Representing graphs with sparse vectors.

Fred Callaway

April 17, 2016

Introduction

The representation of structure is a fundamental prerequisite for sophisticated cognition. Whether an agent wants to navigate a physical environment, select a socially successful mate, or read an undergraduate's half-baked honors thesis, she will need an internal model of the relevant system. While simple reinforcement learning may be able to explain some basic behaviors, there is a strong consensus that even basic organisms rely on internal models to learn about and interact with their environment [citation]. Thus, studying these internal representations makes up a major part of current work cognitive science.

A significant challenge for this line of work is that representation can—indeed must—be described at multiple levels of abstraction (Marr 1982). Although the representations of terrestrial animals must ultimately be in terms of synaptic weights and neural activations, a theory at this level is only partially explanatory. To fully understand a representational system, one must identify larger functional units that emerge from the representational substrate. It is possible that a combination of neural and behavior evidence cannot uniquely identify a single explanation [cite anderson], however this does not invalidate the explanatory power of any one functional explanation. TODO

- edelman 2008
- difficulty connecting models and levels

One modeling tool that may be especially useful for modeling the representation of structure is a graph. A graph represents elements and the connections between them. In the simplest case, elements contain no information besides a unique identifier; their function or meaning is defined entirely by their connection to other elements. Brains themselves closely resemble a graph of this type: a weighted directed graph with neurons as vertices and synapses as edges. However, abstracting away from neurons one can embed information and perhaps function into the nodes and edges themselves. As this additional information becomes more complex, the graph becomes capable of representing highly abstract theories, such as those put forward in psychology and linguistics.

Thus, graphs are a flexible and powerful tool capable of representing theories in many domains posed at many levels of analysis. By representing different models in a common format, it becomes far easier to connect these models. For example, a linguist may incorporate many different edge labels to represent different dependency relationships between words. Meanwhile, a neuroscientist may discover a representational technique brains use to represent different connection types using only unlabeled edges (i.e. synapses). These models can then be combined, allowing the abstract linguistic model to make specific neural predictions. Furthermore, a social psychologist studying the representation of social structure in baboons could use the same neural model to augment her own work.

- graph as tool to represent structure
- language as a graph
- graph can represent both simple and complex models
- graph as a bridge from simple statistical models to complex linguistic models

Graphical models of language

The earliest statistical language models, and many of the most commonly used models today [??], are based on transitional probabilities. These models attempt to capture regularities in language by learning the statistical dependencies between adjacent words. The simplest such model is the bigram model, which treats language as a first order Markov process: each word is assumed to depend stochastically on only the previous word. A bigram model is generally represented as a transitional probability matrix, or equivalently a graph with words as nodes and transition probabilities as edges. In this model, an utterance can be produced by starting at a node n_0 (often a special START node), and then choosing the next a node n_1 with probability equal to the edge weight from n_0 to n_1. This process can be iterated until a termination criteria is reached (often the arrival at a special STOP node).

Even under the false assumption that people speak purely based on statistical dependencies between words, the bigram model is fundamentally lacking. Language is rife with long distance dependencies such as "either-or" that a bigram model cannot possibly capture. One strategy to capture long distance dependencies is to increase the order of the Markov process. For example, a second order Markov process, or trigram model, assumes that a word depends on both the previous word and the word before that one. With some squinting, a trigram model can be represented as a standard directed graph with two words in each node. For example, the transitional probability $p(w_i = z | w_{i-1} = y, w_{i-2} = x)$ would be represented as the edge between the node n_{xy} and n_{yz}. [keep/expand?]

However, increasing the Markov order has the undesirable side effect of exponentially increasing the dimensionality of the space. There are n^N possible N-grams, where N is the Markov order and n is the vocabulary size. Thus, as N increases, the percentage of grammatically valid N-grams that the learner will actually be exposed to will decrease exponentially. Many techniques in Natural Language Processing are designed to get around this problem of data sparsity, such as smoothing or variable order N-grams. For example, the back-off algorithm measures all N-gram probabilities of $N < N_{max}$, and dynamically decides which order, N to use in a probability estimation based on the number of relevant N-grams it has stored for each N (Katz 1987). This idea of tracking variable length sequences is a fundamental basis of the present model, and the two models upon which it is based.

The ADIOS model (Solan et al. 2005) explores one such technique that aims to respect the hierarchical nature of language. Unlike N-gram models which always predict the single next word based on some number of previous words, ADIOS directly models the statistical dependencies between multi-word units, e.g. between "the dog" and "ate the steak". These multi-word units or "patterns" are constructed recursively through an iterative batch-learning algorithm. When two nodes (each of which may represent any number of words) are found to frequently occur adjacently in the corpus, they are combined into a new node. Later iterations may discover that this node occurs frequently with another node, allowing the creation of deep hierarchical patterns. TODO: More?

Although ADIOS demonstrated the utility of graphical representations in language modeling, the batch learning algorithm it employed casts some doubt on its relevance as a psychological model. To address this concern Kolodny, Lotem, and Edelman (2015) created U-MILA, an incremental model based on ADIOS but intended to more closely reflect human learning abilities. The model is incremental, passing through the corpus a single time, building up the graph from an initially clean slate. TODO: More.

Graphical models and traditional syntactic theories

Network theories of language often suggest that language is most fundamentally described by relationships between individual words (Hudson 2003), while traditional linguistic theory emphasizes composition and relationships between constituents (Chomsky 1965; Chomsky 1995; Stabler 1996). Although these views are frequently seen as opposing, they may serve complimentary roles. Perhaps direct word-word relationships characterize early linguistic knowledge that is only later developed into more complex syntactic structures (Bannard, Lieven, and Tomasello 2009). It is also possible that the representations of formal linguistics should be viewed as theoretical abstractions that roughly characterizes a fuzzier, network-based language system (Lamb 1999). Marr (1982) calls Chomsky's (1965) theory of transformational grammar a "true computational theory", suggesting that Chomsky's performance-competence distinction reflects Marr's algorithmic-computational distinction. Under this interpretation, network and phrase structure theories may be posed at different levels of analysis; if the theories make roughly similar predictions, they may not conflicting.

BindGraph (sorry)

We describe an augmented graphical data structure, intended to represent domains that are defined by complex inter-relationships and compositional structure. Language involves more than the relationships between words and constituents [citation]; however, the structural aspects of language may be modeled well with such a data structure. The fundamental principle of a BindGraph are (1) elements are defined by their binary relationships with other elements, and (2) elements can be meaningfully combined to form new elements. Both of these principles have a long history in linguistics. Firth (1957) observed that "you shall know a word by the company it keeps" (p. 11), and perhaps additionally by "how it keeps it" (Jones and Mewhort 2007, 2). In theoretical syntax, dependency grammars [citation] and combinatory categorial grammar (Steedman 2000) are both based on the relationships between individual elements (although moderated by categories).

The notion of composition has an even richer history in linguistics. Nearly all modern syntactic theories take some sort of binding operation as the fundamental operation. The operation may be called "Merge" (Chomsky 1995), "(function) application" (Steedman 2000), or "Unification" (Hagoort 2004); however, in each of these frameworks (with widely varying theoretical motivations), the principle of combining two elements to form one element remains. Despite this similarity, these three frameworks use widely differing representational machinery, and as a result, it is sometimes difficult to specify the differences and similarities between theories. Graphs may provide a general representational framework with which different linguist theories could be compared.

A BindGraph is based on a weighted, labeled, directed, multigraph. That is, a node may have multiple edges pointing to another node, representing different types of connections. Additionally, nodes in a BindGraph may be composed of other nodes, inspired by Higraphs (Harel 1988). For example, the constituent `[the dog]` could be a single node, composed of the nodes `the` and `dog`. This node could have two edges to the node `scared` labeled *argument* and *subject.*

A BindGraph is a tuple $G = TODO$ where

- V is a finite set of vertices.
- Σ_E is a finite set of edge labels.
- E is finite set of edges, each of which is a tuple (x, y, ℓ) where x and y are vertices, and ℓ is label $\in \Sigma_E$
- $fweight$ is a function where $fweight(x, y, \epsilon)$ is the edge weight of type ϵ from x to y
- $fbump$ is a function where $fbump(X, Y, \epsilon)$ increases the edge weight of type ϵ from X to Y
- $fbind$ is a function where, for a list of nodes $\mathbf{x}$, $fbind(\mathbf{x})$ is a node y where y represents the ordered composition of the nodes in $\mathbf{x}$.

[TODO Not sure how to formally describe a mutable data structure. Note that the node actually contains all its edges, which is what makes bind work. A node can have outgoing edges without being "in" the graph.]

Like a standard graph, BindGraph is an abstract data type, meaning it can be implemented in a number of ways. In the case of a BindGraph, however, the implementation can have a profound impact on the behavior. In particular, the bind function plays a major role in the intelligence of the graph. This function can be specified by the modeler or it can be learned.

Graphs in space

We describe an implementation of a graph using vectors in a large, fixed-dimension space. The motivation for such an implementation is two-fold. First, recent years have seen promising work in composition operations for vectors, much of it with a focus on semantic composition in language (see Mitchell and Lapata 2010 for a review). More generally, fixed-dimension vectors (e.g. feature vectors) are the basis of many machine learning algorithms; representing a graph with such vectors allows a modeler to draw on this work, for example when constructing an algorithm to learn a binding function. Second, any model that employs distributed representations bears a closer resemblance to brains, which are fundamentally distributed computers. Implementing a symbolic graph with distributed vectors may provide a bridge between the symbolic representations preferred by linguists and the distributed representations proposed by the PDP and Connectionist frameworks (see Smolensky and Legendre 2006 for an alternative approach).

TODO expand on each of the above points?

To construct a spatial representation of a graph, we begin with the traditional adjacency matrix. Noting similarities between this matrix and the co-occurrence matrices employed by distributional semantic models. Inspired by recent work in this field, we construct an approximate representation of an adjacency matrix using *random indexing*, an incremental and efficient method for approximating a large, sparse matrix (see Sahlgren 2005 for an accessible review). The resulting data structure closely mimics the behavior of an adjacency matrix representation, but additionally affords transformation, similarity, and composition operations defined in fixed-dimension spaces.

Distributional semantic models

The core idea underlying distributional semantic models such as HAL (Lund and Burgess 1996), LSA (Landauer and Dumais 1997), and Topic Models (Griffiths, Steyvers, and Tenenbaum 2007), is that a words meaning can be approximated by the contexts in which it is used. The data come in the form of a very large and sparse matrix, with one row for each word and one column for each document (or word, depending on how context is defined). The row can be interpreted as a feature vector specifying the word's meaning, and this vector resides in a very high dimensional word/document space. This space is, in fact, so large that the raw vectors are too large to effectively use. To address this, distributional models employ some form of dimensionality reduction such as singular value decomposition. However, this operation is very costly and it must be rerun from scratch if one wishes to add more data. This is not so problematic in an engineering context, when a model can be trained once and used many times. However, as a cognitive model, batch processes such as these leave much to be desired.

Kanerva, Kristofersson, and Holst (2000) demonstrate that random indexing can be used as an efficient, online dimensionality reduction tool for distributional models. Rather than constructing the full word by document matrix and then applying dimensionality reduction, this technique avoids building this matrix to begin with. The number of columns is fixed ahead of time to be some constant $D << N$ (where N is the number of documents). Each document is then assigned an *index vector* which is a sparse ternary vectors (i.e. containing many 0's and a few 1s and −1s). The rows of the matrix, called *context vectors* are produced by adding a document's index vector to the context vector of every word occurring in the document. That is, a word's context vector is the sum of the id vectors for every document it occurs in. This technique has been found to produce similar results to LSA at a fraction of the computational cost (Karlgren and Sahlgren 2001).

A similar approach is taken by Jones and Mewhort (2007), who describe a method for incorporating order information into distributional semantic models using holographic reduced representations (Plate and others 1995). This work makes the important contribution of including multiple types of information in one vector. The BEAGLE model encodes N-grams of various sizes surrounding a word using circular convolution with permutation (to preserve order, as suggested by Plate and others 1995).

Random indexing for graphs

A standard representation of a graph is an adjacency matrix. This matrix is $M_{N \times N}$, where each row represents the outgoing edges of one node. Applying this interpretation to the co-occurrence matrix used in a word-word distributional semantic models, we have a graph with words as nodes and co-occurrence counts as edge weights. If a co-occurrence matrix can be interpreted as a graph, and a co-occurrence matrix can be approximated with random indexing, perhaps a graph can also be approximated with random indexing. Indeed, viewing a co-occurrence matrix as a special case of a graph, this is simply a generalization of the random indexing technique for co-occurrence graphs to any other kind of graph.

We can construct such a generalization by directly mapping elements of Kanerva's algorithm to elements of a graph. Context vectors, which represent the meaning of a word, become *row vectors*, which represent all outgoing edges of a node. Just as each word/document receives an index vector, each node receives an index vector. To increase the weight of the link from x to y, we add id_y to row_x. Similarity between nodes is defined as cosine similarity. Nodes that have similar outgoing edges, or *edge profiles*, will be similar because their row vectors will contain many of the same id vectors. Importantly, because random vectors in a high dimensional space tend to be nearly orthogonal, row vectors for nodes that share no outgoing edges will have similarity close to 0. Additionally, we can use the cosine operation between a row_x and id_y to recover the edge weight of x to y.

An interesting attribute of this representation is that edge weights behave somewhat like probabilities. That is, increasing the weight from x to y will slightly decrease the weight from x to z. Visually, y is pulling x towards it, and thus away from z. However, unlike probabilities, there is no hard bound on the sum of all edge weights for a node. The total outgoing edge weight for a single node increases as the number of outgoing edges increase, but at a decelerating rate, as shown in figure #.

Edge labels

A key design choice is in how edge labels are implemented. One option is to encode all edges in one vector, using permutations to differentiate edges with different labels. Basile, Caputo, and Semeraro (2011) use this technique to encode syntactic dependency information in a spatial semantic model. Specifically, each dependency relation (e.g. *subject*) receives a unique permutation vector, which is used to permute a word's index vector before adding it to the context vector. This method has the advantage of keeping the dimensionality constant, regardless of the number of unique edge labels.

However, the additive combination of edges of different type has some side effects. First, they will compete with each other—if one edge type gets used more frequently, it will overpower the other. Secondly, any similarity computation will use additive feature combination. This is problematic in settings where the *intersection* of features is critical. For example, while measuring the similarity of `the` and `me`, an edge type that points to commonly preceding elements will have similar profiles for both words. This could result in `the` and `me` being classified as the same part of speech. If edges with different labels were stored separately, on the other hand, we could measure the similarity for each separately and take the product of similarities, which would be low if an edge point to commonly following elements was included. [TODO this is messy. There's a great Levy citation here I can't find.]

Experiment 1: Effect of dimensionality on storage capacity

To confirm that the sparse vector implementation of a graph reasonably matches a traditional graph representation, we compare the HoloGraph to a graph with directly calculated transitional probabilities as edges: ProbGraph (recall that HoloGraph edges roughly mirror probabilities). We expect that, as more edges are stored in a single vector, non-orthogonal index vectors will interfere with each other, resulting in noisy recovered edge weights. However, as the dimensionality of the

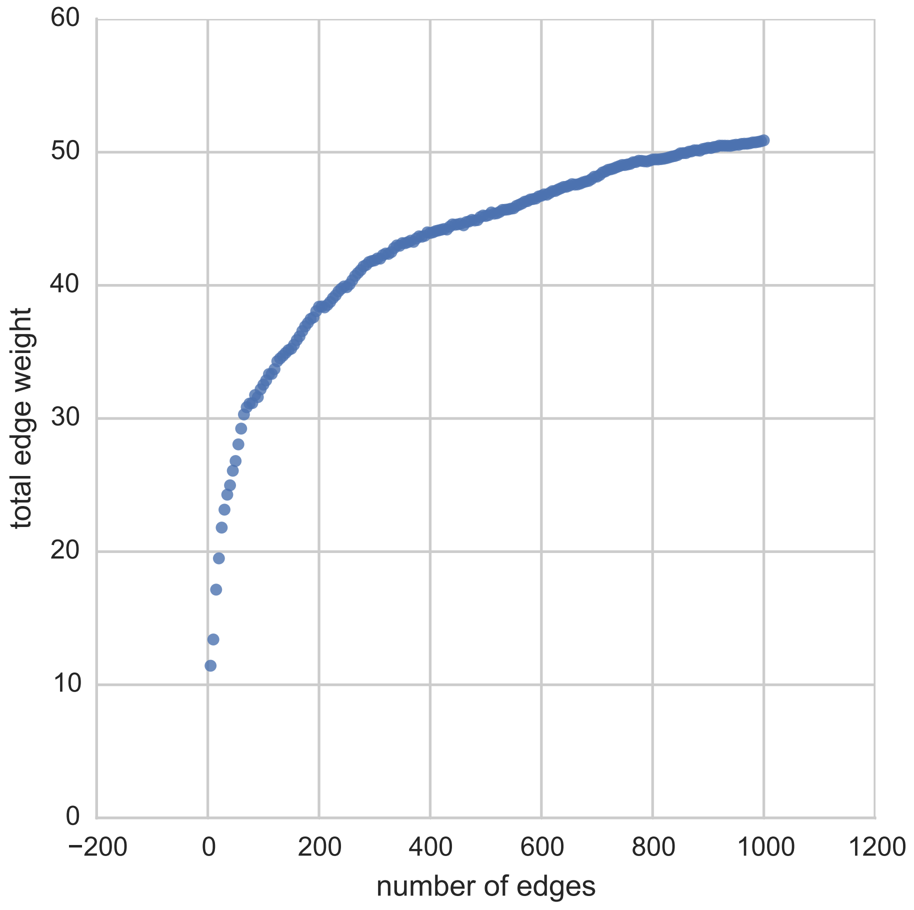


Figure 1: Total edge weight as a function of number of edges. Vectors are length 500.

vector increases, the chance of two random vectors being non-orthogonal decreases, making the edge weights more accurate.

To test this hypothesis, we provide a HoloGraph and a ProbGraph with the same random training data. We vary the number of different nodes in the training data, as well as the dimensionality of the HoloGraph vectors. If the HoloGraph implementation is sound, we expect the recovered edge weights after training to be very similar to the edge weights of the ProbGraph. However, this similarity may degrade with many nodes and low dimensionality. As shown in figure #, the results match our expectations.

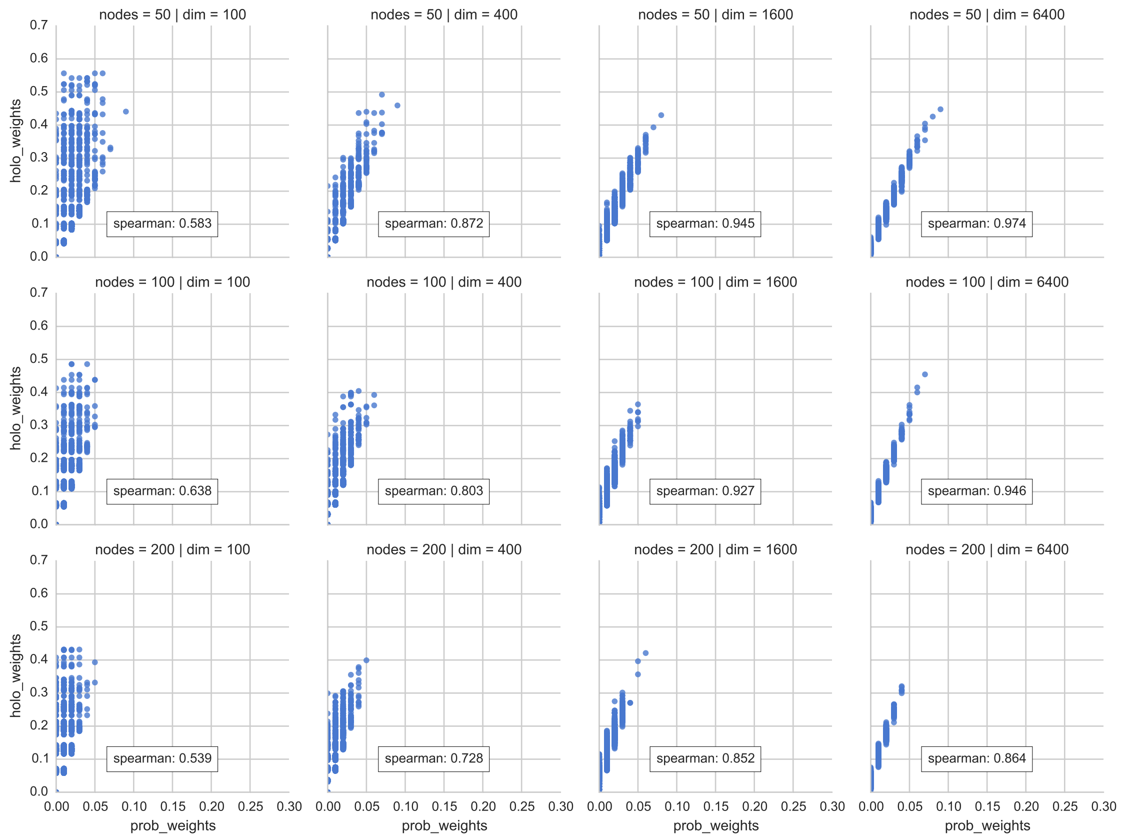


Figure 2: Correlation of sparse vector and probabilistic graph edge weights with varying number of nodes and vector dimensionality. Node count increases from top to bottom. Dimensionality increases from left to right.

TODO: Effect of initial row vector size?

Nümila

To demonstrate how a BindGraph can be used as a component of another model, we created a simple language acquisition model, based on previous graphical models (Solan et al. 2005; Kolodny, Lotem, and Edelman 2015) and chunking models (McCauley and Christiansen 2011). Like these previous models, Nümila reduces the problem of language acquisition to the much simpler problem of producing grammatical utterances based on statistical patterns in speech, specifically transitional probabilities between words and phrases. In reality, language acquisition is heavily dependent on the semantic and social aspects of language (Tomasello 2003; Goldstein et al. 2010), aspects which the present model does not capture. However it is generally agreed that linguistic pattern recognition plays at least some role in language acquisition; thus, the present model can be seen as a baseline that could be improved upon by enriching the input to the model with environmental cues. TODO We discuss ways that these cues could be incorporated into a graphical model.

The model is theoretically aligned with usage-based psycholinguistics, a field that emphasizes psychological plausibility and observed behavior, studying language as it occurs in the world (sometimes referred to as "surface phenomena" [ciation]). The model is highly incremental, processing one utterance at a time. For each utterance, the model applies a parsing algorithm that simultaneously assigns structure to and learns from the utterance. Producing utterances relies on the same basic principles as parsing an utterance, and the representations underlying all processing take the same form as the main knowledge base, i.e. a graph with words and chunks connected by temporal adjacency.

In line with many usage-based psycholinguistic models, the present model is fairly simplistic (although see **???**). It does not explicitly represent abstractions such as syntactic categories and dependencies. However, this is a characteristic of the particular model, and not of graphical models in general. Although we suggest the possibility that rule-like behavior could emerge from the present model without explicit rule-like representations, we do not make any strong theoretical claims about human linguistic representations.

Graphical model

The model represents its knowledge using a BindGraph as described in section #. Words and phrases ("chunks") are stored as nodes, and transitional probabilities between those elements as edges.

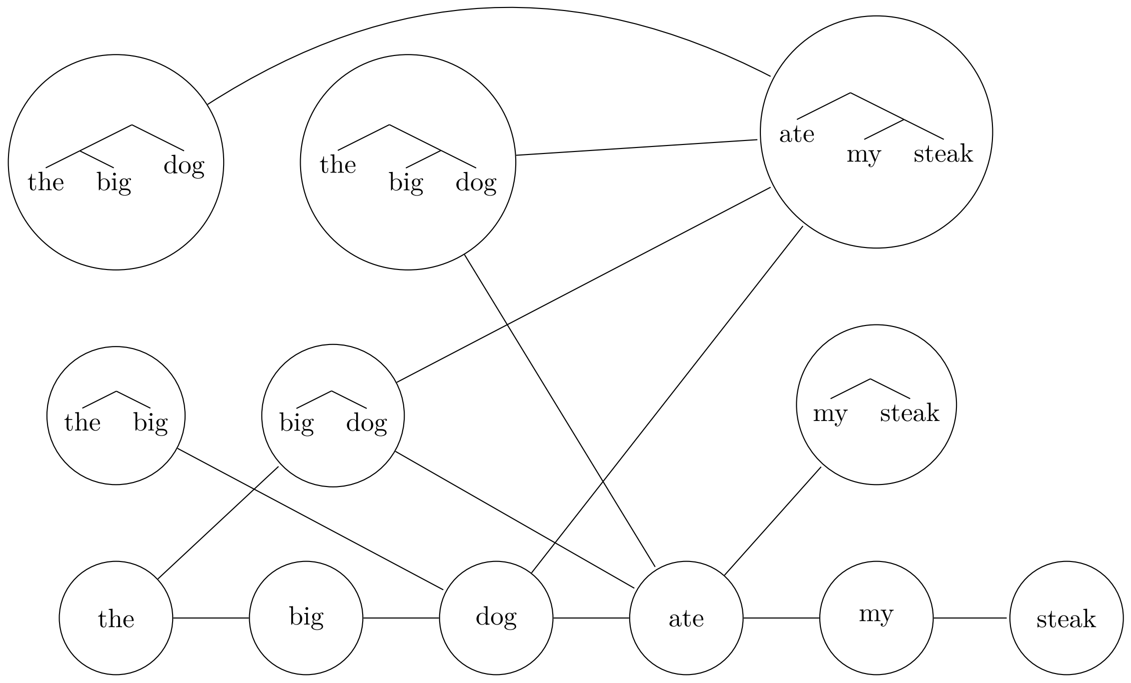


Figure 3: A simplified graph representing the utterance "the big dog ate my steak". For display, FTP and BTP edges are represented as a single undirected edge.

Edges

The model has two edge-types representing forward and backward transitional probabilities, that is the probability of one word following or preceding a given word: $p(w_i = x|w_{i-1} = y)$ and

$p(w_i = x|w_{i+1} = y)$ respectively. Although forward transitional probability (FTP) is the standard in N-gram models, some evidence suggests that infants are more sensitive to BTP (Pelucchi, Hay, and Saffran 2009), and previous language acquisition models have been more successful when employing it (McCauley and Christiansen 2011). To examine the relative contribution of each type of TP, we make their relative weight an adjustable parameter. Although ADIOS and U-MILA have only one type of temporal edge (co-occurrence count), their learning algorithms compute something very similar to FTP and BTP [TODO proof?]. By using two edge types, we build this computation into the representational machinery.

Merge

When two nodes (initially words) are determined to co-occur at an unexpectedly high frequency (see below), the graph's bind function is applied to create a new node. As discussed above, the bind function, $fbind$, is a parameter of the graph, and must be a function from a list of nodes to a single node. As a simplifying assumption, we follow U-MILA by only considering binary binds; thus the function is of type $N \times N \rightarrow N$. Importantly, we do not make the theoretical claim that linguistic composition must be binary (as others have, c.f. Everaert et al. 2015); this decision was made for ease of modeling. We implement two such bind functions, one hierarchical and the other flat. Given arguments `[A B]` and `[C D]`, hierarchical bind returns `[[A B] [C D]]`, whereas flat bind returns `[A B C D]`. Both hierarchical (Solan et al. 2005) and flat (Kolodny, Lotem, and Edelman 2015, McCauley and Christiansen (2011)) merge rules have been used in previous models.

In the simplest case, the bind function determines only the identity of the new node. This quality alone has important ramifications. Hierarchical bind is a bijective function; that is, there is a one-to-one mapping from inputs to outputs. Conversely, flat bind is not bijective because multiple inputs can produce the same output. For example, if `[A B C]` and `D` occur together frequently, a new node `[A B C D]` will be created and added to the graph. Later on, if `[A B]` and `[C D]` are bound, we will get the existing node, `[A B C D]` with all its learned edge weights. In more practical terms, a model using a flat bind rule will treat every instance of a given string e.g. "Psychology department chair" as the same entity. Although there are clear semantic reasons why a flat bind function would be undesirable, it is not clear to what extent hierarchical information will be useful for the purely grammatical tasks we test our models with TODO.

The full power of the bind function, however, comes from its ability to construct initial edge weights for the new node, allows truly compositional structure, where the behavior of larger units is predictable based on its constituents. This is critical for syntactic processing: Even if you had never heard the phrase "honest politician", you could still predict its syntactic behavior. In a graph, the syntactic behavior of a node is represented in its edge weights. Thus, predicting syntactic behavior comes down to constructing an initial edge profile for the newly created node based on the edge profiles of its element nodes. This function can be specified by the modeler or learned. [TODO specified bind function]

Learning

The model constructs a Higraph given an initially blank slate by processing each utterance, one by one. Thus, the model has more limited memory resources than both ADIOS and U-MILA. The graph is constructed with three operations: (1) adding newly discovered base tokens to the graph, (2) increasing weights between nodes in the graph, and (3) creating new nodes by binding existing nodes. We implement two processing algorithms that employ these basic operations to learn from an utterance. The first is meant to replicate U-MILA's bottom up chunking mechanism, learning transitional probabilities between all possible nodes in the utterance. The second is inspired by the Now-or-Never bottleneck (Christiansen and Chater 2015), incorporating an even more severe memory constraint, and building up a structural interpretation of the utterance word by word.

FullParse

The FullParse algorithm is similar to U-MILA's bottom up learning algorithm in that it finds all pairs of adjacent nodes (potentially overlapping) in the utterance and then applies a learning process to the pair. The algorithms differ in the edge weights that are updated, and the criteria for creating a chunk. For every pair of nodes (X, Y) such that X directly precedes Y in the utterance:

1. Increase weights
   (a) Forward transition probability $E_F(X, Y)$
   (b) Backward transition probability $E_B(Y, X)$
2. Attempt to create a chunk
   (a) Check that binding the pair would not result in a node already in the graph
   (b) Check that the *chunkiness* of the pair exceeds a fixed threshold, where *chunkiness* is the geometric mean of the forward and backward transition probabilities between the nodes: $\sqrt{E_F(X, Y) \cdot E_B(Y, X)}$
   (c) Create a new node by binding X and Y
   (d) Add this node to the graph

Unlike U-MILA's algorithm, this algorithm requires the full utterance to be in memory before any processing is done. This is because the specific algorithm isn't meant to be cognitively realistic [TODO should this be discussed, perhaps footnoted?]

GreedyParse

The GreedyParse algorithm follows the same basic principles as FullParse in that it is based on updating transitional probability edges and binding nodes. However, unlike FullParse, GreedyParse incorporates severe memory limitations and online processing restraints in line with the Now-or-Never bottleneck (Christiansen and Chater 2015). In contrast to FullParse, which finds all possible structures for an utterance given the current graph, GreedyParse finds a single structure by making a sequence of locally optimal decisions, hence "Greedy". Upon receiving each word it can create at most one chunk and the nodes used in this chunk can not be used later in a different chunk. Thus, the algorithm may not assign the optimal structure to the utterance.

```
# Shift.
append a new token onto memory
if token is not in graph:
    add token to graph
fi
update weights between the previous token and the new token

# Chunk.
select pair of adjacent nodes with maximum chunkiness
if chunkiness of pair exceeds threshold:
    create chunk by binding the two nodes
    if chunk is not in graph:
        add chunk to graph
        update weights between new chunk and adjacent nodes
    fi
else:
    remove the oldest node from memory
fi
```

Generalization

The model is essentially a bigram models with variable length elements. Thus, like any N-gram model, it will suffer from the problem of data sparsity. Incorporating variable-length units increases this problem dramatically because the number of tracked elements is no longer limited to the vocabulary size. A generalization strategy is essential to counteract this sparsity problem. In a classical linguistic view, generalization is accomplished with categories such as VERB and NOUN. The reality of parts of speech is a basic assumption in linguistics, and acquisition models often use parts of speech as input [Bod (2009); TODO]. The problem of learning these categories, on the other hand, has received less attention in linguistics. Algorithms for part of speech induction based on distributional statistics have had some success (Schütze 1995; Clark 2000). However these models come from natural language processing, employing computationally intensive employing batch processing algorithms.

Some work has sought to connect POS induction models with syntax induction models, learning language structure in stages (Klein and Manning 2005). However, as a cognitive model, this approach assumes that word categories are learned and finalized before any syntax is learned. While category learning may begin earlier, there are good reasons for the learning of categories and syntax to occur simultaneously. Seeing as a large motivation for categories is the role they play in syntactic pattern recognition, it would be desirable for the syntactic learning process to affect category learning. While linear context may provide a decent cue to syntactic category, the role a word plays in higher level syntactic patterns may hold more information. For this reason, we pursue a single stage approach in which word-categories and phrasal patterns are learned in tandem.

Another common assumption that deserves examination is the one that syntactic categories are explicit and distinct. Having suffered through explicit instruction on the rules of grammar, most students are able to classify most words into the basic parts of speech. However, it is not clear to what extent these categories are the product of normal language acquisition as opposed to academic analysis. Furthermore, even for the student who can correctly label words, it is unclear to what extent this knowledge is called upon in every day language use. Given this, we take the path of least commitment by not attempting to model explicit categories. Instead, similarity based generalization is done on the fly, each time it is called for.

We implement a simple, domain-neutral, generalization algorithm for the HoloGraph. The algorithm can be applied whenever an edge weight is recovered. The basic idea is that if a node x is similar to a node y in many ways, it might also be similar in other ways. In vector space, we create a generalized row vector as the sum of the all other nodes' row vectors weighted by the similarity of each node to the target node. The generalized row vector for edge e of $\mathbf{n}_0$ is

$$\sum sim(\mathbf{n}_0, \mathbf{n}_i) \cdot R_{e,i}$$

where $R_{e,i}$ is the row vector for n_i and $sim(x, y)$ is the geometric mean of pairwise cosine similarities for each row vector of x and y respectively. UGH

Compositionality

Presumably, experience with pairs of similar words informs your expectations for how the syntactic role of this novel phrase. The standard linguistic explanation involves rules and categories: "honest" is an `Adj`; "politician" is a `N`; and thus by the rule `N' -> Adj N`, "honest politician" is an `N'`. The extensive regularity of language implies that such theories are at least a useful computational-level theory (as Marr 1982 suggests). However, it is an open question whether adult speakers truly represent rules and discrete categories. It's possible that rule-like behavior emerges from a system that explicitly represents only relationships between individual items [citation].

- rules and hard categories not characteristic of brains
- computational vs algorithmic

- how to learn?

perhaps based on your experience with other adjective-noun pairs. the new nodes edges would be similar to other nodes that were composed of similar elements.

- talk a lot about baroni

Testing the model on natural language

To test the model, we use naturalistic child directed speech. We use corpora prepared by Phillips and Pearl (2014). For each corpus, the input can be tokenized by word, syllable, or phoneme, giving a total of $7 \times 3 = 21$ different corpora. All models are trained on the first 4000 utterances of each corpus, and tested on the next 1000. We test several instantiations of Nümila using different BindGraph implementations, parsing algorithms, and bind functions.

Experiment 1: Grammaticality judgment

As a first test, we use the common task of discriminating grammatical from ungrammatical utterances. This task is appealing because it is theory agnostic (unlike evaluating tree structures) and it does not require that the model produce normalized probabilities (unlike perplexity).

Generating an acceptability score

Statistical language modeling is sometimes equated with determining the probability of word sequences (c.f. Goodman 2001), something that Nümila cannot easily do given that outgoing edges for one node (labeled e.g. FTP) are not required to sum to 1, and are thus not probabilities. However, many tasks that employ utterance probability can be performed just as well with scores that are only proportional to a true probability.

In a generative language model, the probability of an utterance is the sum of the probabilities of all possible ways to produce the utterance (e.g. all tree structures). The probability of each structure is the product of the probabilities of every rule that is applied in creating the utterance. With a PCFG, the rules are all of the form $NT \rightarrow \alpha$, where NT is a nonterminal symbol (representing one branch of the tree, a constituent) and α is a sequence of symbols, either terminal or nonterminal. With an N-gram model, on the other hand, the rules are all of the form $\alpha \rightarrow \alpha \cdot w$, where α is the $n-1$ most recent words and w is the next word.

Because Nümila incorporates structural elements (chunks) and transitional probabilities, it uses both types of rules. For chunks, the PCFG rule is applied; however, because each node has exactly one compositional structure, the rule probability is always 1. When an utterance has a series of nodes that cannot be combined, the bigram rule is applied: For each adjacent pair of nodes, (X, Y), we apply the rule $X \rightarrow X \cdot Y$ with probability proportional to $E_F(X, Y)$. The result is simply the product of FTPS between each pair of nodes spanning the utterance. [TODO similar to something, but what?!] Finally, to avoid penalizing longer utterances, we take the result to the $n-1$ root where n is the length of the utterance.

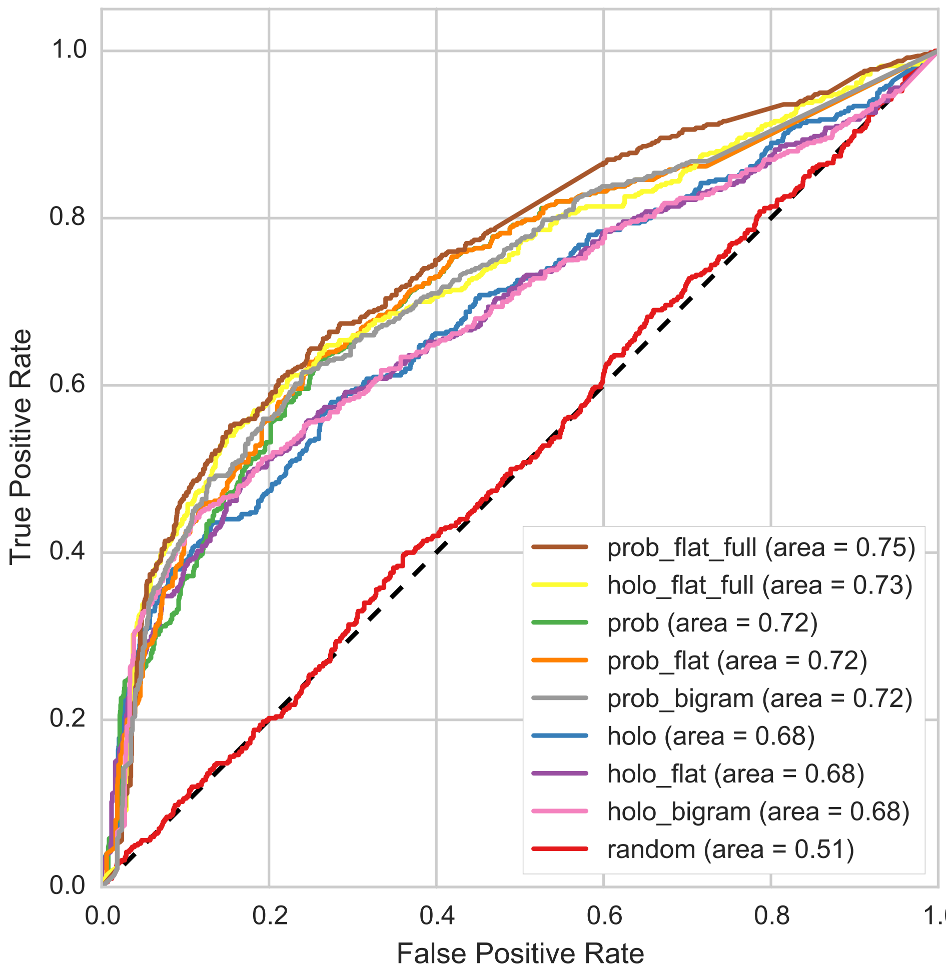
Given a function that assigns scores to a single parse of an utterance, it is straightforward to create a function that assigns scores to the utterance itself. With a PCFG (where the scores are probabilities) the probability of the utterance is the sum of the probabilities of each parse. This is a result of the assumption that the utterance is generated by the PCFG model, along with Kolmogorov's third axiom that the probability of the union of independent events is the sum of the probabilities of each event. Although Nümila's scores are not true probabilities, we apply the same rule. That is, the score of an utterance is the sum of the scores for all parses of that utterance. [TODO does this maintain proportional probability?]

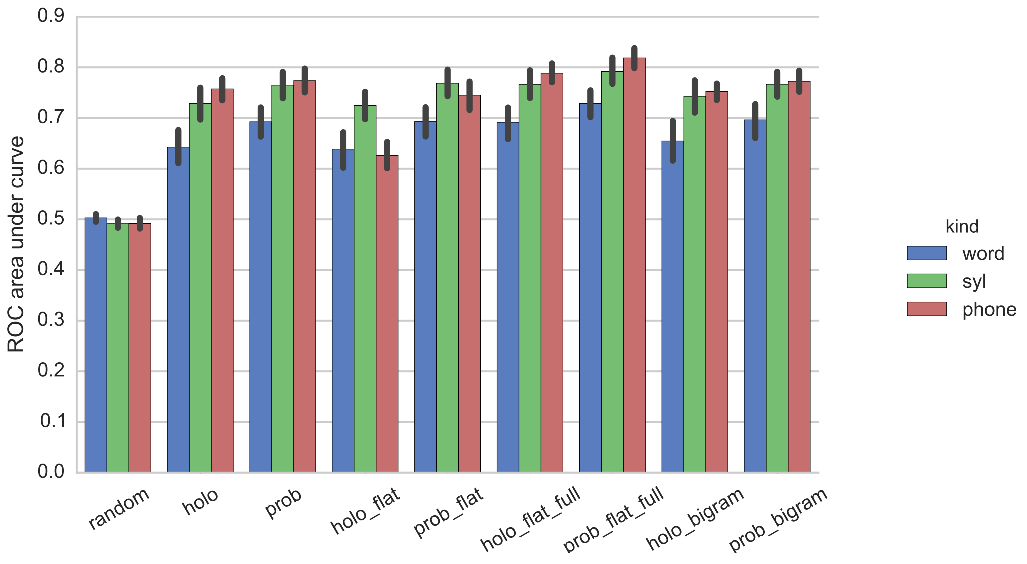
Preparation of stimuli and analysis of performance

To construct a test corpus, we first take # unseen utterances from the corpus, which are labeled "grammatical". For each utterance, we create a set of altered utterances, each with one adjacent pair of tokens swapped. For example, given "the big dog", we create "big the dog" and "the dog big". These altered utterances are added to the test corpus with the label "ungrammatical". The models task is to separate grammatical from ungrammatical. Often, this task is modeled by setting a threshold, all utterances being predicted to be grammatical. However, it is unclear how to set such a threshold without either specifying it arbitrarily or giving the model access to the test labels. Thus, we employ a metric from signal detection theory, the Receiver Operator Characteristic.

The ROC curve plots true positive rate against false positive rate. As the acceptability threshold is lowered, both values will increase. With random scores, they will increase at the same rate, resulting in a line at $y = x$. A model that captures some regularities in the data, however, will initially have a faster increasing true positive rate than a false positive rate because the high-scored utterances will tend to be grammatical ones. This results in a higher total area under the curve, a scalar metric that is often used as broad metric of the power of a binary classifier. This measure is closely related to precision and recall, but has the benefit of allowing interpolation between data points, resulting in a smoother curve (Davis and Goadrich 2006).

Results


True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
prob_flat_full (area = 0.75)
holo_flat_full (area = 0.73)
prob (area = 0.72)
prob_flat (area = 0.72)
prob_bigram (area = 0.72)
holo (area = 0.68)
holo_flat (area = 0.68)
holo_bigram (area = 0.68)
random (area = 0.51)


ROC area under curve
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
random
holo
prob
holo_flat
prob_flat
holo_flat_full
prob_flat_full
holo_bigram
prob_bigram
kind
word
syl
phone

Experiment 2: Production

As a second test, we use the task of ordering a bag of words—a proxy for production. A more direct test of production would be to generate utterances without any input, for example, by concatenating nodes in the graph based on transitional probabilities. However, this task has two disadvantages. First, it is difficult to evaluate the acceptability of generated utterances without querying human subjects. Second, utterance production in humans likely involves semantic as well as structural information, the first of which the present model does not attempt to capture. To avoid these problems, we follow previous work (Chang, Lieven, and Tomasello 2008; McCauley and Christiansen 2014) by using a word-ordering task to isolate structural knowledge. A bag of words is taken as an approximate representation of the thought a speaker wishes to convey; speaking then becomes simply the task of saying the words in the right order.

Ordering a bag of words

We treat ordering a bag of words as an optimization problem, using the acceptability score described above as a utility function. The optimal but inefficient strategy is to enumerate all possible orderings of the words and choose the one with the highest acceptability score. However, with $n!$ possible orderings, this becomes intractable for longer utterances. As with learning, we propose a greedy algorithm to approximate the optimal solution. Typically such an algorithm starts from the beginning of the utterance and iteratively appends the best word or chunk to the end, producing the utterance in the same order it was spoken (e.g. McCauley and Christiansen 2014; Kolodny, Lotem, and Edelman 2015). This parallel has some theoretical appeal, however, it not clear that utterances are planned in the same order that they are spoken. Indeed, research in predictive sentence processing indicates that listeners actively predict upcoming clauses [citation]—it is thus reasonable to think that speakers may plan ahead beyond the next few words. Lacking strong theoretical motivation for purely incremental sentence planning, we explore a more flexible approach.

The algorithm begins by greedily constructing chunks using the input nodes. When no more chunks can be made, the chunks are combined to form an utterance. This is done by iteratively adding a node to either the beginning or the end of the utterance, whichever maximizes chunkiness between the new adjacent pair.

```
# Create chunks.
while a chunk can be made:
    select pair of nodes with highest chunkiness
    if chunkiness < threshold:
        break while
    replace the pair with the chunk constructed by binding the pair

# Create utterance.
utterance = []
add chunk with highest chunkiness to utterance
while there are nodes left:
    select the node that has the highest chunkiness with either
      the first or last element of the utterance
    add the node to the beginning or end of the utterance
```

Preparation of stimuli and analysis of performance

To test the model on this task, we take an unseen item from the corpus, convert it into a bag of words, and then compare the model's ordering to the original utterance. A simple comparison strategy is to assign a score of 1 if the model's output perfectly matches the original, and 0 otherwise (as in McCauley and Christiansen 2014). However, this metric selectively lowers the average score of longer utterances, which have $n!$ possible orderings. If the average score varies across utterance lengths, utterances of different lengths will have varying discrimination power (in the extreme, no discrimination power if all models fail all utterances of a given length). Given this, we use the BLEU metric (Papineni et al. 2002), which is more agnostic to utterance length. Specifically, we use the percentage of bigrams that are shared between the two utterances.

Results

The current results are placeholders.

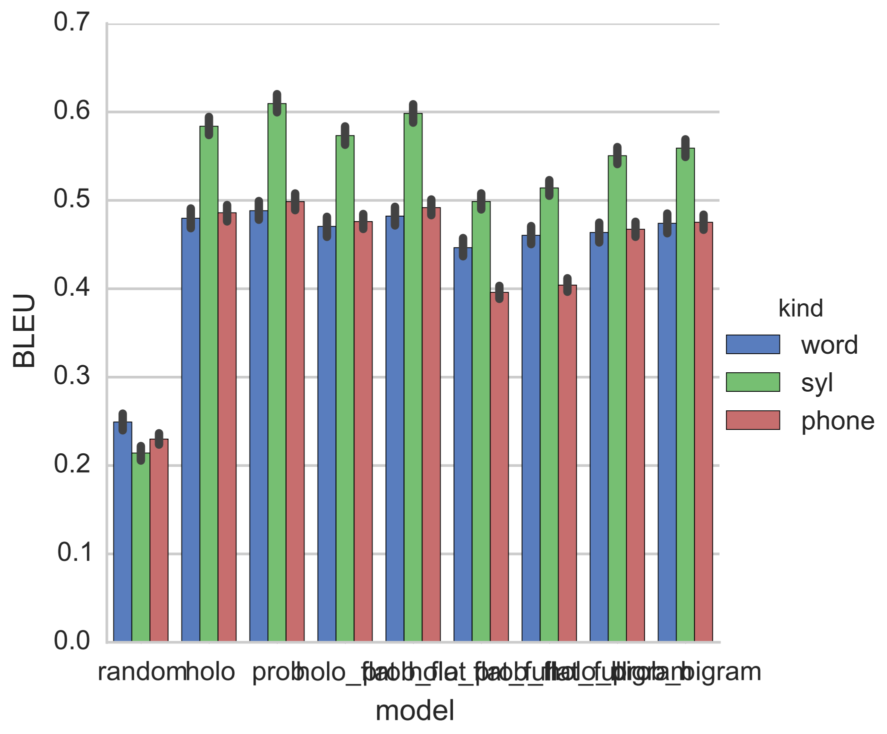


Figure 4: Production results.

Experiment 3: Generalization

Although generalization does not appear to improve … TODO

- can't test generalization algorithm on full corpus because computationally intractable
- dynamic doesn't help on natural language

More results?

- CFGs of increasing complexity
- contrived examples

Introspection

- MDS of 10–50th most frequent words
- number of chunks over time

Conclusion

This was fun.

References

Bannard, Colin, Elena Lieven, and Michael Tomasello. 2009. "Modeling Children's Early Grammatical Knowledge." *Proceedings of the National Academy of Sciences* 106 (41). National Acad Sciences: 17284–9.

Basile, Pierpaolo, Annalina Caputo, and Giovanni Semeraro. 2011. "Encoding Syntactic Dependencies by Vector Permutation." In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*, 43–51. Association for Computational Linguistics.

Bod, Rens. 2009. "From Exemplar to Grammar: A Probabilistic Analogy-Based Model of Language Learning." *Cognitive Science* 33 (5). Wiley Online Library: 752–93.

Chang, Franklin, Elena Lieven, and Michael Tomasello. 2008. "Automatic Evaluation of Syntactic Learners in Typologically-Different Languages." *Cognitive Systems Research* 9 (3). Elsevier: 198–213.

Chomsky, Noam. 1965. *Aspects of the Theory of Syntax.* MIT Press.

———. 1995. *The Minimalist Program.* Vol. 1765. Cambridge Univ Press.

Christiansen, Morten H, and Nick Chater. 2015. "The Now-or-Never Bottleneck: A Fundamental Constraint on Language." *Behavioral and Brain Sciences.* Cambridge Univ Press, 1–52.

Clark, Alexander. 2000. "Inducing Syntactic Categories by Context Distribution Clustering." In *Proceedings of the 2nd Workshop on Learning Language in Logic and the 4th Conference on Computational Natural Language Learning-Volume 7*, 91–94. Association for Computational Linguistics.

Davis, Jesse, and Mark Goadrich. 2006. "The Relationship Between Precision-Recall and ROC Curves." In *Proceedings of the 23rd International Conference on Machine Learning*, 233–40. ACM.

Everaert, Martin BH, Marinus AC Huybregts, Noam Chomsky, Robert C Berwick, and Johan J Bolhuis. 2015. "Structures, Not Strings: Linguistics as Part of the Cognitive Sciences." *Trends in Cognitive Sciences* 19 (12). Elsevier: 729–43.

Firth, John R. 1957. "A Synopsis of Linguistic Theory, 1930-1955." Blackwell.

Goldstein, Michael H, Heidi R Waterfall, Arnon Lotem, Joseph Y Halpern, Jennifer A Schwade, Luca Onnis, and Shimon Edelman. 2010. "General Cognitive Principles for Learning Structure in Time and Space." *Trends in Cognitive Sciences* 14 (6). Elsevier: 249–58.

Goodman, Joshua T. 2001. "A Bit of Progress in Language Modeling." *Computer Speech & Language* 15 (4). Elsevier: 403–34.

Griffiths, Thomas L, Mark Steyvers, and Joshua B Tenenbaum. 2007. "Topics in Semantic Representation." *Psychological Review* 114 (2). American Psychological Association: 211.

Hagoort, Peter. 2004. "How the Brain Solves the Binding Problem for Language." In *28th International Congress of Psychology.*

Harel, David. 1988. "On Visual Formalisms." *Communications of the ACM* 31 (5). ACM: 514–30.

Hudson, Richard. 2003. "Word Grammar." Walter de Gruyter.

Jones, Michael N, and Douglas JK Mewhort. 2007. "Representing Word Meaning and Order Information in a Composite Holographic Lexicon." *Psychological Review* 114 (1). American Psychological Association: 1.

Kanerva, Pentti, Jan Kristofersson, and Anders Holst. 2000. "Random Indexing of Text Samples for Latent Semantic Analysis." In *Proceedings of the 22nd Annual Conference of the Cognitive Science Society*. Vol. 1036. Citeseer.

Karlgren, Jussi, and Magnus Sahlgren. 2001. "From Words to Understanding." In *Foundations of Real-World Intelligence*, edited by Y Uesaka, Pentti Kanerva, and H Asoh. Stanford: CSLI Publications.

Katz, Slava M. 1987. "Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer." *Acoustics, Speech and Signal Processing, IEEE Transactions on* 35 (3). IEEE: 400–401.

Klein, Dan, and Christopher D Manning. 2005. "Natural Language Grammar Induction with a Generative Constituent-Context Model." *Pattern Recognition* 38 (9). Elsevier: 1407–19.

Kolodny, Oren, Arnon Lotem, and Shimon Edelman. 2015. "Learning a Generative Probabilistic Grammar of Experience: A Process-Level Model of Language Acquisition." *Cognitive Science* 39 (2). Wiley Online Library: 227–67.

Lamb, Sydney M. 1999. *Pathways of the Brain: The Neurocognitive Basis of Language*. Vol. 170. John Benjamins Publishing.

Landauer, Thomas K., and Susan T. Dumais. 1997. "A Solution to Plato's Problem: The Latent Semantic Analysis Theory of Acquisition, Induction, and Representation of Knowledge." *Psychological Review* 104 (2). American Psychological Association (APA): 211–40.

Lund, Kevin, and Curt Burgess. 1996. "Producing High-Dimensional Semantic Spaces from Lexical Co-Occurrence." *Behavior Research Methods, Instruments, & Computers* 28 (2). Springer: 203–8.

Marr, David. 1982. "Vision: A Computational Investigation into the Human Representation and Processing of Visual Information." San Francisco: WH Freeman.

McCauley, Stewart M, and Morten H Christiansen. 2011. "Learning Simple Statistics for Language Comprehension and Production: The CAPPUCCINO Model." In *Proceedings of the 33rd Annual Conference of the Cognitive Science Society*, 1619–24.

———. 2014. "Acquiring Formulaic Language: A Computational Model." *The Mental Lexicon* 9 (3). John Benjamins Publishing Company: 419–36.

Mitchell, Jeff, and Mirella Lapata. 2010. "Composition in Distributional Models of Semantics." *Cognitive Science* 34 (8). Wiley Online Library: 1388–1429.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. "BLEU: A Method for Automatic Evaluation of Machine Translation." In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, 311–18. Association for Computational Linguistics.

Pelucchi, Bruna, Jessica F Hay, and Jenny R Saffran. 2009. "Learning in Reverse: Eight-Month-Old Infants Track Backward Transitional Probabilities." *Cognition* 113 (2). Elsevier: 244–47.

Phillips, Lawrence, and Lisa Pearl. 2014. "Bayesian Inference as a Viable Cross-Linguistic Word Segmentation Strategy: It's All About What's Useful." In *Proceedings of the 36th Annual Conference of the Cognitive Science Society*, 2775–80. Citeseer.

Plate, Tony, and others. 1995. "Holographic Reduced Representations." *Neural Networks, IEEE Transactions on* 6 (3). IEEE: 623–41.

Sahlgren, Magnus. 2005. "An Introduction to Random Indexing." In *Methods and Applications of Semantic Indexing Workshop at the 7th International Conference on Terminology and Knowledge Engineering, TKE*. Vol. 5.

Schütze, Hinrich. 1995. "Distributional Part-of-Speech Tagging." In *Proceedings of the Seventh Conference on European Chapter of the Association for Computational Linguistics*, 141–48. Morgan Kaufmann Publishers Inc.

Smolensky, Paul, and Géraldine Legendre. 2006. *The Harmonic Mind: From Neural Computation to Optimality-Theoretic Grammar (Vol. 1: Cognitive Architecture)*. MIT Press.

Solan, Zach, David Horn, Eytan Ruppin, and Shimon Edelman. 2005. "Unsupervised Learning of Natural Languages." *Proceedings of the National Academy of Sciences of the United States of America* 102 (33). National Acad Sciences: 11629–34.

Stabler, Edward. 1996. "Derivational Minimalism." In *Logical Aspects of Computational Linguistics*, 68–95. Springer.

Steedman, Mark. 2000. *The Syntactic Process*. Vol. 24. MIT Press.

Tomasello, Michael. 2003. "On the Different Origins of Symbols and Grammar." *Evolution of Language* 3. Oxford University Press: 94–110.