Arguments and Modifiers from the Learner's Perspective

Leon Bergen
MIT
Brain and Cognitive Science
bergen@mit.edu

Edward Gibson
MIT
Brain and Cognitive Science
egibson@mit.edu

Timothy J. O'Donnell
MIT
Brain and Cognitive Science
timod@mit.edu

Abstract

We present a model for inducing sentential argument structure, which distinguishes arguments from optional modifiers. We use this model to study whether representing an argument/modifier distinction helps in learning argument structure, and whether a linguistically-natural argument/modifier distinction can be induced from distributional data alone. Our results provide evidence for both hypotheses.

1 Introduction

A fundamental challenge facing the language learner is to determine the content and structure of the stored units in the lexicon. This problem is made more difficult by the fact that many lexical units have *argument structure*. Consider the verb *put*. The sentence, *John put the socks* is incomplete; when hearing such an utterance, a speaker of English will expect a location to also be specified: *John put the socks in the drawer*. Facts such as these can be captured if the lexical entry for *put* also specifies that the verb has three required arguments: (i) who is doing the putting (ii) what is being put (iii) and the destination of the putting.

The problem of acquiring argument structure is further complicated by the fact that not all phrases in a sentence fill an argument role. Instead, many are *modifiers*. Consider the sentence *John put the socks in the drawer at 5 o'clock*. The phrase *at 5 o'clock* occurs here with the verb *put*, but it is not an argument. Removing this phrase does not change the core structure of the PUTTING event, nor is the sentence incomplete without this phrase.

The distinction between arguments and modifiers has a long history in traditional grammar and is leveraged in many modern theories of syntax (Haegeman, 1994; Steedman, 2001; Sag et al., 2003). Despite the ubiquity of the distinction in syntax, however, there is a lack of consensus on the necessary and sufficient conditions for argumenthood (Schütze, 1995; Schütze and Gibson, 1999). It remains unclear whether the argument/modifier distinction is purely semantic or is also represented in syntax, whether it is binary or graded, and what effects argument/modifierhood have on the distribution of linguistic forms.

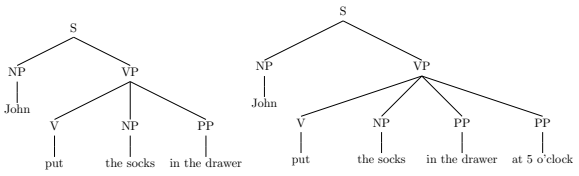

Figure 1: The VP's in these sentences only share structure if we separate arguments from modifiers.

In this work, we take a new approach to these problems. We propose that the argument/modifier distinction is inferred on a phrase–by–phrase basis using probabilistic inference. Crucially, allowing the learner to separate the core argument structure of phrases from peripheral modifier content increases the generalizability of argument constructions. For example, the two sentences in Figure 1 intuitively share the same argument structures, but this overlap can only be identified if the prepositional phrase, "at 5 o'clock," is treated as a modifier. Thus representing the argument/modifier distinction can help the learner find useful argument structures which generalize robustly.

Although, like the majority of theorists, we agree that the argument/adjunct distinction is fundamentally semantic, in this work we focus on its distributional correlates. Does the optionality of modifier phrases help the learner acquire lexical items with the right argument structure?

2 Approach

We adopt an approach where the lexicon consists of an inventory of stored tree fragments. These

Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, pages 115–119,
Sofia, Bulgaria, August 4-9 2013. ©2013 Association for Computational Linguistics

tree fragments encode the necessary phrase types (i.e., arguments) that must be present in a structure before it is complete. In this system, sentences are generated by recursive *substitution* of tree fragments at the frontier argument nodes of other tree fragments. This approach extends work on learning probabilistic Tree–Substitution Grammars (TSGs) (Post and Gildea, 2009; Cohn et al., 2010; O'Donnell, 2011; O'Donnell et al., 2011).[1]

To model modification, we introduce a second structure–building operation, *adjunction*. While substitution must be licensed by the existence of an argument node, adjunction can insert constituents into well–formed trees. Many syntactic theories have made use of an adjunction operation to model modification. Here, we adopt the variant known as *sister–adjunction* (Rambow et al., 1995; Chiang and Bikel, 2002) which can insert a constituent as the sister to any node in an existing tree.

In order to derive the complete tree for a sentence, starting from an `S` root node, we recursively sample arguments and modifiers as follows.[2] For every nonterminal node on the frontier of our derivation, we sample an elementary tree from our lexicon to substitute into this node. As already noted, these elementary trees represent the argument structure of our tree. Then, for each argument nonterminal on the tree's interior, we sister–adjoin one or more modifier nodes, which themselves are built by the same recursive process.

Figure 2 illustrates two derivations of the same tree, one in standard TSG without sister–adjunction, and one in our model. In the TSG derivation, at top, an elementary tree with four arguments – including the intuitively optional temporal PP – is used as the backbone for the derivation. The four phrases filling these arguments are then substituted into the elementary tree, as indicated by arrows. In the bottom derivation, which uses sister–adjunction, an elementary tree with only three arguments is used as the backbone. While the right-most temporal PP needed to be an argument of the elementary tree in the TSG derivation, the bottom derivation uses sister–adjunction to insert this PP as a child of the VP. Sister–adjunction therefore allows us to use an argument structure that matches the true argument structure of the verb "put."

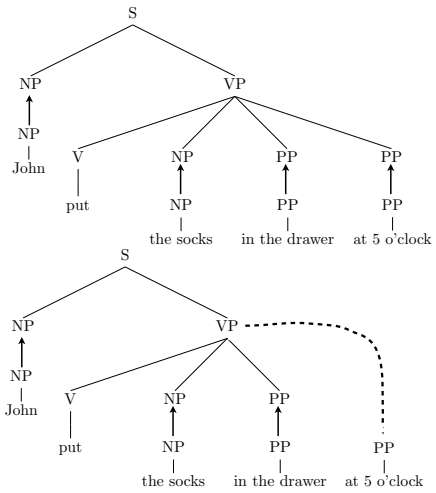

Figure 2: The first part of the figure shows how to derive the tree in TSG, while the second part shows how to use sister-adjunction to derive the same tree in our model.

This figure illustrates how derivations in our model can have a greater degree of generalizability than those in a standard TSG. Sister–adjunction will be used to derive children which are not part of the core argument structure, meaning that a greater variety of structures can be derived by a combination of common argument structures and sister-adjoined modifiers. Importantly, this makes the learning problem for our model less sparse than for TSGs; our model can derive the trees in a corpus using fewer types of elementary trees than a TSG. As a result, the distribution over these elementary trees is easier to estimate.

To understand what role modifiers play during learning, we will develop a learning model that can induce the lexicon and modifier contexts used by our generative model.

3 Model

Our model extends earlier work on induction of Bayesian TSGs (Post and Gildea, 2009; O'Donnell, 2011; Cohn et al., 2010). The model uses a Bayesian non–parametric distribution—the Pitman-Yor Process, to place a prior over the lexicon of elementary trees. This distribution allows the complexity of the lexicon to grow to arbitrary size with the input, while still enforcing a bias for more compact lexicons.

[1] Note that we depart from many discussions of argument structure in that we do not require that every stored fragment has a head word. In effect, we allow completely abstract phrasal constructions to also have argument structures.

[2] Our generative model is related to the generative model for Tree–Adjoining Grammars proposed in (Chiang, 2000)

For each nonterminal c, we define:

$$G_c|a_c, b_c, P_E \sim \text{PYP}(a_c, b_c, P_E(\cdot|c)) \quad (1)$$

$$e|c, G_c \sim G_c, \quad (2)$$

where $P_E(\cdot|c)$ is a context free distribution over elementary trees rooted at c, and e is an elementary tree.

The context-free distribution over elementary trees $P_E(e|c)$ is defined by:

$$P_E(e|c) = \prod_{i \in I(e)} (1 - s_{c_i}) \prod_{f \in F(e)} s_{c_f} \prod_{c' \to \alpha \in e} P_{c'}(\alpha|c'), \quad (3)$$

where $I(e)$ is the set of internal nodes in e, $F(e)$ is the set of frontier nodes, c_i is the nonterminal category associated with node i, and s_c is the probability that we stop expanding at a node c. For this paper, the parameters s_c are set to 0.5.

In addition to defining a distribution over elementary trees, we also define a distribution which governs modification via sister–adjunction. To sample a modifier, we first decide whether or not to sister–adjoin into location l in a tree. Following this step, we sample a modifier category (e.g., a PP) conditioned on the location l's *context*: its parent and left siblings. Because contexts are sparse, we use a backoff scheme based on hierarchical Dirichlet processes similar to the ngram backoff schemes defined in (Teh, 2006; Goldwater et al., 2006). Let c be a nonterminal node in a tree derived by substitution into argument positions. The node c will have $n \geq 1$ children derived by argument substitution: $d_0, ..., d_n$. In order to sister–adjoin between two of these children d_i, d_{i+1}, we recursively sample nonterminals $s_{i,1}, ..., s_{i,k}$ until we hit a STOP symbol:

$$\begin{aligned} & P_a(s_{i,1}, ..., s_{i,k}, STOP|C_0) \quad (4) \\ = & \prod_{j=1}^{k} P_a(s_{i,j}|C_j) \cdot (1 - P_{C_j}(STOP)) \\ & \cdot P_{C_{k+1}}(STOP) \end{aligned}$$

where $C_j = d_1, s_{1,1}, ..., d_i, s_{i,1}, ..., s_{i,j-1}, c$ is the context for the j'th modifier between these children. The distribution over sister–adjoined nonterminals is defined using a hierarchical Dirichlet process to implement backoff in a prefix tree over contexts. We define the distribution $G(q_l, ..., q_1)$ over sister–adjoined nonterminals $s_{i,j}$ given the context $q_l, ..., q_1$ by:

$$G(q_l, ..., q_1) \sim \text{DP}(\alpha, G(q_{l-1}, ..., q_1)). \quad (5)$$

The distribution G at the root of the hierarchy is not conditioned on any prior context. We define G by:

$$G \sim \text{DP}(\alpha, \text{Multinomial}(\mathbf{m})) \quad (6)$$

where $\mathbf{m}$ is a vector with entries for each nonterminal, and where we sample $\mathbf{m} \sim$ Dir(1,...,1).

To perform inference, we developed a local Gibbs sampler which generalizes the one proposed by (Cohn et al., 2010).

4 Results

We evaluate our model in two ways. First, we examine whether representing the argument/modifier distinction increases the ability of the model to learn highly generalizable elementary trees that can be used as argument structures across a variety of sentences. Second, we ask whether our model is able to induce the correct argument/modifier distinction according to a linguistic gold–standard. We trained our model on sections 2–21 of the WSJ part of the Penn Treebank (Marcus et al., 1999). The model was trained on the trees in this corpus, without any further annotations for substitution or modification.

To address the first question, we compared the structure of the grammar learned by our model to a grammar learned by a version of our model without sister–adjunction (i.e., a TSG similar to the one used in Cohn et al.). Our model should find more common structure among the trees in the input corpus, and therefore it should learn a set of elementary trees which are more complex and more widely shared across sentences. We evaluated this hypothesis by analyzing the average complexity of the most probable elementary trees learned by these models. As Table 1 shows, our model discovers elementary trees that have greater depth and more nodes than those found by the TSG. In addition, our model accounts for a larger portion of the corpus with fewer rules: the top 50, 100, and 200 most common elementary trees in our model's lexicon account for a greater portion of the corpus than the corresponding sets in the TSG.

Figure 3 illustrates a representative example from the corpus. By using sister-adjuntion to separate the ADVP node from the rest of the sentence's derivation, our model was able to use a common depth-3 elementary tree to derive the backbone of the sentence. In contrast, the TSG cannot give the same derivation, as it needs to include the ADVP

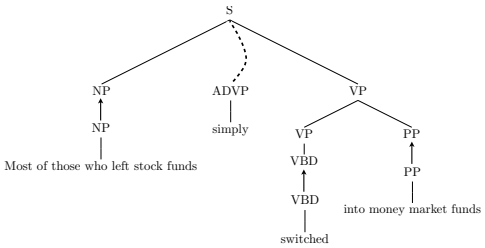

Figure 3: Part of a derivation found by our model.

Model	Rank	Avg tree depth	Avg tree size	#Tokens
Modifier	50	1.59	3.42	97282
TSG	50	1.38	2.98	88023
Modifier	100	1.84	3.98	134205
TSG	100	1.58	3.38	116404
Modifier	200	1.97	4.27	170524
TSG	200	1.77	3.84	146040

Table 1: This table shows the average depth and node count for elementary trees in our model and the TSG. The results are shown for the 50, 100, and 200 most frequent types of elementary trees.

node in the elementary tree; this wider elementary tree is much less common in the corpus.

We next examined whether our model learned to correctly identify modifiers in the corpus. Unfortunately, marking for argument/modifiers in the Penn Treebank is incomplete, and is limited to certain adverbials, e.g. locative and temporal PP's. To supplement this markup, we made use of the corpus of (Kaeshammer and Demberg, 2012). This corpus adds annotations indicating, for each node in the Penn Treebank, whether that node is a modifier. This corpus was compiled by combining information from Propbank (Palmer et al., 2005) with a set of heuristics, as well as the NP-branching structures proposed in (Vadas and Curran, 2007). It is important to note that this corpus can only serve as a rough benchmark for evaluation of our model, as the heuristics used in its development did not always follow the correct linguistic analysis; the corpus was originally constructed for an alternative application in computational linguistics, for which non–linguistically–natural analyses were sometimes convenient. Our model was trained on this corpus, after it had been stripped of argument/modifier annotations.

We compare our model's performance to a random baseline. Our model constrains every nonterminal to have at least one argument child, and our Gibbs sampler initializes argument/modifier choices randomly subject to this constraint. We

Model	Precision	Recall	#Guessed	#Correct
Random	0.27	0.19	298394	82702
Modifier	0.62	0.15	108382	67516

Table 2: This table shows precision and recall in identifying modifier nodes in the corpus.

therefore calculated the probability that a node that was randomly initialized as a modifier was in fact a modifier, i.e. the precision of random initialization. Next, we looked at the precision of our model following training. Table 2 shows that among nodes that were labeled as modifiers, 0.27 were labeled correctly before training and 0.62 were labeled correctly after. This table also shows the recall performance for our model decreased by 0.04. Some of this decrease is due to limitations of the gold standard; for example, our model learns to classify infinitives and auxiliary verbs as arguments — consistent with standard linguistic analyses — whereas the gold standard classifies these as modifiers. Future work will investigate how the metric used for evaluation can be improved.

5 Summary

We have investigated the role of the argument/modifier distinction in learning. We first looked at whether introducing this distinction helps in generalizing from an input corpus. Our model, which represents modification using sister–adjunction, learns a richer lexicon than a model without modification, and its lexicon provides a more compact representation of the input corpus. We next looked at whether the traditional linguistic classification of arguments and modifiers can be induced from distributional information. Without supervision from the correct labelings of modifiers, our model learned to identify modifiers more accurately than chance. This suggests that although the argument/modifier distinction is traditionally drawn without reference to distributional properties, the distributional correlates of this distinction are sufficient to partially reconstruct it from a corpus. Taken together, these results suggest that representing the difference between arguments and modifiers may make it easier to acquire a language's argument structure.

Acknowledgments

We thank Vera Demberg for providing the gold standard, and Tom Wasow for helpful comments.

References

David Chiang and Daniel Bikel. 2002. Recovering latent information in treebanks. In *Proceedings of COLING 2002*.

David Chiang. 2000. Staistical parsing with an automatically–extracted tree adjoining grammar. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Trevor Cohn, Phil Blunsom, and Sharon Goldwater. 2010. Inducing tree–substitution grammars. *Journal of Machine Learning Research*, 11:3053–3096.

Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. 2006. Interpolating between types and tokens by estimating power–law generators. In *Advances in Neural Information Processing Systems 18*, Cambridge, Ma. MIT Press.

Liliane Haegeman. 1994. *Government & Binding Theory*. Blackwell.

Mirian Kaeshammer and Vera Demberg. 2012. German and English treebanks and lexica for tree–adjoining grammars. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2012)*.

Mitchell P. Marcus, Beatrice Santorini, Mary Ann Marcinkiewicz, and Ann Taylor. 1999. Treebank–3. Technical report, Linguistic Data Consortium, Philadelphia.

Timothy J. O'Donnell, Jesse Snedeker, Joshua B. Tenenbaum, and Noah D. Goodman. 2011. Productivity and reuse in language. In *Proceedings of the 33rd Annual Conference of the Cognitive Science Society*.

Timothy J. O'Donnell. 2011. *Productivity and Reuse in Language*. Ph.D. thesis, Harvard University.

Martha Palmer, P. Kingsbury, and Daniel Gildea. 2005. The proposition bank. *Computational Linguistics*, 31(1):71–106.

Matt Post and Daniel Gildea. 2009. Bayesian learning of a tree substitution grammar. 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*.

Owen Rambow, K. Vijay-Shanker, and David Weir. 1995. D–tree grammars. In *Proceedings of the 33rd annual meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Ivan A. Sag, Thomas Wasow, and Emily M. Bender. 2003. *Syntactic Theory: A Formal Introduction*. CSLI, Stanford, CA, 2 edition.

Carson T Schütze and Edward Gibson. 1999. Argumenthood and english prepositional phrase attachment. *Journal of Memory and Language*, 40(3):409–431.

Carson T. Schütze. 1995. PP attachment and argumenthood. Technical report, Papers on language processing and acquisition, MIT working papers in linguistics, Cambridge, Ma.

Mark Steedman. 2001. *The syntactic process*. The MIT press.

Yee Whye Teh. 2006. A Bayesian interpretation of interpolated Kneser-Ney. Technical Report TRA2/06, National University of Singapore, School of Computing.

David Vadas and James Curran. 2007. Adding noun phrase structure to the penn treebank. In *Proceedings of the 45th annual meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.