : \langle\langle \mathtt{e},\mathtt{t}\rangle,\mathtt{e}\rangle, e_1 : \langle \mathtt{e},\mathtt{t}\rangle \rightarrow p(e_1) : \mathtt{e}$

$$\text{e.g.}\ \begin{array}{|l|}\hline p = \iota : \langle\langle \mathtt{e},\mathtt{t}\rangle,\mathtt{e}\rangle \\ e_1 = \lambda x.cube(x) \wedge sg(x) : \langle \mathtt{e},\mathtt{t}\rangle \\ \hline \iota(\lambda x.cube(x) \wedge sg(x)) : \mathtt{e} \\ \hline \end{array}$$

- $p : \langle \mathtt{t},\mathtt{t}\rangle, e_1 : \langle \mathtt{e},\mathtt{t}\rangle \rightarrow \lambda x.p(e_1(x)) : \langle \mathtt{e},\mathtt{t}\rangle$

$$\text{e.g.}\ \begin{array}{|l|}\hline p = \neg : \langle \mathtt{t},\mathtt{t}\rangle \\ e_1 = \lambda x.red(x) : \langle \mathtt{e},\mathtt{t}\rangle \\ \hline \lambda x.\neg(red(x)) : \langle \mathtt{e},\mathtt{t}\rangle \\ \hline \end{array}$$

- $p : \langle \mathtt{e}, \langle \mathtt{e},\mathtt{t}\rangle\rangle, e_1 : \mathtt{e} \rightarrow \lambda x.(p(x))(e_1)$

$$\text{e.g.}\ \begin{array}{|l|}\hline p = equal : \langle \mathtt{e}, \langle \mathtt{e},\mathtt{t}\rangle\rangle \\ e_1 = \mathcal{A}(\lambda y.cube(y) \wedge sg(y)) : \mathtt{e} \\ \hline \lambda x.equal(x, \mathcal{A}(\lambda y.cube(y) \wedge sg(y))) \\ \hline \end{array}$$

- $p : \langle \mathtt{e}, \langle \mathtt{e},\mathtt{e}\rangle\rangle, e_1 : \mathtt{e}, e_2 : \mathtt{e} \rightarrow (p(e_1))(e_2) : \mathtt{e}$

$$\text{e.g.}\ \begin{array}{|l|}\hline p = \setminus : \langle \mathtt{e}, \langle \mathtt{e},\mathtt{e}\rangle\rangle \\ e_1 = \iota(\lambda x.cube(x) \wedge plu(x)) : \mathtt{e} \\ e_2 = Every(\lambda x.object(x) \wedge sg(x)) : \mathtt{e} \\ \hline Every(\lambda x.object(x) \wedge sg(x)) \setminus \\ \quad \iota(\lambda x.cube(x) \wedge plu(x)) : \mathtt{e} \\ \hline \end{array}$$

- $p : \langle \mathtt{t}, \langle \mathtt{t},\mathtt{t}\rangle\rangle, e_1 : \langle \mathtt{e},\mathtt{t}\rangle, e_2 \langle \mathtt{e},\mathtt{t}\rangle \rightarrow \lambda x.(p(e_1(x)))(e_2(x)) : \langle \mathtt{e},\mathtt{t}\rangle$

$$\text{e.g.}\ \begin{array}{|l|}\hline p = \wedge : \langle \mathtt{t}, \langle \mathtt{t},\mathtt{t}\rangle\rangle \\ e_1 = \lambda x.red(x) : \langle \mathtt{e},\mathtt{t}\rangle \\ e_2 = \lambda x.cube(x) : \langle \mathtt{e},\mathtt{t}\rangle \\ \hline \lambda x.red(x) \wedge cube(x) : \mathtt{e} \\ \hline \end{array}$$

Figure 2: The five rules used during generation. Each rule is a template which takes a predicate $p : t$ of type t and one or two arguments $e_i : t_i$, with type t_i. The output is the logical expression after the arrow $\rightarrow$, constructed using the inputs as shown.

To achieve these constraints, we define an inductive procedure for enumerating $\mathcal{Z}$, in order of complexity. We first define $\mathcal{A}_j$ to be the set of all $\mathtt{e}$- and $\langle \mathtt{e}, \mathtt{t}\rangle$-type expressions that contain exactly j logical constants. Figure 2 presents five rules that can be used to construct $\mathcal{A}_j$ by induction, for $j = 1, \ldots, \infty$, by repeatedly adding new constants to expressions in $\mathcal{A}_{j'}$ for $j' < j$. Intuitively, $\mathcal{A}_j$ is the set of all complexity j expressions that can be used as subexpressions for higher complexity entires in our final set $\mathcal{Z}$. Next, we define $\mathcal{Z}_j$ to be the $\mathtt{e}$-type expressions in $\mathcal{A}_j$ that name the correct set G. And, finally, $\mathcal{Z} = \cup_{j=1 \ldots M} \mathcal{Z}_j$ of all correct expressions up to a maximum complexity of M.

This construction allows for a finite $\mathcal{Z}$ with good

[3] We do not attempt to model underspecified or otherwise incorrect expressions, although our model could handle this by considering all $\mathtt{e}$-type expressions.

empirical coverage, as we will see in the experiments in Section 8. However, $\mathcal{Z}$ is still prohibitively large for the maximum complexities used in practise (for example $M = 20$). Section 5.3 presents an approach for learning models to prune $\mathcal{Z}$, while still achieving good empirical coverage.

5.2 Global Model

Given a finite $\mathcal{Z}$, we can now define our desired globally normalized log-linear model, conditioned on the state S and set of target objects G:

$$P_G(z \mid S, G; \theta) = \frac{1}{C} e^{\theta \cdot \phi(z,S,G)} \tag{1}$$

where $\theta \in \mathbb{R}^n$ is a parameter vector, $\phi(z, S, G) \in \mathbb{R}^n$ is a feature function and C is the normalization constant. Section 5.4 defines the features we use.

5.3 Pruning $\mathcal{Z}$

As motivated in Section 5.1, the key challenge for our global model in Equation 1 is that the set $\mathcal{Z}$ is too large to be explicitly enumerated. Instead, we designed an approach for learning to approximate $\mathcal{Z}$ with a subset of the highly likely entries, and use this subset as a proxy for $\mathcal{Z}$ during inference.

More specifically, we define a binary distribution that is used to classify whether each $a \in \mathcal{A}_j$ is likely to be used as a sub-expression in $\mathcal{Z}$, and prune each $\mathcal{A}_j$ to keep only the top k most likely entries. This distribution is a logistic regression model:

$$P_j(a \mid S, G; \pi_j) = \frac{e^{\pi_j \cdot \phi(a,S,G)}}{1 + e^{\pi_j \cdot \phi(a,S,G)}} \tag{2}$$

with features $\phi(a, S, G) \in \mathbb{R}^n$ and parameters $\pi_j \in \mathbb{R}^n$. This distribution uses the same features as the global model presented in Equation 1, which we describe in Section 5.4.

Together, the pruning model and global model define the distribution $\hat{P}(z \mid G, S; \theta, \Pi)$ over $z \in \mathcal{Z}$, conditioned on the world state S and target set G, and parameterized by both the parameters θ of the global model and the parameters $\Pi = \{\pi_1, \ldots, \pi_M\}$ of the pruning models.

5.4 Features

We use three kinds of features: logical expression structure features, situated features and a complexity feature. All features but the complexity feature are shared between the global model in Equation 1 and the pruning model in Equation 2. In order to avoid overly specific features, the attribute value predicates in the logical expressions are replaced with their attribute type (ie. $red \rightarrow color$). In addition, the special constants sg and plu are ignored when computing features.

In the following description of our features, all examples are computed for the logical expression $\iota(\lambda x.red(x) \wedge object(x) \wedge plu(x))$, with respect to the scene and target set in Figure 1a.

Structure Features We use binary features that account for the presence of certain structures in the logical form, allowing the model to learn common usage patterns.

- **Head Predicate** - indicator for use of a logical constant as a head predicate in every subexpression of the expression. A head predicate is the top-level operator of an expression. For example, the head predicate of the expression "$\lambda x.red(x) \wedge object(x)$" is "$\wedge$" and the head of $\lambda x.red(x)$ is red. For our running example, the head features are ι, $\wedge$, $color$, $object$.

- **Head-Predicate Bigrams and Trigrams** - head-predicate bigrams are defined to be the head predicate of a logical form, and the head predicate of one of its children. Trigrams are similarly defined. E.g. bigrams: $[\iota, \wedge]$, $[\wedge, color]$, $[\wedge, object]$, and trigrams: $[\iota, \wedge, red]$, $[\iota, \wedge, object]$.

- **Conjunction Duplicate** - this feature fires if a conjunctive expression contains duplicate subexpressions amongst its children.

- **Coordination Children** - this feature set indicates the presence of a coordination subexpression ($\wedge$, $\vee$, $\cup$ or $\setminus$) and the head expressions of all pairs and triples of its child expressions. E.g. $[\wedge; red, object]$.

Situated Features These features take into account the evaluation of the logical form z with respect to the state S and target set G. They capture common patterns between the target set G and the object groups named by subexpressions of z.

- **Head Predicate and Coverage** - this feature set indicates the head predicate of every sub-expression of the logical form, combined with a comparison between the execution of the sub-expression and the target set G. The possible values for this comparison (which we call the "coverage" of the expression with respect to G) are: `EQUAL`, `SUBSET` (`SUB`), `SUPERSET` (`SPR`), `DISJOINT`, `ALL`, `EMPTY` and `OTHER`. E.g. $[\iota, \texttt{SUB}]$, $[\wedge, \texttt{SUB}]$, $[\mathit{color}, \texttt{SUB}]$, $[\mathit{object}, \texttt{ALL}]$

- **Coordination Child Coverage** - this feature set indicates the head-predicate of a coordination subexpression, combined with the coverage of all pairs and triples of its child expressions. E.g. $[\wedge; \texttt{SUB}, \texttt{ALL}]$.

- **Coordination Child Relative Coverage** - this feature set indicates, for every pair of child subexpressions of coordination expressions in the logical form, the coverage of the child subexpressions relative to each other. The possible relative coverage values are: `SUB-SPR`, `DISJOINT`, `OTHER`. E.g. $[\wedge; \texttt{SUB-SPR}]$.

Complexity Features We use a single real-numbered feature to account for the complexity of the logical form. We define the complexity of a logical form to be the number of logical constants used. Our running example has a complexity of 4. This feature is only used in the global model, since the pruning model always considers logical expressions of fixed complexity.

6 Learning

Figure 3 presents the complete learning algorithm. The algorithm is online, using stochastic gradient descent updates for both the globally scored density estimation model and the learned pruning model. The algorithm assumes a dataset of the form $\{(Z_i, S_i, G_i) : i = 1 \dots n\}$ where each example scene includes a list of logical expressions Z_i, a world state S_i, and a target set of objects, G_i, which will be identified by the resulting logical expressions. The output is learned parameters for both the globally scored density estimation model θ, and for the learned pruning models Π.

Inputs: Training set $\{(Z_i, S_i, G_i) : i = 1 \dots n\}$, where Z_i is a list of logical forms, S_i is a world state, and G_i is a target set of objects. Number of iterations T. Learning rate α_0. Decay parameter c. Complexity threshold M, as described in Section 5.3.

Definitions: Let $\hat{P}(z \mid G_i, S_i; \theta, \Pi)$ be the predicted global probability from Equation 1. Let $\hat{P}_j(z \mid G_i, S_i; \pi_j)$ be the predicted pruning probability from Equation 2. Let $\hat{\mathcal{A}}_j$ be the set of all complexity-M logical expressions, after pruning (see Section 5.1). Let $SUB(j, z)$ be all complexity-j sub-expressions of logical expression z. Let $Q_i(z \mid S_i, G_i)$ be the empirical probability over $z \in \mathcal{Z}$, estimated from Z_i. Finally, let $\phi_i(z)$ be a shorthand for the feature function $\phi(z, S_i, G_i)$ as defined in Section 5.4.

Algorithm:

```
Initialize $\theta \leftarrow \vec{0}$, $\pi_j \leftarrow \vec{0}$ for $j = 1 \dots M$
For $t = 1 \dots T, i = 1 \dots n$:
Step 1: (Update Global Model)
  a. Compute the stochastic gradient:
     $\Delta_\theta \leftarrow E_{Q_i(z|S_i,G_i)}[\phi_i(z)] - E_{\hat{P}(z|G_i,S_i;\theta,\Pi)}[\phi_i(z)]$
  b. Update the parameters:
     $\gamma \leftarrow \frac{\alpha_0}{1+c\times\tau}$ where $\tau = i + t \times n$
     $\theta \leftarrow \theta + \gamma\Delta_\theta$
Step 2: (Update Pruning Model)
  For $j = 1 \dots M$
    a. Construct a set of positive and negative examples:
       $\mathcal{D}^+ \leftarrow \bigcup_{z \in Z_i} SUB(j, z)$.
       $\mathcal{D}^- \leftarrow \hat{\mathcal{A}}_j \setminus \mathcal{D}^+$
    b. Compute mini-batch stochastic gradient, normalizing
       for data skew:
       $\Delta_{\pi_j} \leftarrow \frac{1}{|\mathcal{D}^+|}\sum_{z\in\mathcal{D}^+}(1 - P_j(z \mid S_i, G_i; \pi_j))\phi_i(z)$
              $- \frac{1}{|\mathcal{D}^-|}\sum_{z\in\mathcal{D}^-} P_j(z \mid S_i, G; \pi_j)\phi_i(z)$
    c. Update complexity-$j$ pruning parameters:
       $\pi_j \leftarrow \pi_j + \gamma\Delta_{\pi_j}$
```

Output: θ and $\Pi = [\pi_1, \dots, \pi_M]$

Figure 3: The learning algorithm.

6.1 Global Model Updates

The parameters θ of the globally scored density estimation model are trained to maximize the log-likelihood of the data:

$$O_i = \log \prod_{z \in Z_i} P_G(z \mid S_i, G_i) \tag{3}$$

Taking the derivative of this objective with respect to θ yields the gradient in Step 1a of Figure 3. The marginals, $E_{\hat{P}(z|G_i,S_i;\theta,\Pi)}(\phi_i(z))$, are computed over the approximate finite subset of $\mathcal{Z}$ constructed with the inference procedure described in Section 5.3.

6.2 Pruning Model Updates

To update each of the M pruning models, we first construct a set of positive and negative examples (Step 2a). The positive examples, $\mathcal{D}^+$, include those sub-expressions which should be in the beam - these are all complexity j sub-expressions of logical expressions in Z_i. The negative examples, $\mathcal{D}^-$, include all complexity-j expressions constructed during beam search, minus those which are in $\mathcal{D}^+$. The gradient (Set 2b) is a binary mini-batch gradient, normalized to correct for data skew.

7 Experimental Setup

Data Collection Our dataset consists of 118 images, taken with a Microsoft Kinect camera. These are the same images used by Matuszek et al. (2012a), but we create multiple prompts for each image by circling different objects, giving 269 scenes in total. These scenes were shown to workers on Amazon Mechanical Turk[4] who were asked to imagine giving instructions to a robot and complete the sentence "Please pick up _____" in reference to the circled objects. Twenty referring expressions were collected for each scene, a total of 5380 expressions.

From this data, 43 scenes (860 expressions) were held-out for use in a test set. Of the remaining scenes, the sentences of 30 were labeled with logical forms. 10 of these scenes (200 expressions) are used as a labeled initialization set, and 20 are used as a development test set (400 expressions). A small number of expressions ($\sim$5%) from the labeled initial set were discarded, either because they did not correctly name the target set, or because they used very rare attributes (such as texture, or location) to name the target objects.

Surrogate Labeling To avoid hand labeling the large majority of the scenes, we label the data with a learned semantic parser (Zettlemoyer and Collins, 2005). We created a hand-made lexicon for the entire training set, which greatly simplifies the learning problem, and learned the parameters of the parsing moder on the 10-scene initialization set. The weights were then further tuned using semi-supervised techniques (Artzi and Zettlemoyer, 2011, 2013b) on the data to be labeled. Testing on the development set shows that this parser achieves roughly 95% precision and 70% recall.

Using this parser, we label the sentences in our training set. We only use scenes where at least 15 sentences were successfully parsed. This gives a training set of 141 scenes (2587 expressions). Combining the automatically labeled training set with the hand-labelled initialization, development and held-out data, our labelled corpus totals 3938 labeled expressions. By contrast, the popular TUNA furniture sub corpus (Gatt et al., 2007) contains 856 descriptions of 20 scenes, and although some of these refer to sets, these sets contain two objects at most.

Framework Our experiments were implemented using the University of Washington Semantic Parsing Framework (Artzi and Zettlemoyer, 2013a).

Hyperparameters Our inference procedure requires two hyperparameters: M, the maximum complexity threshold, and k, the beam size. In practice, we set these to the highest possible values which still allow for training to complete in a reasonable amount of time (under 12 hours). M is set to 20, which is sufficient to cover 99.5% of the observed expressions. The beam-size k is 100 for the first three complexity levels, and 50 thereafter.

For learning, we use the following hyperparameters, which were tuned on the development set: learning rate $\alpha_0 = .25$, decay rate $c = .02$, number of epochs $T = 10$.

Evaluation Metrics Evaluation metrics used in REG research have assumed a system that produces a single output. Our goal is to achieve a distribution over logical forms that closely matches the distribution observed from human subjects. Therefore, we compare our learned model to the labeled test data with mean absolute error:

$$MAE = \frac{1}{2n} \sum_{i=1}^{n} \sum_{z \in \mathcal{Z}} |P(z \mid S_i, G_i) - Q(z \mid S_i, G_i)|$$

where Q is the empirical distribution observed in the training data. MAE measures the total probability mass which is assigned differently in the predicted distribution than in the empirical distribution. We use MAE as opposed to KL divergence or data likelihood as both of these measures are uninformative when the support of the two distributions differ.

[4] http://www.mturk.com

–	MAE	$\%_{dup}$	$\%_{uniq}$	Top1
VOA	39.7	98.2	92.5	**72.7**
GenX	**25.8** (5.0)	**100** (0)	**100** (0)	**72.7** (0)

Table 1: Single object referring expression generation results. Our approach (GenX) is compared to the approach from Mitchell et al. (2013) (VOA). Standard deviation over five shuffles of training set is reported in parentheses.

This metric is quite strict; small differences in the estimated probabilities over a large number of logical expressions can result in a large error, even if the relative ordering is quite similar. Therefore, we report the percentage of observed logical expressions which the model produces, either giving credit multiple times for duplicates ($\%_{dup}$) or counting each unique logical expression in a scene once ($\%_{uniq}$). Put another way, $\%_{dup}$ counts logical expression tokens, whereas $\%_{uniq}$ counts types. We also report the proportion of scenes where the most likely logical expression according to the model matched the most common one in the data (Top1).

Single Object Baseline In order to compare our method against the state of the art for generating referring expressions for single objects, we use the subset of our corpus where the target set is a single object. This sub-corpus consists of 44 scenes for training and 11 held out for testing.

For comparison we re-implemented the probabilistic Visual Objects Algorithm (VOA) of Mitchell et al. (2013). We refer the readers to the original paper for details of the approach. The parameters of the model were tuned on the training data: the prior likelihood estimates for each of the four attribute types (α_{att}) were estimated as the relative frequency of each attribute in the data. We pick the ordering of attributes and the length penalty, λ, from the cross-product of all possible $4!$ orderings and all integers on the range of [1,10], choosing the setting which results in the lowest average absolute error (AAE) on the training set. This process resulted in the following parameter settings: $\alpha_{color} = .916$, $\alpha_{shape} = .586$, $\alpha_{type} = .094$, $\alpha_{object} = .506$, AP ordering = $[type, shape, object, color]$, $\lambda = 4$. Inference was done using 10,000 samples per scene.

–	MAE	$\%_{dup}$	$\%_{uniq}$	Top1
Full GenX	**54.3** (4.5)	**87.4** (0.6)	**72.9** (1.1)	**52.6** (8.3)
NoPrune	71.8 (2.5)	42.2 (2.7)	16.1 (1.7)	40.0 (5.0)
NoCOV	87.0 (6.7)	26.0 (3.7)	11.2 (2.1)	14.9 (9.7)
NoSTRUC	60.2 (1.7)	79.6 (0.3)	61.9 (0.5)	44.6 (4.5)
HeadExpOnly	88.8 (6.4)	21.9 (8.6)	9.3 (3.5)	14.0 (7.9)

Table 2: Results on the complete corpus for the complete system (Full GenX), ablating the pruning model (NoPrune) and the different features: without coverage features (NoCOV), without structure features (NoSTRUC) and using only the logical expression HeadExp features (HeadExpOnly). Standard deviation over five runs is shown in parentheses.

8 Results

We report results on both the single-object subset of our data and the full dataset. Since our approach is online, and therefore sensitive to data ordering, we average results over 5 different runs with randomly ordered data, and report variance.

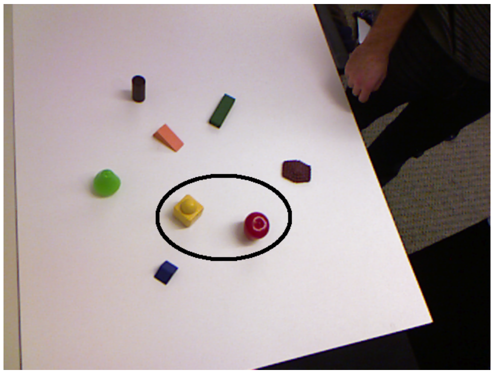
Single Objects Table 1 shows the different metrics for generating referring expression for single objects only. Our approach outperforms VOA (Mitchell et al., 2013) on all metrics, including an average of approximately 35% relative reduction in MAE. In addition, unlike VOA, our system (GenX) produces every logical expression used to refer to single objects in our dataset, including a small number which use negation and equality.

Object Sets Table 2 lists results on the full dataset. Our learned pruning approach produces an average 72.9% of the unique logical expressions used present in our dataset – over 87% when these counts are weighted by their frequency. The globally scored model achieves a mean absolute error of 54.3, and correctly assigns the highest probability to the most likely expression over 52% of the time.

Also shown in Table 2 are results obtained when elements of our approach are ablated. Using the global model for pruning instead of an explicitly trained model causes a large drop in performance, demonstrating that our global model is inappropri-

Q	$\hat{P}$	z
.750	.320	$\iota(\lambda x.object(x) \wedge (yellow(x) \vee red(x)))$
	.114	$\iota(\lambda x.lego(x)) \cup \iota(\lambda x.red(x) \wedge apple(x))$
	.114	$\iota(\lambda x.yellow(x) \wedge lego(x))) \cup \iota(\lambda x.apple(x))$
	.044	$\iota(\lambda x.lego(x) \vee (red(x) \wedge apple(x)))$
	.044	$\iota(\lambda x.(yellow(x) \wedge lego(x)) \vee apple(x))$
	.036	$\iota(\lambda x.lego(x)) \cup \iota(\lambda x.red(x) \wedge sphere(x))$
	.026	$\iota(\lambda x.red(x) \wedge lego(x)) \cup \iota(\lambda x.red(x) \wedge sphere(x))$
.050	.021	$\iota(\lambda x.(lego(x) \wedge yellow(x)) \vee (red(x) \wedge apple(x)))$
	.017	$\iota(\lambda x.(lego(x) \wedge yellow(x)) \vee (red(x) \wedge sphere(x)))$
	.014	$\iota(\lambda x.yellow(x) \wedge lego(x)) \cup \iota(\lambda x.red(x) \wedge sphere(x))$
.100	.010	$\iota(\lambda x.yellow(x) \wedge object(x)) \cup \iota(\lambda x.apple(x))$
.050	.007	$\iota(\lambda x.yellow(x) \wedge object(x)) \cup \iota(\lambda x.red(x) \wedge sphere(x))$
.050	.005	$\iota(\lambda x.yellow(x) \wedge object(x)) \cup \iota(\lambda x.red(x) \wedge object(x))$

(a)



(b)

Figure 4: Example output of our system for the scene on the right. We show the top 10 expressions (z) from the predicted distribution ($\hat{P}$) compared to the empirical distribution estimated from our labeled data (Q). The bottom section shows the predicted probability of the three expressions which were not in the top 10 of the predicted distribution. Although the mean absolute error (MAE) of $\hat{P}$ and Q is 63.8, $\hat{P}$ covers all of the entries in Q in the correct relative order and also fills in many other plausible candidates.

ate for pruning. We also ablate subsets of our features, demonstrating that the coverage and structural features are both crucial for performance.

Qualitatively, we found the learned distributions were often higher quality than the seemingly high mean absolute error would imply. Figure 4 shows an example output where the absolute error of the predicted distribution was 63.8. Much of the error can be attributed to probability mass assigned to logical expressions which, although not observed in our test data, are reasonable referring expressions. This might be due to the fact that our estimate of the empirical distribution comes from a fairly small sample (20), or other factors which we do not model that make these expressions less likely.

9 Conclusion

In this paper, we modeled REG as a density-estimation problem. We demonstrated that we can learn to produce distributions over logical referring expressions using a globally normalized model. Key to the approach was the use of a learned pruning model to define the space of logical expression that are explicitly enumerated during inference. Experiments demonstrate state-of-the-art performance on single object reference and the first results for learning to name sets of objects, correctly recovering over 87% of the observed logical forms.

This approach suggests several directions for future work. Lambda-calculus meaning representations can be designed for many semantic phenomena, such as spatial relations, superlatives, and graded properties, that are not common in our data. Collecting new datasets would allow us to study the extent to which the approach would scale to domains with such phenomena.

Although the focus of this paper is on REG, the approach is also applicable to learning distributions over logical meaning representations for many other tasks. Such learned models could provide a range of possible inputs for systems that map logical expressions to sentences (White and Rajkumar, 2009; Lu and Ng, 2011), and could also provide a valuable prior on the logical forms constructed by semantic parsers in grounded settings (Artzi and Zettlemoyer, 2013b; Matuszek et al., 2012a).

Acknowledgements

This research was supported in part by the Intel Science and Technology Center for Pervasive Computing, by DARPA under the DEFT program through the AFRL (FA8750-13-2-0019) and the CSSG (N11AP20020), the ARO (W911NF-12-1-0197), and the NSF (IIS-1115966). The authors wish to thank Margaret Mitchell, Mark Yatskar, Anthony Fader, Kenton Lee, Eunsol Choi, Gabriel Schubiner, Leila Zilles, Adrienne Wang, and the anonymous reviewers for their helpful comments.

References

Areces, C., Koller, A., and Striegnitz, K. (2008). Referring expressions as formulas of description logic. In *Proceedings of the International Natural Language Generation Conference*.

Artzi, Y. and Zettlemoyer, L. (2011). Bootstrapping semantic parsers from conversations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Artzi, Y. and Zettlemoyer, L. (2013a). UW SPF: The University of Washington Semantic Parsing Framework.

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

Barzilay, R. and Lapata, M. (2005). Collective content selection for concept-to-text generation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Berg, A. C., Berg, T. L., Daume, H., Dodge, J., Goyal, A., Han, X., Mensch, A., Mitchell, M., Sood, A., Stratos, K., et al. (2012). Understanding and predicting importance in images. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Carenini, G., Ng, R. T., and Pauls, A. (2006). Multi-document summarization of evaluative text. In *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics*.

Carpenter, B. (1997). *Type-Logical Semantics*. The MIT Press.

Chen, D., Kim, J., and Mooney, R. (2010). Training a multilingual sportscaster: using perceptual context to learn language. *Journal of Artificial Intelligence Research*, 37(1):397–436.

Chen, D. and Mooney, R. (2011). Learning to interpret natural language navigation instructions from observations. In *Proceedings of the National Conference on Artificial Intelligence*.

Dale, R. and Reiter, E. (1995). Computational interpretations of the gricean maxims in the generation of referring expressions. *Cognitive Science*, 19:233–264.

Dale, R. and Reiter, E. (2000). *Building natural language generation systems*. Cambridge University Press.

Gardent, C. (2002). Generating minimal definite descriptions. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Gatt, A. and van Deemter, K. (2007). Incremental generation of plural descriptions: Similarity and partitioning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Gatt, A., Van Der Sluis, I., and Van Deemter, K. (2007). Evaluating algorithms for the generation of referring expressions using a balanced corpus. In *Proceedings of the European Workshop on Natural Language Generation*.

Grice, H. P. (1975). Logic and conversation. *1975*, pages 41–58.

Horacek, H. (2004). On referring to sets of objects naturally. In *Natural Language Generation*, pages 70–79. Springer.

Kim, J. and Mooney, R. J. (2012). Unsupervised PCFG induction for grounded language learning with highly ambiguous supervision. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Konstas, I. and Lapata, M. (2012). Unsupervised concept-to-text generation with hypergraphs. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*.

Krahmer, E. and van Deemter, K. (2012). Computational generation of referring expressions: A survey. *Computational Linguistics*, 38(1):173–218.

Krishnamurthy, J. and Kollar, T. (2013). Jointly learning to parse and perceive: Connecting natural language to the physical world. *Transactions of the Association for Computational Linguistics*, 1(2):193–206.

Liang, P., Jordan, M., and Klein, D. (2009). Learning semantic correspondences with less supervision. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Lu, W. and Ng, H. T. (2011). A probabilistic forest-to-string model for language generation from typed lambda calculus expressions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Matuszek, C., FitzGerald, N., Zettlemoyer, L., Bo, L., and Fox, D. (2012a). A joint model of language and perception for grounded attribute learning. *Proceedings of the International Conference on Machine Learning*.

Matuszek, C., Herbst, E., Zettlemoyer, L. S., and Fox, D. (2012b). Learning to parse natural language commands to a robot control system. In *Proceedings of the International Symposium on Experimental Robotics*.

Mitchell, M., van Deemter, K., and Reiter, E. (2011a). Applying machine learning to the choice of size modifiers. In *Proceedings of the PRE-CogSci Workshop*.

Mitchell, M., Van Deemter, K., and Reiter, E. (2011b). Two approaches for generating size modifiers. In *Proceedings of the European Workshop on Natural Language Generation*.

Mitchell, M., van Deemter, K., and Reiter, E. (2013). Generating expressions that refer to visible objects. In *Proceedings of Conference of the North American Chapter of the Association for Computational Linguistics*.

Ren, Y., Van Deemter, K., and Pan, J. Z. (2010). Charting the potential of description logic for the generation of referring expressions. In *Proceedings of the International Natural Language Generation Conference*.

Scontras, G., Graff, P., and Goodman, N. D. (2012). Comparing pluralities. *Cognition*, 123(1):190–197.

Steedman, M. (2011). *Taking Scope*. The MIT Press.

Stone, M. (2000). On identifying sets. In *Proceedings of the International Conference on Natural Language Generation*.

van Deemter, K. (2002). Generating referring expressions: Boolean extensions of the incremental algorithm. *Computational Linguistics*, 28:37–52.

van Deemter, K., Gatt, A., Sluis, I. v. d., and Power, R. (2012a). Generation of referring expressions: Assessing the incremental algorithm. *Cognitive Science*, 36(5):799–836.

van Deemter, K., Gatt, A., van Gompel, R. P., and Krahmer, E. (2012b). Toward a computational psycholinguistics of reference production. *Topics in Cognitive Science*, 4(2):166–183.

Viethen, J. and Dale, R. (2010). Speaker-dependent variation in content selection for referring expression generation. In *Proceedings of the Australasian Language Technology Workshop*.

Viethen, J., Mitchell, M., and Krahmer, E. (2013). Graphs and spatial relations in the generation of referring expressions. In *Proceedings of the European Workshop on Natural Language Generation*.

White, M. and Rajkumar, R. (2009). Perceptron reranking for ccg realization. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Winograd, T. (1972). Understanding natural language. *Cognitive Psychology*, 3(1):1–191.

Zelle, J. and Mooney, R. (1996). Learning to parse database queries using inductive logic programming. In *Proceedings of the National Conference on Artificial Intelligence*.

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

Zettlemoyer, L. and Collins, M. (2007). Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Zitnick, C. L. and Parikh, D. (2013). Bringing semantics into focus using visual abstraction. In *IEEE Conference on Computer Vision and Pattern Recognition*.