Proceedings of SALT 30: 000–000, 2020

Montague Grammar Induction*

Gene Louis Kim
University of Rochester

Aaron Steven White
University of Rochester

Abstract We propose a computational modeling framework for inducing combinatory categorial grammars from arbitrary behavioral data. This framework provides the analyst fine-grained control over the assumptions that the induced grammar should conform to: (i) what the primitive types are; (ii) how complex types are constructed; (iii) what set of combinators can be used to combine types; and (iv) whether (and to what) the types of some lexical items should be fixed. In a proof-of-concept experiment, we deploy our framework for use in distributional analysis. We focus on the relationship between s(emantic)-selection and c(ategory)-selection, using as input a lexicon-scale acceptability judgment dataset focused on English verbs' syntactic distribution (the MegaAcceptability dataset) and enforcing standard assumptions from the semantics literature on the induced grammar.

Keywords: grammar induction, combinatory categorial grammar, semantic selection, experimental semantics, computational semantics, experimental syntax, computational syntax

1 Introduction

Semantic theories aim to capture two kinds of facts about languages' expressions: (i) their distributional characteristics; and (ii) their inferential affordances. The descriptive adequacy of any such theory is evaluated in terms of its coverage of these facts—an evaluation that is commonly carried out informally on a relatively small number of test cases. While this approach to theory-building and evaluation has yielded deep insights, it also carries significant risks: generalizations that appear unassailable based on a small number of high-frequency examples (and theories built on them) can collapse when evaluated on a more diverse range of expressions purportedly covered by the generalization (see White accepted for recent discussion).

Ameliorating these risks requires developing scalable methods for building and evaluating semantic theories. Until recently, a major obstacle to developing such methods was that sufficiently large-scale behavioral datasets were not available. This

* We are grateful to the audience at SALT 30 for discussion of this work. Funding was provided by NSF-BCS-1748969 (*The MegaAttitude Project: Investigating selection and polysemy at the scale of the lexicon*) and NSF EAGER grant IIS-1908595 (*Learning a High-Fidelity Semantic Parser*).

situation has changed with the advent of lexicon-scale acceptability and inference judgment datasets, such as the MegaAttitude datasets (White & Rawlins 2016, 2018, accepted; White, Rudinger, Rawlins & Van Durme 2018; An & White 2020; Moon & White 2020). Concomitant advances in computational modeling have cleared the way for automating distributional and inferential analysis for the purposes of theory-building and evaluation. The remaining challenge is one of integration. On the one hand, powerful methods for learning structured representations from corpus data and (to some extent) behavioral data now exist (Le & Zuidema 2014, 2015; Williams, Drozdov & Bowman 2018; Shen, Lin, Huang & Courville 2018; Kim, Rush, Yu, Kuncoro, Dyer & Melis 2019; Drozdov, Verga, Chen, Iyyer & McCallum 2019a; Drozdov, Verga, Yadav, Iyyer & McCallum 2019b), but the relationship between these models' representations and grammars posited under standard frameworks (Montague 1973 *et seq*) assuming some form of (combinatory) categorial grammar (Steedman 2000) remains unclear.[1] On the other hand, powerful methods for inducing such grammars now exist (Zettlemoyer & Collins 2005, 2007, 2009; Kwiatkowksi, Zettlemoyer, Goldwater & Steedman 2010; Kwiatkowski, Zettlemoyer, Goldwater & Steedman 2011; Bisk & Hockenmaier 2012a,b, 2013, 2015), but they do not straightforwardly generalize to the full range of behavioral data of interest in experimental semantics.

To address these limitations, we propose a general deep learning-based, computational modeling framework for inducing full-fledged combinatory categorial grammars from multiple distinct types of behavioral data. Beyond providing the ability to synthesize arbitrary distributional and inferential data within a single model, our framework provides the analyst fine-grained control over the assumptions that the induced grammar should conform to: (i) what the primitive types are (e.g. e, s, t, etc.); (ii) how complex types are constructed (e.g. that $\langle t_1, t_2 \rangle$ is a type if t_1 and t_2 are types); (iii) what set of combinators can be used to combine types (e.g. application, composition, etc.); and (iv) whether (and to what) the types of some lexical items should be fixed. As a proof of concept, we deploy our framework for use in distributional analysis. We focus, in particular, on the relationship between s(emantic)-selection and c(ategory)-selection, using as input a lexicon-scale acceptability judgment dataset focused on English verbs' syntactic distribution (the MegaAcceptability dataset; White & Rawlins 2016, accepted) and enforcing standard assumptions from the semantics literature on the induced grammar. As a case study, we analyze the typing that the induced grammar infers for clausal complements. Clausal complements are useful in this regard, since their syntactic complexity provides a rigorous test of our framework's ability to recover interpretable types.

1 Other systems *assume* prior knowledge of some, often quite exquisite, structure (see Baroni, Bernardi & Zamparelli 2014 and references therein). We focus here on systems that *learn* said structure.

We begin with some brief background on the deep learning-based approach to grammar induction that we build on (§2). We then describe how we extend that approach to learn full-fledge combinatory categorial grammars (§3) before turning to our proof-of-concept experiments with the MegaAcceptability dataset (§4) and the results of these experiments (§5). We conclude with a discussion of future directions for our framework (§6).

2 Deep learning-based approaches to grammar induction

Our framework fits within the broader context of *grammar induction* (or *grammatical inference*; see Heinz & Sempere 2016 for recent reviews)—in particular, distributional learning-based approaches (see Clark & Yoshinaka 2016 for a recent review). The goal of a grammar induction system is to produce a grammar for some language based on a dataset containing (at least) well-formed expressions of that language. These datasets may simply be (multi)sets or sequences of such expressions or they may further associate labels with expressions or sequences thereof—e.g. associating a (possibly malformed) expression with its acceptability or a pair of expressions with a label indicating whether the first entails the second. We focus specifically on *supervised* grammar induction, wherein some labeling of expressions is assumed, since most behavioral datasets fit this description.

The grammars (or weightings thereon) output by such a system can take a variety of forms. Here, we build on grammar induction systems that learn to encode both expressions and grammars in a vector space, treating the problem as one of training the parameters of a machine learning model to predict the labels in some dataset—e.g. to predict acceptability or entailment. A major benefit of using such a system is that it is straightforward to train on arbitrary behavioral data (see Potts 2019 for further discussion): given a way of mapping expressions to vectors, those vectors can be input to standard regression models from the experimental literature and off-the-shelf optimization routines employed to jointly train the grammar induction system and regression model (see Goldberg 2017 for a technical review).

We specifically build on recently developed methods that generalize the *inside-outside algorithm* (Baker 1979) for training probabilistic syntactic parsers to vector space syntactic parsers. These methods recursively construct vector representations of expressions aimed at capturing their syntactic features and/or semantic properties. These methods have two interlocking components: (i) a generalization of the *inside algorithm*, which constructs a vector representation for an expression based on the vector representations of its subexpressions; and (ii) a generalization of the *outside algorithm*, which constructs a vector respresentation for an expression based on the vector representations of surrounding expressions (Le & Zuidema 2014, 2015; Drozdov et al. 2019a,b). The former can be thought of very roughly as implementing

a parameterized version of Minimalism's MERGE (or similar "bottom-up" operations); and the latter can be thought of (again, very roughly) as implementing a parameterized version of Minimalism's AGREE (or similar "top-down" operations).

Both the inside and outside algorithms compute probabilities relative to a string and a probabilistic context free grammar (see Manning & Schütze 1999 for an in-depth discussion). But for understanding the relevance of these algorithms here, it is possible to ignore the probabilistic aspects, starting by assuming a vanilla context free grammar (CFG) in Chomsky Normal Form—i.e. with rules of the form $A \rightarrow B\,C$ or $A \rightarrow w$, where A, B, and C nonterminals and w a terminal—and then generalizing.

In this non-probabilistic context, the inside algorithm is analogous to the CKY algorithm for recognition and parsing (Younger 1967): given a grammar $\mathcal{G}$ and a sentence S consisting of a sequence of words $w_0 ... w_{|S|-1}$, the algorithm computes, for each subsequence of words $w_{i:j} = w_i ... w_{j-1}$ in S (where $0 \leq i < j \leq |S|$), the set N_{ij} of nonterminals that could yield $w_{i:j}$—i.e. the nonterminals for which some sequence of rewrites would result in $w_{i:j}$. These sets are computed recursively: let $N_{i(i+1)} \equiv \text{UNARY}_{\mathcal{G}}(w_i)$ for $0 \leq i < |S|$ and $N_{ij} \equiv \bigcup_{k=i+1}^{j-1} \text{COMBINE}_{\mathcal{G}}(N_{ik}, N_{kj})$ for $0 \leq i < j-1 < |S|$, where $\text{UNARY}_{\mathcal{G}}(w) = \{A \mid (A \rightarrow w) \in \mathcal{G}\}$ and $\text{COMBINE}_{\mathcal{G}}(N, N') = \bigcup_{\langle B,C\rangle \in N \times N'} \{A \mid (A \rightarrow B\,C) \in \mathcal{G}\}$ (see Shieber, Schabes & Pereira 1995).

Defined in this way, N_{ij} specifies what kinds of constituent $w_{i:j}$ could be (e.g. a noun phrase, a verb phrase, etc.), ignoring the rest of the sentence. The non-probabilistic analogue of the outside algorithm computes O_{ij}: what kinds of constituent $w_{i:j}$ could be considering only the parts of the sentence outside $w_{i:j}$. The intersection $N_{ij} \cap O_{ij}$ then indicates all and only the kinds of constituents $w_{i:j}$ could be in the context of the entire sentence. Like N_{ij}, O_{ij} is computed recursively: assume a fixed $O_{0|S|}$ and let $O_{ij} \equiv \bigcup_{m=j+1}^{|S|} \text{SPLIT}^R_{\mathcal{G}}(O_{im}, N_{jm}) \cup \bigcup_{m=0}^{i-1} \text{SPLIT}^L_{\mathcal{G}}(O_{mj}, N_{mi})$ for $0 \leq i < j-1 < |S|$, where $\text{SPLIT}^R_{\mathcal{G}}(O, N) = \bigcup_{\langle A,C\rangle \in O \times N} \{B \mid (A \rightarrow B\,C) \in \mathcal{G}\}$ and $\text{SPLIT}^L_{\mathcal{G}}(O, N) = \bigcup_{\langle A,B\rangle \in O \times N} \{C \mid (A \rightarrow B\,C) \in \mathcal{G}\}$.

Importantly for our purposes, this algorithm does not require the grammar to be a CFG or even a probabilistic generalization thereof. For example, with straightforward modifications to UNARY, COMBINE, SPLIT^L, and SPLIT^R, it can be adapted to arbitrary combinatory categorial grammars (CCGs) and therefore the sorts of type grammars familiar in semantic theory. UNARY returns the set of types listed for (the denotation of) a word; and given some set of combinators $\mathcal{C}$—e.g. APPLICATION, COMPOSITION, etc.—$\text{COMBINE}_G(N, N') = \bigcup_{\langle t,t'\rangle \in N \times N'} \{c(t, t') \mid c \in \mathcal{C} \wedge \langle t, t'\rangle \in \text{dom}(c)\}$ (with SPLIT^L and SPLIT^R analogously defined). This possibility becomes important for our framework, discussed in detail in §3.

The generalization of these algorithms to vector spaces involves four main changes: (i) substituting terminals with vectors in $\mathcal{V}_{\text{lex}} = \mathbb{R}^{M_{\text{lex}}}$ and sets N_{ij}, O_{ij} with

vectors in $\mathcal{V}_{\text{node}} = \mathbb{R}^{M_{\text{node}}}$; (ii) substituting the function UNARY with a parameterized function from $\mathcal{V}_{\text{lex}}$ to $\mathcal{V}_{\text{node}}$ and the functions COMBINE, SPLIT^R, and SPLIT^L with parameterized functions from $\mathcal{V}^2_{\text{node}}$ to $\mathcal{V}_{\text{node}}$; (iii) substituting unions with weighted vector sums; and (iv) substituting intersections and products with vector concatenations.[2] The analogue to the symbolic grammar $\mathcal{G}$ is then the mapping from terminals to vectors and the parameterization of these functions.

More specifically, we define *inside vectors* $\mathbf{h}^{\triangle}_{ij}$, analogous to N_{ij}, and *outside vectors* $\mathbf{h}^{\triangledown}_{ij}$, analogous to O_{ij}. Assuming some mapping from $w_0 \ldots w_{|S|-1}$ to a sequence of vectors $\mathbf{x}_0 \ldots \mathbf{x}_{|S|-1}$ and that $\mathbf{h}^{\triangle}_{i(i+1)} = \text{UNARY}(\mathbf{x}_i)$ for $0 \le i < |S|$:

$$\mathbf{h}^{\triangle}_{ij} = \sum_{k=i+1}^{j-1} \alpha_{ijk} \text{COMBINE}\left(\mathbf{h}^{\triangle}_{ik} \oplus \mathbf{h}^{\triangle}_{kj}\right) \text{ where } \boldsymbol{\alpha}_{ij} = \text{ATTEND}\left(\begin{bmatrix} \mathbf{h}^{\triangle}_{i(i+1)} \oplus \mathbf{h}^{\triangle}_{(i+1)j} \\ \cdots \\ \mathbf{h}^{\triangle}_{i(j-1)} \oplus \mathbf{h}^{\triangle}_{(j-1)j} \end{bmatrix}\right)$$

where $\oplus$ denotes vector concatenation and $\boldsymbol{\alpha}_{ij}$ is a vector of non-negative weights such that $\sum_{k=i+1}^{j-1} \alpha_{ijk} = 1$.[3] These weights can be thought of as probabilities, serving to indicate which subexpressions $w_{i:k}$ and $w_{k:j}$ likely form the expression $w_{i:j}$.

The outside vectors are analogously defined:

$$\mathbf{h}^{\triangledown}_{ij} = \sum_{m=j+1}^{|S|} \alpha_{imj} \text{SPLIT}^R\left(\mathbf{h}^{\triangledown}_{im} \oplus \mathbf{h}^{\triangle}_{jm}\right) + \sum_{m=0}^{i-1} \alpha_{mji} \text{SPLIT}^L\left(\mathbf{h}^{\triangledown}_{mj} \oplus \mathbf{h}^{\triangle}_{mi}\right)$$

On analogy with $N_{ij} \cap O_{ij}$ providing a complete picture of $w_{i:j}$ in the context of S, we then take the concatenation $\mathbf{h}^{\diamond}_{ij} = \mathbf{h}^{\triangle}_{ij} \oplus \mathbf{h}^{\triangledown}_{ij}$ as the representation for $w_{i:j}$, using it to predict arbitrary labels on $w_{i:j}$. In §4, we describe how we train the parameters of UNARY, COMBINE, SPLIT^R, and SPLIT^L to predict the acceptability of S. But our aim is to go beyond merely predicting acceptability: we furthermore aim to induce a coherent symbolic typing on all nodes.

3 Integrating symbolic types

Building on the model described in §2, we propose a framework for jointly inferring a sentence's syntactic structure and a coherent mapping from that syntactic structure

2 Unless otherwise specified, we implement parameterized functions as multi-layer perceptrons with a single LeakyReLU hidden layer of the same size as the output (Maas, Hannun & Ng 2013): $\text{MLP}(\mathbf{x}) = \mathbf{W}_1 \text{LeakyReLU}(\mathbf{W}_2\mathbf{x} + \mathbf{b}_2) + \mathbf{b}_1$, where the $\mathbf{W}$s and $\mathbf{b}$s are the parameters to be optimized.

3 ATTEND is implemented with *additive attention* (Bahdanau, Cho & Bengio 2015), a standard deep-learning method of modeling weighted judgments with parameterized functions, $\text{ATTEND}(\mathbf{X}) = \sigma(\mathbf{v}^\top \tanh(\mathbf{X}\mathbf{W}))$, where σ is the softmax function, $\sigma(\mathbf{x}) = \left[\frac{e^{x_1}}{\Sigma_j e^{x_j}}, \frac{e^{x_2}}{\Sigma_j e^{x_j}}, \ldots\right]$.

to semantic types from arbitrary behavioral data. To the vector space syntactic parser described in §2, we add two components: (i) a vector space *interpretation function* that implements $[\![\cdot]\!]$ as a parameterized function INTERPRET from the syntactic parser space $\mathcal{V}_{\text{node}}$ to a denotation space $\mathcal{V}_{\text{interp}} = \mathbb{R}^{M_{\text{interp}}}$; and (ii) a vector space *type grammar* that takes vector representations produced by the interpretation function and returns the associated symbolic type (or a distribution thereon).

The high-level idea behind our framework is to train the parser, the interpretation function, and the type grammar to produce an assignment of types to expressions that is coherent in the sense that some type (possibly, but not necessarily, a specific type, like $\langle \mathtt{s}, \mathtt{t} \rangle$) can be derived for the entire sentence from the types assigned to its subexpressions. We first describe the type grammar (§3.1) and then describe how we enforce type coherence during training (§3.2). Throughout, we assume a standard recursive definition for the set of types $\mathcal{T}$, given primitive types $\mathcal{P}$—e.g. $\mathtt{e}$, $\mathtt{s}$, and $\mathtt{t}$.

$$\mathcal{T}_0 \equiv \mathcal{P} \qquad \mathcal{T}_i \equiv \left[\bigcup_{j=0}^{i-1} \mathcal{T}_j\right] \times \mathcal{D} \times \left[\bigcup_{j=0}^{i-1} \mathcal{T}_j\right] \qquad \mathcal{T} \equiv \bigcup_{i=0}^{\infty} \mathcal{T}_i$$

where the set of type constructors $\mathcal{D}$ is a singleton here, but could be straightforwardly extended—e.g. for representing directed types, as in standard and modal syntactic CCG (Steedman 2000; Baldridge 2002), or more exotic types, as in generative lexicon theory (Asher 2011; Pustejovsky 2013; Asher & Pustejovsky 2013).

Throughout this section, we focus only on structural aspects of our framework, specifying how its components hang together. We defer discussion of how we train the parameters of each component to carry out its intended function to §4.

3.1 Type grammar

The type grammar consists of four components: (i) a set of *primitive type embeddings* $\mathbf{x}_t$ for primitive types $t \in \mathcal{P}$ and *type constructor embeddings* $\mathbf{x}_d$ for $d \in \mathcal{D}$; (ii) a *type encoder* (§3.1.1), which maps symbolic types $t \in \mathcal{T}$ to their vector space *type embeddings* $\boldsymbol{\tau}_t$; (iii) *type decoders* (§3.1.2), which implement vector space combinators and map from the type embeddings corresponding to the combinator arguments to (a distribution over) the resulting types; and (iv) a *combinatory controller* (§3.1.3), which determines which combinator to use for combining a pair of types.[4]

4 Our actual implementation of the type encoder uses a stacked binary tree long-short term memory network (binary tree bi-LSTMs; Le & Zuidema 2015; Tai, Socher & Manning 2015; Miwa & Bansal 2016) and the implementation of the type decoders uses a unidirectional variant. We give a somewhat simplified treatment below that ignores bidirectionality, stacking, and the distinction between hidden states and cell states because they are not relevant to understanding the core ideas.

3.1.1 Type encoder

The type encoder views types as binary trees—e.g. $\langle \mathtt{e}, \langle \mathtt{e}, \mathtt{t} \rangle \rangle$ and $\langle \langle \mathtt{e}, \mathtt{e} \rangle, \mathtt{t} \rangle$ are viewed as right-branching and left-branching trees, respectively, both with leaves e, e, and t. It recursively maps these binary type trees to type embeddings by defining the embedding $\boldsymbol{\tau}_t$ of complex type $t = [_d\, t_0\, t_1]$ in terms of embeddings of its (possibly complex) constituent types t_0 and t_1 and the type constructor d using parameterized functions WRAP and CONSTRUCT.

$$\boldsymbol{\tau}_t = \begin{cases} \text{WRAP}\left(\mathbf{x}_t\right) & \text{if } t \in \mathcal{P} \\ \text{CONSTRUCT}\left(\mathbf{x}_d, \boldsymbol{\tau}_{t_0} \oplus \boldsymbol{\tau}_{t_1}\right) & \text{if } t = [_d\, t_0\, t_1] \end{cases}$$

3.1.2 Type decoders

Type decoders implement combinators of M arguments (here, M is always 1 or 2) by taking M type embeddings and outputting a probability distribution over output types. For instance, a type decoder implementing the identity combinator should take $\boldsymbol{\tau}_t$ and yield a probability distribution that assigns a probability to t of approximately 1, while a type decoder implementing the application combinator should take $\boldsymbol{\tau}_{\langle t_0, t_1 \rangle}$ and $\boldsymbol{\tau}_{t_0}$ and yield a probability distribution that assigns a probability to t_1 of approximately 1. The reason that we need these combinators to produce probability distributions rather than types, is that multiple types may be compatible with a certain expression—e.g. because it is ambiguous—and so, when we apply these decoders to the vector space interpetation of an expression, we need to allow for multiple types—even if, when applied to a type embedding, we only want the decoder to yield the single correct type.

Type decoders can be thought of as the reverse of encoders, "reading" a distribution over symbolic types off of a vector space embedding. Each decoder consists of three parameterized functions: (i) STRUCTURE, which determines the probability that the decoder produces a primitive type v. a complex type; (ii) PRIMITIVE, which determines which primitive type to select if the decoder chooses to produce a primitive type; and (iii) FACTOR, which determines how to update the decoder's state based on its previous state (by "factoring out" the previous decision to recurse from the state).[5] The start state of the decoder is always the input type embedding(s).

For ease of exposition, we describe how a single type would be sampled from the distribution defined by STRUCTURE, PRIMITIVE, FACTOR, and the input type embedding(s), though it is also possible to determine the probability of particular types as well as the most likely types using related procedures. Starting with the input embedding(s) $\boldsymbol{\tau}$, the SAMPLE($\boldsymbol{\tau}$) procedure chooses to generate a complex type

5 When $\mathcal{D}$ is a non-singleton, a fourth parameterized function is required for selecting the constructor.

with probability $\theta_{\text{complex}} = \text{STRUCTURE}(\boldsymbol{\tau})$ and a primitive type with probability $1 - \theta_{\text{complex}}$. If it chooses to generate a primitive type, it chooses that primitive type based on primitive type probabilities $\boldsymbol{\theta}_{\text{primitive}} = \text{PRIMITIVE}(\boldsymbol{\tau})$ and returns it; otherwise, it returns $\langle \text{SAMPLE}(\boldsymbol{\tau}_0'), \text{SAMPLE}(\boldsymbol{\tau}_1') \rangle$, where $\langle \boldsymbol{\tau}_0', \boldsymbol{\tau}_1' \rangle = \text{FACTOR}(\boldsymbol{\tau})$.

3.1.3 Combinatory controller

When types are represented symbolically, it is straightforward to determine whether they are in the domain of a combinator and, if not, whether type raising one of the types—i.e. mapping $t \in \mathcal{T}$ to $\langle\langle t, t' \rangle, t' \rangle$ for some $t' \in \mathcal{T}$—would make them so. When types are represented as vectors, this is not the case (except in the case of the identity combinator, which can apply to any type embedding).

To deal with this issue, we need some way of selecting (i) which combinator to use for decoding the input; (ii) whether to type-raise one of the children; and (iii) if a particular child is to be raised, which type to raise that child to. In addition, we need to define how raising is carried out in a vector space.

These decisions are made based on a set of *combinatory action types* $\mathcal{A}$, which are triples specifying which combinator to use and whether to raise the left type, right type or neither, as well as two parameterized functions: (i) ACTION, which selects an action type from $\mathcal{A}$ for $\boldsymbol{\tau}$ and $\boldsymbol{\tau}'$ based on $\boldsymbol{\phi}^{(\text{action})} = \text{ACTION}\left(\boldsymbol{\tau}, \boldsymbol{\tau}'\right)$; and (ii) RAISE, which selects a type with which to raise $\boldsymbol{\tau}$ with probability $\boldsymbol{\phi}^{(raise)} = \text{RAISE}(\boldsymbol{\tau})$ (e.g. if t is selected to raise $\boldsymbol{\tau}_{\mathsf{e}}$, $\boldsymbol{\tau}_{\langle\langle \mathsf{e},\mathsf{t}\rangle,\mathsf{t}\rangle}$ should result).[6] To implement type raising of $\boldsymbol{\tau}_t$ with t', we run the type encoder on a *type embedding tree* $[_{\mathbf{x}_d}[_{\mathbf{x}_d} \boldsymbol{\tau}\ \boldsymbol{\tau}_{t'}]\ \boldsymbol{\tau}_{t'}]$ corresponding to the raised type, as though $\boldsymbol{\tau}_t$ were a primitive type embedding.

3.2 Optimizing type coherence

In order to learn an interpretation function from the syntactic parser space to the denotation space that produces coherent types, we minimize what we term a *type coherence loss* between the type assigned to the interpretation of an expression by the identity combinator with the type produced by combining its children using the combinatory action selected by the combinatory controller. To compute the type coherence loss for the vector space interpretation $\boldsymbol{\lambda}_{ij} = \text{INTERPRET}(\mathbf{h}_{ij}^{\diamond})$ of expression $w_{i:j}$ and the vector space interpretations $\boldsymbol{\lambda}_{ik} = \text{INTERPRET}(\mathbf{h}_{ik}^{\diamond})$ and $\boldsymbol{\lambda}_{kj} = \text{INTERPRET}(\mathbf{h}_{kj}^{\diamond})$ (with $i < k < j$) for a pair of subexpressions $\langle w_{i:k}, w_{k:j} \rangle$

6 We only consider raising with primitive types. There are multiple way one might extend vector space type raising to complex types. We take this issue to be somewhat unimportant: it is unclear whether there is a real need for vector space type raising, since the same result might already be achievable by the model by positing ambiguity.

of $w_{i:j}$, we compute the probability distribution over types $\mathbb{P}^{(\text{parent})}$ produced by applying the identity decoder to $\boldsymbol{\lambda}_{ij}$ and the probability distributions over types $\mathbb{P}_a^{(\text{children})}$ produced by conducting combinatory action $a \in \mathcal{A}$ on $\boldsymbol{\lambda}_{ik}$ and $\boldsymbol{\lambda}_{kj}$. The agreement between $\mathbb{P}^{(\text{parent})}$ and $\mathbb{P}_a^{(\text{children})}$ is computed using cross-entropy H.[7] The contribution to this agreement score from action a is then weighted by the action probabilities $\boldsymbol{\phi}_{ijk}^{(\text{action})} = \text{ACTION}(\boldsymbol{\lambda}_{ik}, \boldsymbol{\lambda}_{kj})$ assigned by the combinatory controller.

$$\begin{aligned}\mathcal{L}_{\text{type}}^{(\text{pair})}(i,j,k) &= \sum_{a \in \mathcal{A}} \phi_{ijka}^{(\text{action})} H\left(\mathbb{P}^{(\text{parent})}(\cdot \mid \boldsymbol{\lambda}_{ij}), \mathbb{P}_a^{(\text{children})}(\cdot \mid \boldsymbol{\lambda}_{ik}, \boldsymbol{\lambda}_{kj})\right) \\ &= -\sum_{a \in \mathcal{A}} \phi_{ijka}^{(\text{action})} \sum_{t \in \mathcal{T}} \mathbb{P}^{(\text{parent})}(t \mid \boldsymbol{\lambda}_{ij}) \log \mathbb{P}_a^{(\text{children})}(t \mid \boldsymbol{\lambda}_{ik}, \boldsymbol{\lambda}_{kj})\end{aligned}$$

This value is small when $\mathbb{P}_a^{(\text{children})}$ assigns probabilities to types that are similar to those assigned by $\mathbb{P}^{(\text{parent})}$. The type coherence for an expression $w_{i:j}$ is then computed by summing over all possible pairs of subexpressions $\langle w_{i:k}, w_{k:j} \rangle$, weighting by the likelihood α_{ijk} that the syntactic parser assigns to that particular pair: $\mathcal{L}_{\text{type}}^{(\text{expr})}(i,j) = \sum_{k=i+1}^{j-1} \alpha_{ijk} \mathcal{L}_{\text{type}}(i,j,k)$. This value is small when the type coherence loss for high probability pairs is small. Finally, the type coherence loss for an entire sentence is computed by summing over the type coherence loss for all expressions: $\mathcal{L}_{\text{type}}^{(\text{sent})} = \sum_{i=0}^{|S|-2} \sum_{j=i+2}^{|S|} \mathcal{L}_{\text{type}}(i,j)$. This value is small when the type coherence loss for all expressions is small, and thus, our aim is to find parameterizations for the syntactic parser, type grammar, and interpretation function that drive $\mathcal{L}_{\text{type}}^{(\text{sent})}$ down.

4 Deploying the framework

As a proof-of-concept experiment using our framework, we fit a model that simultaneously aims to predict sentence acceptability and find a coherent typing for the expressions contained in those sentences. Our framework allows us to specify the that grammar has an arbitrary number of primitive types $\mathcal{P}$ and type constructors $\mathcal{D}$ as well as arbitrary combinators $\mathcal{C}$. For these experiments, we follow standard treatments building on Montague 1973 in assuming three primitive types (e, s, and t) and a single type constructor. For the set of combinators $\mathcal{C}$, we assume (undirected) application—where $\text{APPLY}(t_0, \langle t_0, t_1 \rangle) = \text{APPLY}(\langle t_0, t_1 \rangle, t_0) = t_1$—and (first-order undirected) composition—where $\text{COMPOSE}(\langle t_0, t_1 \rangle, \langle t_1, t_2 \rangle) = \text{COMPOSE}(\langle t_1, t_2 \rangle, \langle t_0, t_1 \rangle) = \langle t_0, t_2 \rangle$. We first describe the acceptability dataset that we train the model on (§4.1) and then discuss how we train the model to simultaneously predict acceptability and coherent types (§4.2).

7 We approximate this value by computing the probability distribution over type trees of at most a certain depth (here, depth 4). Alternative methods for comparing probability distributions—e.g. Kullback-Leibler divergence—could be used in place of cross-entropy.

4.1 The MegaAcceptability dataset

We train our model to predict the acceptability judgments in the MegaAcceptability dataset (White & Rawlins 2016, accepted). This dataset contains acceptability judgments for 1,000 clause-embedding verbs—including a variety of cognitive verbs (e.g. *believe*, *forget*, *doubt*), communicative verbs (e.g. *say*, *claim*, *explain*), emotive verbs (e.g. *upset*, *disgust*, *anger*), among many other classes—in 50 different syntactic frames. These frames include simple intransitives (NP ___), transitives (NP ___ NP), and ditransitives (NP ___ NP NP) as well as a large variety of frames including subordinate clauses either with or without a direct object or prepositional phrase (always headed by *to*). These subordinate clauses vary in terms of their complementizer (∅, *that*, *for*, *whether*, and constituent question), the presence of an embedded subject, and embedded tense (past, future, tenseless, *to*-infinitival, bare infinitival, and present participial). (See White & Rawlins for a full list of frames.)

A sentence is constructed for each verb-frame pair by instantiating all lexical category words with bleached terms to minimize the effects of lexical idiosyncrasies in the acceptability judgments. All noun phrases are instantiated as either *someone* or *something*, all untensed verb phrases as either *do something* or *have something* and tensed verb phrases as *happened*, and sentences as *something (would) happen(ed)*.

(1) a. *believe* + NP ___ *that* S → Someone believed that something happened.
b. *ask* + NP ___ *whether* S → Someone asked whether something happened.
c. *force* + NP ___ NP to VP → Someone forced someone to do something.
d. *tell* + NP ___ NP *which thing* to VP → Someone told sm. which thing to do.

Each sentence is rated by 5 participants on a 1-7 scale. We use the normalized variant of these judgments provided by White & Rawlins (accepted), which adds seven verbs not in the original data and maps each sentence to a single real-valued acceptability value in a way that accounts for annotator bias and annotator quality.

4.2 Training the model

Our model is trained using the Adam optimizer in three stages: (i) syntactic parser training, (ii) type grammar training, and (iii) interpreter training.

4.2.1 Syntactic parser training

The parameters of the syntactic parser—i.e. the parameters of UNARY, COMBINE, SPLIT^R, and SPLIT^L discussed in §2—are trained to predict the normalized acceptability judgments found in MegaAcceptability, minimizing the mean-squared error (as in standard linear regression). The RoBERTa system (Liu, Ott, Goyal, Du,

Joshi, Chen, Levy, Lewis, Zettlemoyer & Stoyanov 2019) is used to map $w_0 \ldots w_{|S|-1}$ to input vectors $\mathbf{x}_0 \ldots \mathbf{x}_{|S|-1}$. The acceptability prediction is done by passing the vector representation $\mathbf{h}^{\diamond}_{0|S|}$ for the entire sentence into a parameterized function ACCEPTABILITY, whose parameters are jointly trained with those of the parser.

4.2.2 Type grammar training

The parameters of the type grammar—i.e. the parameters of the encoder's WRAP and CONSTRUCT, the decoders' STRUCTURE, PRIMITIVE, and FACTOR, and the combinatory controller's ACTION and RAISE—are trained in three stages.

Encoder and identity decoder. First, the type encoder and identity combinator are trained as an *autoencoder*—i.e. so that the probability distribution over types produced by encoding t as $\boldsymbol{\tau}_t$ then applying the identity combinator to $\boldsymbol{\tau}_t$ assigns high probability to t. This training is carried out on randomly sampled types.

Application and composition decoders. The remaining combinators in the grammar are similarly trained. For each combinator, pairs of types are randomly sampled and automatically processed so that they are viable inputs to the corresponding combinator—e.g. ensuring that only pairs of the form $\langle t_0, t_1 \rangle$ and t_0 (or the reverse) are input to application and only pairs of the form $\langle t_0, t_1 \rangle$ and $\langle t_1, t_2 \rangle$ (or the reverse) are input to composition. To train each combinator, the correct outputs for each of these pairs is first determined symbolically based on the combinator—e.g. for application, the output for the pair $\langle t_0, t_1 \rangle$ and t_0 (or the reverse) is t_1—then each element of the pair is encoded using the type encoder. The parameters of the combinator are then trained to produce a high probability for the correct output when applied to the concatenation of the encoded pair.

Combinatory controller. The combinatory controller is trained with a similar sampling procedure to the decoder training. The sampling procedure is extended so that when processing the type pairs into viable inputs, we include the possibility of type-raising one of the inputs with respect to a primitive type before combinator application. Viable combinator action types are symbolically determined for each input pair—e.g. $\langle t_0, t_1 \rangle$ and t_0 could be combined with application and no type raising or with application and type raising t_0 to $\langle \langle t_0, t_1 \rangle, t_1 \rangle$. The parameters of the combinatory controller are then trained to produce a high probability for viable combinatory action types when applied to the concatenation of the input pair.

4.2.3 Interpreter training

The parameters of INTERPRET are trained to minimize the type coherence loss $\mathcal{L}^{(\text{sent})}_{\text{type}}$ (§3.2) and a *type constraint loss*. The type constraint loss encodes standard assumptions about the types that particular lexical items map to. Here, we enforce

that sentence denotations have proposition types $\langle \mathtt{s}, \mathtt{t} \rangle$, quantificational noun phrase denotations (*someone*, *something*) have quantifier types $\langle \langle \mathtt{e}, \langle \mathtt{s}, \mathtt{t} \rangle \rangle, \langle \mathtt{s}, \mathtt{t} \rangle \rangle$, and verb phrases *do something*, *have something*, *happen*, and *happened* denote properties of individuals $\langle \mathtt{e}, \langle \mathtt{s}, \mathtt{t} \rangle \rangle$.[8] To enforce these constraints, we train the interpretation function so that applying the identity combinator to the denotation embedding vector $\boldsymbol{\lambda}_{ij}$ of a span matching one of these expressions assigns high probability to the corresponding type. This training procedure is only applied to the subset of the data where the normalized acceptability value is in the 90th percentile or greater, so that we do not attempt to derive $\langle \mathtt{s}, \mathtt{t} \rangle$ for sentences that should not have such a type. This threshold is fairly stringent—the least acceptable sentence above the threshold is (2)—and likely removes acceptable sentences.

(2) Someone distrusted someone that something happened.

Removing acceptable sentences is fine for our purposes, since it is more important to avoid forcing the model to produce coherent types for unacceptable sentences.

4.3 Predicting types

To find the most likely type for each subexpression of a sentence, we use a probabilistic parsing algorithm based on the standard supertag-factored A* CCG parsing algorithm (Lewis & Steedman 2014; Lewis, Lee & Zettlemoyer 2016). The specific details of this algorithm are not relevant for current purposes.

5 Results

We will first take a look at how well the parser does in predicting acceptability and explore some of the syntactic representations it learns. This is mainly to show that the parser is learning something that could reasonably be thought of as a syntactic representation. Then, we will dive into the types that are learned by our model, drawing two contrasts: (i) what the model infers for both declarative and interrogative complement clauses and (ii) what it infers for finite and infinitival complements.

5.1 Acceptability

We use *k*-fold cross-validation to evaluate our model's ability to predict the normalized acceptability of sentences it has not seen. In cross-validation (see Hastie,

8 Without these constraints, we found that the model would learn a trivial grammar—e.g. every constituent denotes $\langle \mathtt{t}, \mathtt{t} \rangle$ and the combinatory controller assigns composition a high probability. Crucially, we do not enforce that particular clausal complements decode to particular types, since we aim to induce those types.

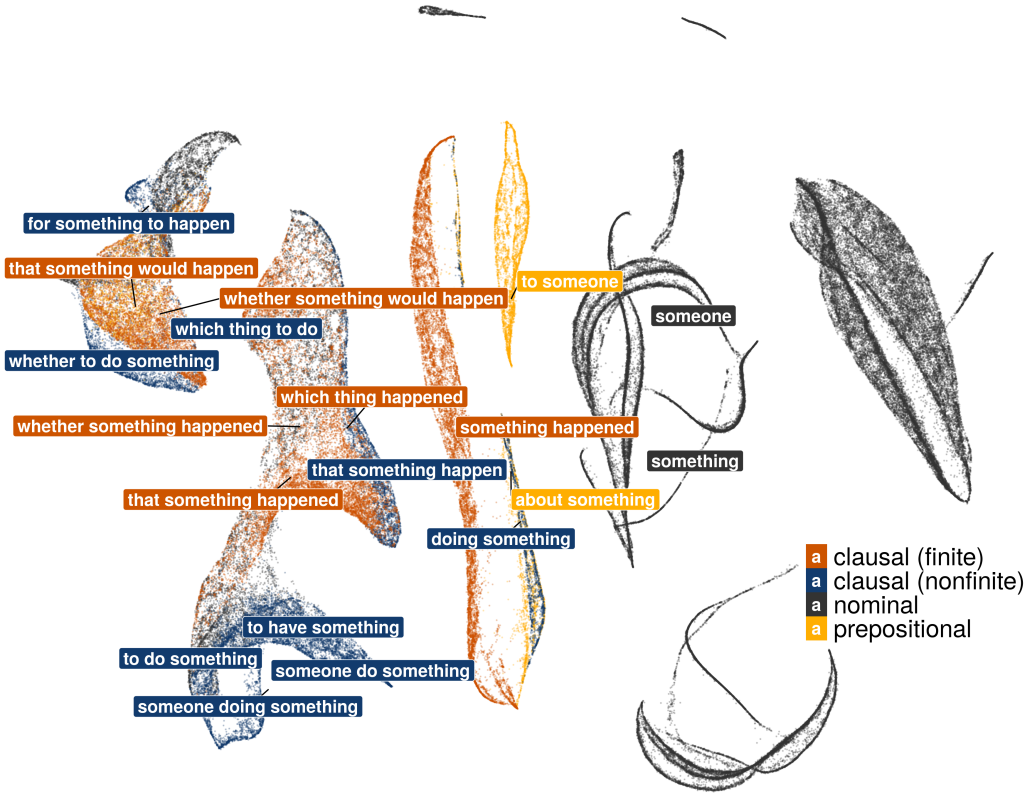

Figure 1 The UMAP visualization for the parser's representation for different expressions colored by coarse-grained expression types. Labels show the mean location of the particular expression.

Tibshirani & Friedman 2009: §7), the data is partitioned into k parts (here, $k = 5$), and for each of the cells, a model is trained on all but that cell and then its performance is evaluated on that cell. Averaging across five validation folds, our model obtains a Pearson correlation of $\rho = 0.71$ (95% CI=[0.69, 0.73]). This correlation is extremely close to the one reported by White & Rawlins (accepted): trained linguists agree on a subset of the MegaAttitude sentences with a Spearman correlation of 0.70 (95% CI=[0.62,0.78]). This result suggests that the models agrees with the normalized scores at about the same level that trained linguists agree with each other.

5.2 Syntactic Representations

Next, we investigate the vector representations for particular constituents. To do this, we run the syntactic parser on all sentences in MegaAcceptability and extract the

vector representations for various nominal, prepositional, and clausal expressions in all contexts they appear. This yields as many vectors for an expression as it has unique tokenings across the dataset—e.g. *whether something happened* appears once in each of 6 frames and so, across 1,007 verbs, we obtain 6,042. It is important to extract a vector for every context in which a particular constituent appears because the outside embedding $\mathbf{h}_{ij}^{\triangledown}$ differs for the same expression $w_{i:j}$ depending on $w_{0:i}, w_{j:|S|}$.

To visualize how these vectors are arranged, we apply the UMAP dimensionality reduction technique (McInnes, Healy & Melville 2018) to map them to two dimensions in a way that preserves the relationships between vectors in the high dimensional space. Figure 1 plots the results. We see a broad split between clausal expressions on the left and nominal and prepositional expressions on the right. The cluster in the upper left contains finite clauses with a future modal as well as infinitival interrogatives and *for-to* clauses. The absence of other infinitivals from this cluster—e.g. subjectless infinitivals and bare infinitivals with and without subjects, which are tightly grouped in a separate cluster—may suggest that the model is formally representing some combination of having a overt subject (or perhaps a complementizer) and containing a modal—in this case, *would* or *to* (Bhatt 1999; Wurmbrand 2014: cf. Stowell 1982; Ogihara 1996; Martin 2001; Katz 2001; Pearson 2016; and see also Grano 2012, 2017; Williamson 2019).

The cluster next to that contains finite clauses—both declarative and interrogative—and tenseless *that* clauses in the upper portion and subjectless and bare infinitivals in the lower portion. The fact that the tenseless *that* clause clusters with the finites may suggest that this grouping is mainly determined by the presence of an overt complementizer. This is bolstered by the fact that the finite clause without a complementizer is contained within its own long thin region just to the right.

5.3 Types

Having established that the syntactic representations our parser learns are reasonable, we now turn to the semantic types our model assigns to different expressions within the high acceptability set used to train the type coherence loss (90th percentile acceptability or above). To obtain these types, we apply the decoding algorithm (see §4) to the vector for an expression in each context in which it is found—i.e. the same vectors analyzed in 5.2—in order to obtain the most likely type(s).[9]

For reasons of space, we focus in particular on the types assigned to expressions along two axes of variation: interrogative v. declarative and finite v. infinitival. Figure 2 shows the proportion of tokenings of a particular expression (y-axis) that

9 An alternative, more computationally intensive method is to decode the type for the expression in the context of the entire sentence. We leave this for future work.

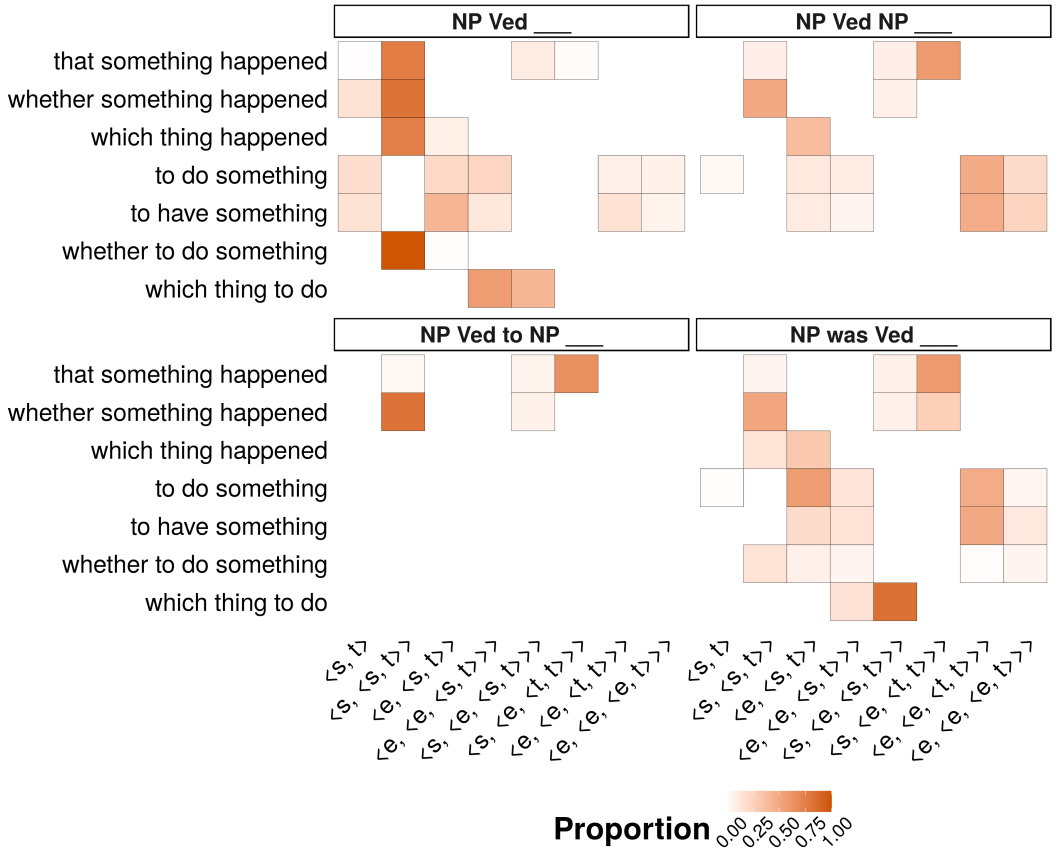

Figure 2 The types most frequently decoded for each expression across verbs.

are assigned particular types (*x*-axis) in a particular context (facets) across verbs.[10]

In intransitive contexts (NP V*ed* ___), declaratives (*that something happened*), finite interrogatives (*whether something happened*, *which thing happened*), and infinitival polar interrogatives (*whether to do something*) all heavily favor the $\langle s, \langle s, t\rangle\rangle$ type, suggesting that our model infers that they denote questions represented as the set of their complete answers (see Hamblin 1958; Groenendijk & Stokhof 1984 among others; cf. Uegaki 2015). Less frequently, declaratives and finite polar interrogatives are assigned the proposition type $\langle s, t\rangle$, and constituent interrogatives are assigned the $\langle e, \langle s, t\rangle\rangle$ type, suggesting a functional question type (Krifka 2011) where the entity argument denotes the "missing piece" corresponding to the WH-phrase (Hintikka 1976; Berman 1991). Interestingly, polar interrogative that are assigned $\langle s, \langle s, t\rangle\rangle$ tend to be complements of responsive predicates—i.e. predicates

10 Indicative of how stringent the 90th percentile threshold is, some rows for some contexts are blank because the relevant expression does not show up in that context within the set of items we investigate.

that take both declaratives and interrogatives, like *know*, *discover*, *find out*, and *realize*—while polar interrogatives that are assigned the proposition type $\langle s,t\rangle$ tend to be rogative predicates (Lahiri 2002)—i.e. predicates that only take interrogatives (or are at least marginal with declaratives), like *wonder* and *ask*. There are potentially interesting exceptions to this generalization, though: *doubt* is responsive, but its polar question complement is assigned a proposition type—possibly related to the intuitive equivalence between *doubt whether* and *doubt that* (Karttunen 1977).

In contrast to finite constituent interrogatives, infinitival constituent interrogatives (*which thing to do*) tend to favor the type corresponding to properties of relations between individuals $\langle e,\langle e,\langle s,t\rangle\rangle\rangle$, with the type corresponding to intensionalized properties of individuals $\langle s,\langle e,\langle s,t\rangle\rangle\rangle$ a close second. At least, one of the entity types in both might be understood as the missing subject, and in the case of $\langle e,\langle e,\langle s,t\rangle\rangle\rangle$, the second might correspond to the WH-phrase, as in a functional question. It may be that this distinction is a real one—between representing constituent interrogatives as functional questions with a missing argument or as partition questions with a missing argument—but in analyzing the distribution over types for particular verbs' complements, we found that there are very few examples where our model assigns $\langle s,\langle e,\langle s,t\rangle\rangle\rangle$ for its highest probability prediction but not $\langle e,\langle e,\langle s,t\rangle\rangle\rangle$ as one of the runner-up predictions and vice-versa. This pattern may indicate uncertainty in the decoder as to which is the correct analysis or it may be that the decoder has trouble distinguishing between the two analyses because they are so close to each other in terms of tree edits—differing by a single primitive type. One potential argument against this possibility is that the decoder appears to be sensitive to the context (and/or the verbs that appear in said context) when deciding between these types: it slightly prefers $\langle e,\langle e,\langle s,t\rangle\rangle\rangle$ in intransitive contexts but strongly prefers $\langle s,\langle e,\langle s,t\rangle\rangle\rangle$ in passivized transitive contexts (NP *was* V*ed* NP ___).

Finally, the distribution over types for the non-interrogative subjectless infinitivals (*to do something*, *to have something*) shows a more diffuse distribution on types, relatively evenly weighting the proposition type $\langle s,t\rangle$, the type corresponding to properties of individuals $\langle e,\langle s,t\rangle\rangle$, and the type corresponding to properties of relations between individuals $\langle e,\langle e,\langle s,t\rangle\rangle\rangle$.[11] The proposition type is expected under theories that posit a covert pronoun in subject position of the infinitival (Rosenbaum 1967 *et seq*) and tends to be associated with verbs of appearance (e.g. raising verbs: *appear* and *seem*), verbs of claiming and pretense (e.g. *claim*, *pretend*), verbs of choice (e.g. *decide*, *choose*), and emotives (e.g. *like*, *love*).[12] The property-of-individuals type is expected under theories that do not assume such a covert pronoun

11 As for infinitival constituent interrogatives, we suspect that the other types $\langle e,\langle e,\langle t,t\rangle\rangle\rangle$ and $\langle e,\langle e,\langle e,t\rangle\rangle\rangle$ may be hard for the decoder to distinguish from $\langle e,\langle e,\langle s,t\rangle\rangle\rangle$.

12 Indeed, many (though not all) of the verbs that are assigned the proposition type are exactly those that would be full tense phrases in Wurmbrand's (2014) theory of infinitival complements.

(Bach 1979; Chierchia 1984; Dowty 1985) and tends to be associated with complements of aspectual verbs (e.g. *start*, *stop*), intention verbs (e.g. *try*, *intend*, *plan*), desire verbs (e.g. *want*, *wish*), and some cognitive verbs (e.g. *remember*, *forget*).

The interpretation of the property-of-relations-between-individuals type is less clear but its assignment appears to correlate with whether the selecting verb can take direct object or PP arguments that can go unexpressed—e.g. *say*, *argue*, *advise*, *look* (for), *long* (for)—and so the model may be learning to "pack" those unexpressed arguments into the clause. This possibility is bolstered by the pattern seen in the transitive (NP V*ed* NP ___, NP *was* V*ed* NP ___) and preposition phrase (NP V*ed* *to* NP ___) contexts, where the possible types for the declarative and finite polar interrogative remain largely the same, though the proportions shift somewhat toward types that have an "extra" e. This additional e might suggest a content individual (Kratzer 2006; Moulton 2009, 2015; Bogal-Allbritten 2016)—i.e. an entity whose content is constrained by the content of the clause—though this would be surprising in light of the correlation with direct object- and preposition-taking behavior. More likely, we suspect, is that the model treats clauses as polysemous between relational and non-relational variants, where the relational variant takes the denotation of a direct object or prepositional phrase as an argument and relates it to the content of the clause via some free relation (Partee 1983/1997) that is further constrained by the verb. This possibility seems perverse until noting that, like the two noun phrase internal arguments in ditransitives, the noun phrase-clause complex can be "non-constituent" coordinated.

(3) a. Jo told Bo that Mo left and Mo that Bo left.
b. Jo told Bo to leave and Mo to stay.

The main difference between standard CCG analyses (see Steedman 2000) and what the model appears to be doing is that, unlike standard analyses, where the coordinated constituent is combined via type-raising followed by composition, the model appears to be imbuing the clause with argument-taking behavior. As such, an alternative take is that the model is attempting to approximate something like a neo-Davidsonian analysis (see Parsons 1990) without primitive event types or a predicate modification rule (see Heim & Kratzer 1998) for ensuring compositional interpretation. Future work might test this possibility within our framework by adding an additional primitive type for events (with concomitant alteration of the type constraints) as well as an additional type decoder for predicate modification.

6 Conclusion

We have presented a computational modeling framework for inducing combinatory categorial grammars from arbitrary behavioral data. This framework provides the

analyst fine-grained control over the assumptions that the induced grammar should conform to: (i) what the primitive types are; (ii) how complex types are constructed; (iii) what set of combinators can be used to combine types; and (iv) whether (and to what) the types of some lexical items should be fixed. As a proof-of-concept experiment, we deployed our framework for use in distributional analysis. We investigated the induced grammar, finding that it assigns a highly interpretable system of types—even for complex clausal expressions of various forms.

This work is only the tip of the iceberg in terms of how our framework might be used in semantic theory-building and evaluation. There are at least three promising areas for future investigation.

Jointly training on multiple distinct datasets. A major benefit of our framework is that it can be used with arbitrary behavioral data. This means not only fitting models to particular datasets, but also using our framework to synthesize multiple datasets into a single induced grammar. In preliminary experiments, we have used our framework to jointly predict acceptability using the syntactic parser representations $\mathbf{h}^{\diamond}_{0|S|}$ (as in this paper) as well as veridicality judgments from the MegaVeridicality data (White & Rawlins 2018; White et al. 2018)—e.g. that *Jo proved that Bo left* entails that *Bo left*, while *Jo suggested that Bo left* does not—using the denotation representations $\boldsymbol{\lambda}_{0|S|}$. We have found that it is possible to jointly predict veridicality and acceptability judgments at native speaker levels, though the types we induce are less interpretable than those presented here.

Beyond behavioral data, it is also possible to use our framework to jointly train on corpus data, building on existing deep learning-based grammar induction methods using only the syntactic parser component of our framework (Drozdov et al. 2019a,b). We are currently experimenting with joint syntactic and semantic CCG induction from corpus data by expanding the set of type constructors and combinators from undirected to directed variants and enforcing additional constraints to capture *combinatory type transparency* (see Steedman 2000).

Alternative grammatical assumptions. Another major benefit of our framework is that primitive types, type constructors, combinators, and type constraints are highly tunable: the analyst need merely specify the set of primitive types, set of constructors, type constraints, and combinator behavior they are interested in. As noted in §5, one potentially interesting direction is to investigate the grammars induced under neo-Davidsonian assumptions. A note of caution is warranted here: we have not presented experiments probing the limits of our frameworks capabilities in terms of the sorts of grammars they can reliably induce. Further work is necessary.

Inducing logical form. Finally, our framework opens up the possibility of not only assigning types to expressions, but also potentially full logical forms. As for types, this induction might be set up as a problem of decoding symbolic expressions conforming to the syntax of some logic from the interpretation vectors $\boldsymbol{\lambda}_{ij}$.

References

An, Hannah & Aaron White. 2020. The lexical and grammatical sources of neg-raising inferences. *Proceedings of the Society for Computation in Linguistics* 3(1). 220–233. doi:10.7275/yts0-q989.

Asher, Nicholas. 2011. *Lexical Meaning in Context: A web of words*. Cambridge: Cambridge University Press.

Asher, Nicholas & James Pustejovsky. 2013. A Type Composition Logic for Generative Lexicon. In James Pustejovsky, Pierrette Bouillon, Hitoshi Isahara, Kyoko Kanzaki & Chungmin Lee (eds.), *Advances in Generative Lexicon Theory* Text, Speech and Language Technology, 39–66. Dordrecht: Springer Netherlands. doi:10.1007/978-94-007-5189-7_3.

Bach, Emmon. 1979. Control in Montague Grammar. *Linguistic Inquiry* 10(4). 515–531. https://www.jstor.org/stable/4178132.

Bahdanau, Dzmitry, Kyunghyun Cho & Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In Yoshua Bengio & Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, Conference Track Proceedings*, San Diego, CA, USA. http://arxiv.org/abs/1409.0473.

Baker, James K. 1979. Trainable grammars for speech recognition. *The Journal of the Acoustical Society of America* 65(S1). S132–S132. doi:10.1121/1.2017061.

Baldridge, Jason. 2002. *Lexically specified derivational control in Combinatory Categorial Grammar*. Edinburgh: University of Edinburgh PhD Thesis.

Baroni, Marco, Raffaella Bernardi & Roberto Zamparelli. 2014. Frege in space: A program for compositional distributional semantics. *Linguistic Issues in Language Technology* 9(6). 5–110.

Berman, Stephen Robert. 1991. *On the semantics and logical form of wh-clauses*. Amherst, MA: University of Massachusetts PhD Thesis.

Bhatt, Rajesh. 1999. *Covert modality in non-finite contexts*. Philadelphia, PA: University of Pennsylvania PhD Thesis.

Bisk, Yonatan & Julia Hockenmaier. 2012a. Induction of linguistic structure with Combinatory Categorial Grammars. In *Proceedings of the NAACL-HLT Workshop on the Induction of Linguistic Structure*, 90–95. Montréal, Canada: Association for Computational Linguistics. https://www.aclweb.org/anthology/W12-1912.

Bisk, Yonatan & Julia Hockenmaier. 2012b. Simple robust grammar induction with Combinatory Categorial Grammars. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, vol. 2 AAAI'12, 1643–1649. Toronto, Ontario, Canada: AAAI Press.

Bisk, Yonatan & Julia Hockenmaier. 2013. An HDP model for inducing Combina-

tory Categorial Grammars. *Transactions of the Association for Computational Linguistics* 1. 75–88. doi:10.1162/tacl_a_00211.

Bisk, Yonatan & Julia Hockenmaier. 2015. Probing the linguistic strengths and limitations of unsupervised grammar induction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 1395–1404. Beijing, China: Association for Computational Linguistics. doi:10.3115/v1/P15-1135.

Bogal-Allbritten, Elizabeth A. 2016. *Building meaning in Navajo*. Amherst, MA: University of Massachusetts PhD Thesis.

Chierchia, Gennaro. 1984. *Topics in the syntax and semantics of infinitives and gerunds*. Amherst, MA: University of Massachusetts PhD Thesis.

Clark, Alexander & Ryo Yoshinaka. 2016. Distributional learning of context-free and multiple context-free grammars. In Jeffrey Heinz & José M. Sempere (eds.), *Topics in grammatical inference*, 143–172. Berlin, Heidelberg: Springer. doi:10.1007/978-3-662-48395-4_6.

Davidson, Donald. 1967. The logical form of action sentences. In Nicholas Rescher (ed.), *The logic of decision and action*, 81–95. Pittsburgh, PA: University of Pittsburgh Press. doi:10.1093/0199246270.003.0006.

Dowty, David R. 1985. On recent analyses of the semantics of control. *Linguistics and Philosophy* 8(3). 291–331. https://www.jstor.org/stable/25001209.

Drozdov, Andrew, Patrick Verga, Yi-Pei Chen, Mohit Iyyer & Andrew McCallum. 2019a. Unsupervised labeled parsing with deep inside-outside recursive autoencoders. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 1507–1512. Hong Kong, China: Association for Computational Linguistics. doi:10.18653/v1/D19-1161.

Drozdov, Andrew, Patrick Verga, Mohit Yadav, Mohit Iyyer & Andrew McCallum. 2019b. Unsupervised latent tree induction with deep inside-outside recursive auto-encoders. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 1129–1141. Minneapolis, Minnesota: Association for Computational Linguistics. doi:10.18653/v1/N19-1116.

Goldberg, Yoav. 2017. *Neural Network Methods for Natural Language Processing*, vol. 10 Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers. doi:10.2200/S00762ED1V01Y201703HLT037.

Grano, Thomas. 2012. *Control and restructuring at the syntax-semantics interface*. Chicago: University of Chicago PhD Thesis.

Grano, Thomas. 2017. Control, temporal orientation, and the cross-linguistic gram-

mar of *trying*. *Glossa* 2(1). 94. doi:10.5334/gjgl.335.

Groenendijk, Jeroen & Martin Stokhof. 1984. *Studies on the semantics of questions and the pragmatics of answers*. Amsterdam: University of Amsterdam PhD Thesis.

Hamblin, Charles Leonard. 1958. Questions. *Australasian Journal of Philosophy* 36(3). 159–168. doi:10.1080/00048405885200211.

Hastie, Trevor, Robert Tibshirani & Jerome Friedman. 2009. *The Elements of Statistical Learning*. New York, NY: Springer-Verlag 2nd edn. doi:10.1007/978-0-387-84858-7.

Heim, Irene & Angelika Kratzer. 1998. *Semantics in Generative Grammar*. Oxford: Blackwell.

Heinz, Jeffrey & José M Sempere (eds.). 2016. *Topics in grammatical inference*. Berlin, Heidelberg: Springer. 10.1007/978-3-662-48395-4.

Hintikka, Jaakko. 1976. The semantics of questions and the questions of semantics: Case studies in the interrelations of logic, semantics and syntax. *Acta Philosophica Fennica* 28(4).

Karttunen, Lauri. 1977. To doubt whether. In *The CLS Book of Squibs*, Chicago Linguistic Society.

Katz, Graham. 2001. (A)temporal complements. In Caroline Fery & Wolfgang Sternefeld (eds.), *Audiator Vox Sapientiae*, 240–258. Berlin: Akademie Verlag.

Kim, Yoon, Alexander Rush, Lei Yu, Adhiguna Kuncoro, Chris Dyer & Gábor Melis. 2019. Unsupervised recurrent neural network grammars. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 1105–1117. Minneapolis, Minnesota: Association for Computational Linguistics. doi:10.18653/v1/N19-1114.

Kratzer, Angelika. 2006. Decomposing attitude verbs. Talk presented at the workshop in honor of Anita Mittwoch. The Hebrew University of Jerusalem.

Krifka, Manfred. 2011. Questions. In Claudia Maienborn, Klaus von Heusinger & Paul Portner (eds.), *Semantics. An international handbook of natural language meaning*, vol. 2, 1742–1758. Berlin: De Gruyter Mouton.

Kwiatkowksi, Tom, Luke Zettlemoyer, Sharon Goldwater & Mark Steedman. 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*, 1223–1233. Cambridge, MA: Association for Computational Linguistics. https://www.aclweb.org/anthology/D10-1119.

Kwiatkowski, Tom, Luke Zettlemoyer, Sharon Goldwater & Mark Steedman. 2011. Lexical generalization in CCG grammar induction for semantic parsing. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, 1512–1523. Edinburgh, Scotland, UK.: Association for Computa-

tional Linguistics. https://www.aclweb.org/anthology/D11-1140.

Lahiri, Utpal. 2002. *Questions and Answers in Embedded Contexts*. Oxford University Press.

Le, Phong & Willem Zuidema. 2014. Inside-outside semantics: A framework for neural models of semantic composition. In *NIPS 2014 Workshop on Deep Learning and Representation Learning*, .

Le, Phong & Willem Zuidema. 2015. Compositional distributional semantics with long short term memory. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics*, 10–19. Denver, Colorado: Association for Computational Linguistics. doi:10.18653/v1/S15-1002.

Lewis, Mike, Kenton Lee & Luke Zettlemoyer. 2016. LSTM CCG parsing. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 221–231. San Diego, California: Association for Computational Linguistics. doi:10.18653/v1/N16-1026.

Lewis, Mike & Mark Steedman. 2014. A* CCG parsing with a supertag-factored model. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 990–1000. Doha, Qatar: Association for Computational Linguistics. doi:10.3115/v1/D14-1107.

Liu, Yinhan, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer & Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv:1907.11692 [cs]* http://arxiv.org/abs/1907.11692. ArXiv: 1907.11692.

Maas, Andrew L, Awni Y Hannun & Andrew Y Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, vol. 30 1, 3.

Manning, Chris & Hinrich Schütze. 1999. *Foundations of statistical natural language processing*. Cambridge, MA: MIT Press.

Martin, Roger. 2001. Null case and the distribution of PRO. *Linguistic Inquiry* 32(1). 141–166.

McInnes, Leland, John Healy & James Melville. 2018. UMAP: Uniform manifold approximation and projection for dimension reduction. https://arxiv.org/abs/1802.03426.

Miwa, Makoto & Mohit Bansal. 2016. End-to-end relation extraction using LSTMs on sequences and tree structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, 1105–1116. Berlin, Germany: Association for Computational Linguistics.

Montague, Richard. 1973. The proper treatment of quantification in ordinary English. In K. J. J. Hintikka, J. M. E. Moravcsik & P. Suppes (eds.), *Approaches to natural language*, 221–242. Springer.

Moon, Ellise & Aaron Steven White. 2020. The source of nonfinite temporal interpretation. In *Proceedings of the 50th Annual Meeting of the North East Linguistic Society*, 11–24. Amherst, MA: GLSA Publications.

Moulton, Keir. 2009. *Natural selection and the syntax of clausal complementation*. Amherst, MA: University of Massachusetts PhD Thesis.

Moulton, Keir. 2015. CPs: Copies and compositionality. *Linguistic Inquiry* 46(2). 305–342.

Ogihara, Toshiyuki. 1996. *Tense, attitudes, and scope* Studies in Linguistics and Philosophy. Springer Netherlands. doi:10.1007/978-94-015-8609-2.

Parsons, Terence. 1990. *Events in the Semantics of English: A study in subatomic semantics*. Cambridge, MA: MIT Press.

Partee, Barbara H. 1983/1997. Uniformity vs. versatility: The genitive, a case study. In Johan van Benthem & Alice ter Meulen (eds.), *The handbook of logic and language*, 464–470. New York: Elsevier.

Pearson, Hazel. 2016. The semantics of partial control. *Natural Language & Linguistic Theory* 34(2). 691–738.

Potts, Christopher. 2019. A case for deep learning in semantics: Response to Pater. *Language* 95(1). e115–e124.

Pustejovsky, James. 2013. Type theory and lexical decomposition. In James Pustejovsky, Pierrette Bouillon, Hitoshi Isahara, Kyoko Kanzaki & Chungmin Lee (eds.), *Advances in Generative Lexicon Theory* Text, Speech and Language Technology, 9–38. Dordrecht: Springer Netherlands. doi:10.1007/978-94-007-5189-7_2.

Rosenbaum, Peter S. 1967. *The Grammar of English Predicate Complement Constructions*. Cambridge, MA: MIT Press.

Shen, Yikang, Zhouhan Lin, Chin-wei Huang & Aaron Courville. 2018. Neural Language Modeling by Jointly Learning Syntax and Lexicon. In *6th International Conference on Learning Representations, (ICLR) Conference Track Proceedings*, Vancouver, BC, Canada. https://openreview.net/forum?id=rkgOLb-0W.

Shieber, Stuart M., Yves Schabes & Fernando C. N. Pereira. 1995. Principles and implementation of deductive parsing. *The Journal of Logic Programming* 24(1). 3–36. doi:10.1016/0743-1066(95)00035-I.

Steedman, Mark. 2000. *The Syntactic Process*, vol. 24. Cambridge, MA: MIT press.

Stowell, Tim. 1982. The tense of infinitives. *Linguistic Inquiry* 13(3). 561–570.

Tai, Kai Sheng, Richard Socher & Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, 1556–1566. Beijing, China: Association for Computational Linguistics.

Uegaki, Wataru. 2015. *Interpreting questions under attitudes*. Cambridge, MA: Massachusetts Institute of Technology PhD Thesis.

White, Aaron Steven. accepted. Believing and hoping whether. *Semantics and Pragmatics* .

White, Aaron Steven & Kyle Rawlins. 2016. A computational model of S-selection. *Semantics and Linguistic Theory* 26(0). 641–663. doi:10.3765/salt.v26i0.3819.

White, Aaron Steven & Kyle Rawlins. 2018. The role of veridicality and factivity in clause selection. In Sherry Hucklebridge & Max Nelson (eds.), *Proceedings of the 48th Annual Meeting of the North East Linguistic Society*, 221–234. Amherst, MA: GLSA Publications.

White, Aaron Steven & Kyle Rawlins. accepted. Frequency, acceptability, and selection: A case study of clause embedding. *Glossa* .

White, Aaron Steven, Rachel Rudinger, Kyle Rawlins & Benjamin Van Durme. 2018. Lexicosyntactic Inference in Neural Models. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 4717–4724. Brussels, Belgium: Association for Computational Linguistics. doi:10.18653/v1/D18-1501.

Williams, Adina, Andrew Drozdov & Samuel R. Bowman. 2018. Do latent tree learning models identify meaningful structure in sentences? *Transactions of the Association for Computational Linguistics* 6. 253–267. doi:10.1162/tacl_a_00019.

Williamson, Gregor. 2019. The temporal orientation of infinitives. In *Proceedings of Sinn und Bedeutung*, vol. 23, 461–478. Issue: 2.

Wurmbrand, Susi. 2014. Tense and aspect in English infinitives. *Linguistic Inquiry* 45(3). 403–447.

Younger, Daniel H. 1967. Recognition and parsing of context-free languages in time n^3. *Information and Control* 10(2). 189–208.

Zettlemoyer, Luke & Michael Collins. 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)*, 678–687. Prague, Czech Republic: Association for Computational Linguistics. https://www.aclweb.org/anthology/D07-1071.

Zettlemoyer, Luke & Michael Collins. 2009. Learning context-dependent mappings from sentences to logical form. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, 976–984. Suntec, Singapore: Association for Computational Linguistics. https://www.aclweb.org/anthology/P09-1110.

Zettlemoyer, Luke S. & Michael Collins. 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 UAI'05, 658–666. Arlington, Virginia, USA: AUAI Press.

Gene Louis Kim
Department of Computer Science
University of Rochester
500 Joseph C. Wilson Blvd.
Rochester, NY, USA 14627
gkim21@cs.rochester.edu

Aaron Steven White
Department of Linguistics
University of Rochester
500 Joseph C. Wilson Blvd.
Rochester, NY, USA 14627
aaron.white@rochester.edu