flc37@cornell.edu

Lorem ipsum dolor sit amet

Fred Callaway
Cornell University

22nd April 2016

Structured representation has a critical role in cognition, thus modeling structured representation has a critical role in cognitive science. However, subfields have generally developed their own tools for representing the structures of their systems of interest. This makes it difficult to connect models of different cognitive phenomena, as well as models posed at different levels of analysis. We suggest that graphs can unite disparate models with a common representation. Additionally, we describe an implementation of a graph using vectors in a high-dimensional space. I have no idea what I'm doing. Please help.

1 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. These internal models go beyond Skinnerian stimulus-response pairings discovered through reinfocement learning: They form a coherent and veridical view of the represented system, improving the agent's ability to interact with that system (Edelman 2008). These internal models play such an important role in cognition that the study of their form, acquisition, and use makes up the majority of work in cognitive science.

Given that the goal of science is to create models of the world, cognitive scientists are presented with a unique challenge: modeling internal models, or representing representations. As in other scientific fields, a model of internal models should be systematic and unified. It should explain how the details of specific internal models (e.g. of language) reflect general principles of mind/brain representations. How can we go about constructing such a model? One possible approach to this problem is to begin with a model of the system itself, and then attempt to explain how the mind/brain might construct this model. (relate to Marr) This approach is often advocated in linguistics, where ... (Everaert et al. 2015) TODO

As a result, different fields of cognitive science have each designed their own representational frameworks, often with no attempt to connect the representations

of different fields (and sometimes knowingly avoiding such connections e.g. Chomsky 2004). Without a method for connecting insights created by the different fields, we are left with only isolated glimpses into the functioning of the brain and mind. If we wish to pursue an integrated and complete theory of cognition, we must first develop tools for connecting the models, and therefore the representations, of widely varying fields.

A significant challenge for this endeavor is that representation can–indeed must–be described at multiple levels of abstraction (Edelman 2008; Marr 1982). Although the representations of terrestrial animals must ultimately be implemented with 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 TODO: which anderson paper do I want?, however this does not invalidate the explanatory power of any one functional explanation. Given that models differ in both domain and level of analysis, uniting these models with a common representational backbone is a task as formidable as it is essential.

We suggest the graph as a potential candidate for this representational backbone. Graphs are domain general, thus they can be used to model disparate cognitive phenomena. Additionally, with augmentations such as labeled edges, they can represent models of widely varying complexity, posed at any level of analysis. Brains themselves closely resemble 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.

If we represent models of different cognitive phenomena posed at different levels of analysis in one common format, it will be far easier to connect them. Two models at the same level may draw insight from each other, while a lower level model may suggest an implementation of a higher level model. 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 see the parallel to her own work, and adopt the same neural model.

2 Graphical models of language

The earliest statistical language models are based on transitional probabilities [citation]. 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.

3 BindGraph (sorry)

Baesd largely on the graphs of U-MILA and ADIOS, we specify an augmented graphical data structure intended to represent domains that are characterized 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 that (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, p. 2). The notion of composition has an even richer history: Nearly all modern syntactic theories take some sort of binding funcion as a fundamental operation. Whether it is called 'Merge' (Chomsky 1995), 'function application' (Steedman 2000), or 'Unification' (Hagoort 2004), the principle of combining two elements to form one element remains. Despite this similarity, these three frameworks use widely differing representational machinery. As a result, it is sometimes difficult to specify the differences and similarities between the theories. Graphs may provide a general representational framework with which different linguistic theories could be compared.

A BindGraph is based on a weighted, labeled, directed, multigraph. That is, a node may have multiple weighted 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. Additionally, a modeler may augment the standard BindGraph with additional operations for generalization or pattern recognition.

4 Graphs in space

In this section, we describe an implementation of a BindGraph with 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 are well understood and frequently used, providing the modelers with abundant resources when attempting to define a new operation or algorithm (e.g. a bind function). Second, a distributed representation of a graph could provide a bridge between the symbolic representations of computational-level models (such as are found in linguistics) with the distributed representations proposed by the PDP and Connectionist frameworks (see Smolensky and Legendre 2006, for an alternative approach).

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, we adopt an incremental dimensionality reduction technique that has been used effectively in distributional models: *random indexing* (see Sahlgren 2005, for an accessible review). The resulting data structure closely mimics the behavior of an adjacency matrix representation when the space is of a sufficiently high dimension. We define the $fbump$ and $fweight$ operations, and suggest one possible domain-general $fbind$ function as well as a generalization technique.

4.1 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 word-context associations are represented with 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). To the extent that words with similar meaning occur in the same contexts, the rows for similar words will be somewhat similar. Typically, cosine similarity is used to quantify this similarity.

With the size of modern data sets, the raw context vectors are generally too large and sparse to use effectively. 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) demonstrates 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 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).

4.2 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

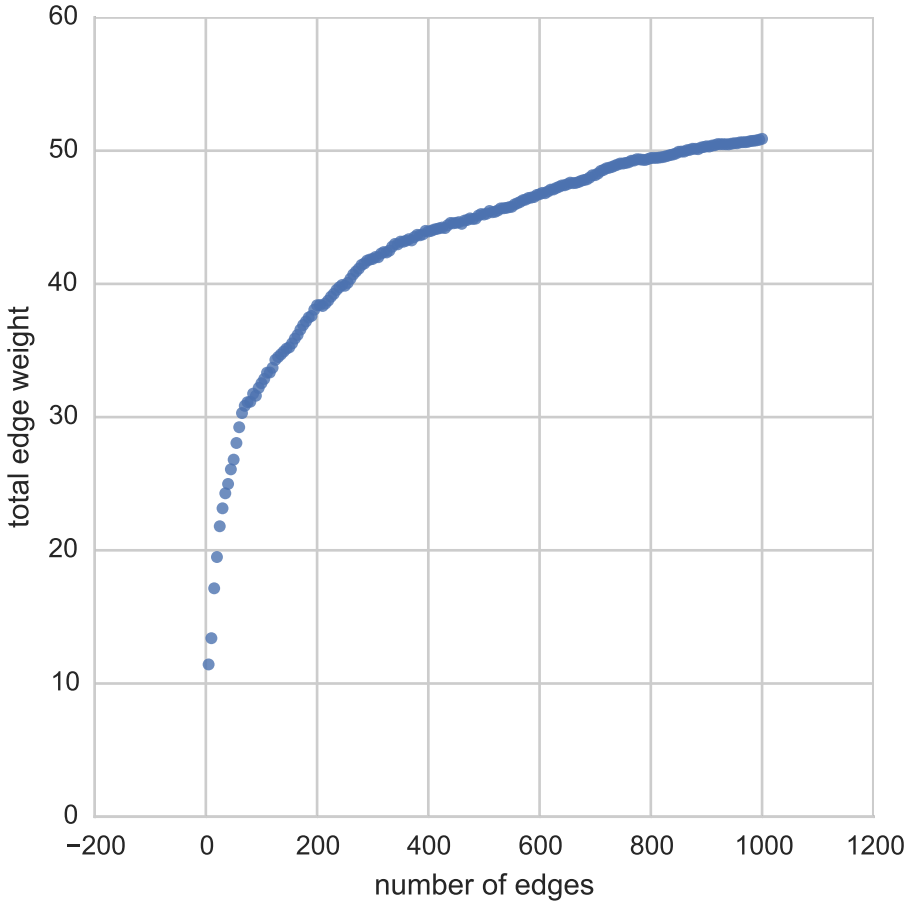

figure #.

4.2.1 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.]

4.2.2 Generalization

Storing information in a structured form allows an agent to inform her decisions with past experience. However, for this knowledge to be widely applicable in the natural world, it cannot be rigid. This is especially true in complex systems such as language, where the exact same situation is unlikely to occur twice. When attempting to understand and react to events in these domains, an agent must generalize based on experiences similar but not identical to the current situation. For example, upon seeing a new breed of dog, you would likely still recognize it as a dog, and thus avoid leaving your meal unatended in its reach.

Traditionally, psychology and linguistics has assumed that *categories* are the driving force behind generalization (Pothos and Wills 2011). Under this view, generalization is mediated by the application of discrete labels to groups of stimuli (as in exemplar models) or features (as in prototype models). For example, in the above example, you first identify the unfamiliar animal as a dog, and only then infer that it, like other dogs, is liable to snatch up your dinner. Perhaps due to the impact of language on thought (at least of the academic variety), this sort of explicit categorization has been assumed to underlie generalization. Indeed, much of the work on generalization treats categorization as an end-goal, rather than simply a means for informing actions (Anderson 1991; Ashby and Alfonso-Reese 1995; Kruschke 1992; Nosofsky 1986). Note a parallel to linguistics, where assigning a structure and denotational meaning to a speech act is assumed as the goal, rather than responding reasonably to the utterance (as pointed out by Clark 1997).

However, explicit categorization is only one possible mechanism for generalization. An alternative approach, coming from the parallel distributed processing group (Rumelhart, McClelland and Group 1986), is to learn higher order patterns in the environment directly, without mediating representations. This approach avoids challenges that arise with explicit categorization such as deciding when a group of items counts as a cateory and deciding which of several overlapping categories to generalize based on for a given property of a given item (Pothos and Wills 2011). A downside to this approach, however, is that the representations and computations underlying generalization are relatively opaque to the modeler, limiting the explanatory power of these models (Griffiths et al. 2010).

Fortunately, we do not need to choose between these two approaches. Graphs are capable of representing both types of models. Artificial neural networks are themselves graphs, many probabalistic models of categorization employ graphical representations (Tenenbaum et al. 2011), and previous graphical language models (Kolodny, Lotem and Edelman 2015; Solan et al. 2005) have represented syntactic categories as a special class of node with edges pointing to category members. However, in addition to these extreme ends of the spectrum, the HoloGraph can take an intermediary approach. We suggest such an approach here. Like the PDP appoaches, the proposed generalization algorithm does not employ explicit

categories or any other latent variables. Like category approaches, the basic units of the algorithm are individual items (and their feature vectors). This gives us a balance between the flexibility of the PDP approach and the inperpretability of the category approach.

We view generalization as a function from a raw representation of an item to a generalized representation of that item. The function may be applied, for example, before retrieving an edge weight or measuring the similarity between nodes. In line with both PDP and categorical approaches, we take as a starting point the assumption that if two items are similar in many ways, they might also be similar in other ways. In spatial terms, if two vectors point in similar directions, they can pull towards each other, making them even more similar. If the vector space is uniformly distributed, this will only result in noise. However, if there is structure to the space (i.e. areas with higher and lower density), this results in a fuzzy version of online clustering in which vectors drift towards heavily populated areas. Intuitively, we can think of vectors as having gravity: As one vector gets close to another, it will be pulled even closer. TODO: mass = sum of vector

Formally, for a node n_0, we create a generalized row vector as the sum of the all other nodes' row vectors weighted by the similarity of each node to n_0. The generalized row vector for edge e of n_0 is

$$\sum sim(n_0, row_{e,i}) row_{e,i}$$

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. TODO: NOTATION

A methodological problem arises with this algorithm. It requires performing an expensive similarity calculation for every node in the graph. The brain is a massively parallel computer, and can likely preform such operations fairly quickly. However, on a digital computer, the algorithm is prohibitively time-intensive when applied to anything but a toy grammar. Thus, we propose an alternative algorithm inspired by dynamic programming, which we call *dynamic association*. The basic principle of dynamic programming is to avoid computing the same thing multiple times, instead computing it just once and storing its solution. Every time we compute the generalized row vector for a given node, we will recalculate all similarities, most of which will have changed minimally. The the naive dynamic programming approach of simply caching each pair wise similarity will not do because this will ignore any information later acquired. Thus, we must go one step deeper into the algorithm to search for an efficient but well defined optimization.

Specifically, we observe that the similarity of two nodes is built up iteratively through a series of edge weight adjustments. By intervening at each of these steps, we can incrementally construct a generalized vector for each node. To do this, each node receives two additional vectors: a *dynamic row vector* and a *dynamic id vector*. The dynamic row vector approximates the generalized row vector as defined above. This vector is the sum of the dynamic id vectors of the nodes it points to. The dynamic id vector is the sum of the (non-dynamic) row vectors of every node that has pointed to it. The effect is that when an edge $a \rightarrow b$ is increased, a's dynamic row vector becomes more similar to every other node that points to b.

This is accomplished by performing two additional operations each time the edge weight from a to b is increased:

2. Add the dynamic id vector of b to the dynamic row vector vector of a.
3. Add the dynamic row vector of a to the dynamic id vector of b.

- TODO: other ideas
  - This separates generalization from the actual usage of the vector, which has some appeal from an interpretability standpoint, but may fail to capture settings in which generalization depends on the task at hand.
  - (just an idea) We compute similarity to all nodes, but we could compute similarity to only a select group of nodes (Edelman 1995). Perhaps the Chorus algorithm could be implemented with a HoloGraph. This would have the advantage of allowing one to compare stimuli that were encoded with different numbers of prototypes.

4.2.3 Compositionality

One form of generalization is unique and significant enough to merit separate discussion. The classical principle of compostionality, often attributed to Frege, states that the meaning of an expression is determined by the meanings of its constituent expressions and the rules for combining them. The principle is most often discussed in linguistics; however, language of thought theories (Fodor 1975; Piantadosi 2011) suggest that compositionality me be a fundamental characteristic of cognition more broadly (see also Werning, Hinzen and Machery 2012). Relaxing the assumption that meaning is determined *entirely* by compositionality, we see that the principle can apply to nearly any complex natural system.

It is important to distinguish between compositionality as a characteristic of a system and compositionality as a characteristic of the *interpretation* of a system. Classic counter-examples to Frege's principle, such as idiomatic expressions (e.g. 'kick the bucket'), demonstrate that compositionality does not always define the true meaning of an expression. Rather, following the first principle of the BindGraph, the meaning of an expression is ultimately determined by its relationships with other expressions and concepts. Thus, rather than definer of meaning, we can view compositionality (in the cognitive sense) as a *cue* to meaning. Compositionality is a cognitive tool for predicting the meaning of a novel item based on previous experience with related items. That is, compositionality is a form of generalization.

In particular, compositionality is a second order generalization: Whereas a first order generalization is based on commonalities between elements, compositionality is based on commonalities in the *relationships between elements*. For example, the ternary relationship ('hire', 'John', 'hire John') is quite similar to the relationship ('kill', 'Jack', 'kill Jack'), but less similar to ('with', 'Jason', 'with Jason').

- the case of composition of words where we only care about syntax is an especially easy case.

Still need to introduce our algorithm. Brain too tired right now.

4.3 Simulations

4.3.1 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 true probabilities as edges, a 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 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. 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, because there is a chance that two randomly selected index vectors will not be orthogonal, the HoloGraph weights will be subject to some noise. We expect that the effect of noise will be greater for lower dimensionality vectors, and for more total nodes. As shown in figure Figure 1, the results match our expectations.

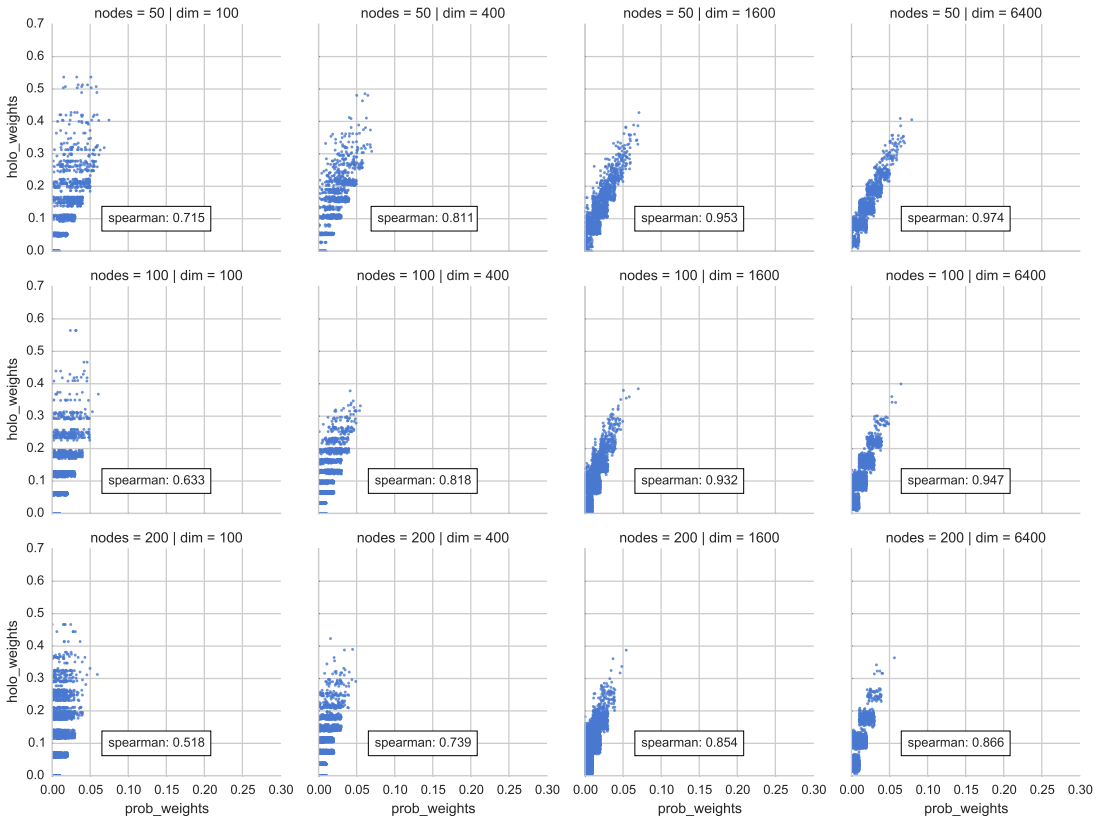

Figure 1: Correlation of sparse vector and probabilistic graph edge weights with varying number of nodes and vector dimensionality. Node count increases from top to bottom. Dimensionality increases from left to right.

4.3.2 Generalization

To test the generalization algorithm, we create a bigram model with a HoloGraph. The graph is trained on two corpora generated with probabilistic context free grammars. The grammars are nearly identical except for one determiner and one noun. The first has 'that' and 'table', while the second has 'my' and 'bunny'. As a result, the strings 'that bunny' and 'my table' never occur in the combined corpus. However, the two determiners and the two nouns will have otherwise similar edge profiles. If the generalization algorithm is successful, it will recognize this similarity and assign a non-zero weight the edge representing transitional probability between the unseen pairs. As shown in figure Figure 2, TODO: both generalization algorithms are successful.

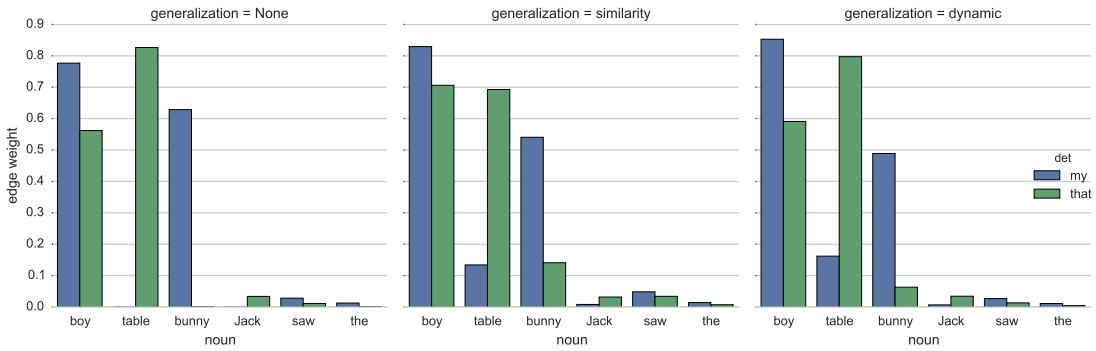

Figure 2: Generalization. A non-zero edge weight is assigned to edges that were never bumped.

4.3.3 Experiment 3: Compositionality

To test the composition algorithm, we begin with the same bigram model as used in the previous simulation. We then create nodes representing noun phrases with all determiner-noun pairs, excluding the `the` and the noun `boy`. For each of these phrases, we assign high edge weights to the `saw` and `ate`. As a test item, we create a new noun phrase node, `[the boy]`, either using the composition algorithm or not. Critically, `[the boy]` receives no direct training. We then measure edge weights from `[the boy]` to `saw` and `ate`. These will be near-zero when no composition is used. However, if the composition algorithm is successful, we expect `[the boy]` to have high edge weights to `saw` and `ate`. This is because previously encountered nodes composed of pairs of nodes like (`the`, `boy`) have high weights to `saw` and `ate`. As showin in figure Figure 3, the results match our expectations.

5 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 (Kolodny, Lotem and Edelman 2015; Solan et al. 2005) and chunking models

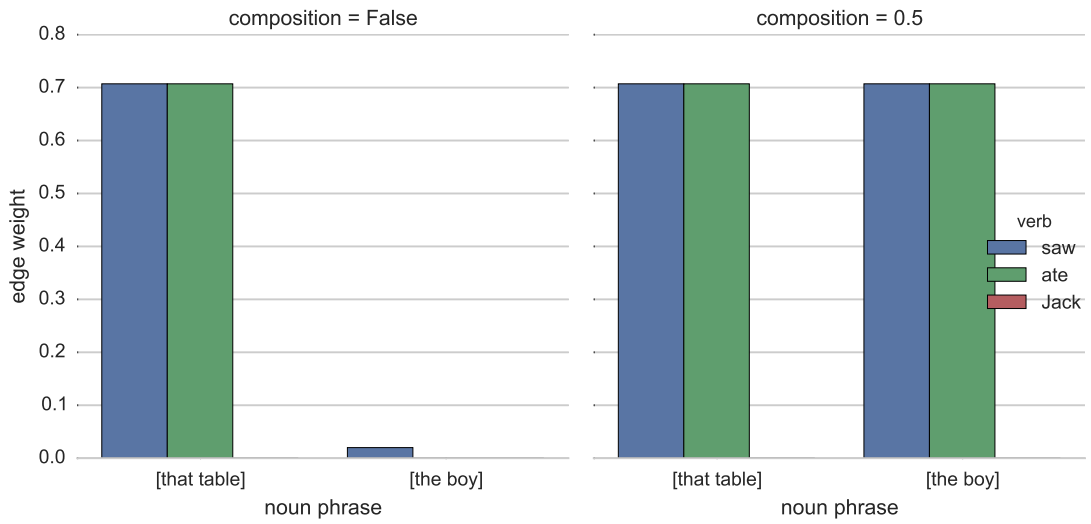

Figure 3: Composition. A newly created node has edge-weights similar to existing nodes composed of similar elements.

(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 (Goldstein et al. 2010; Tomasello 2003), 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 (**bannard09a**). 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.

5.1 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.

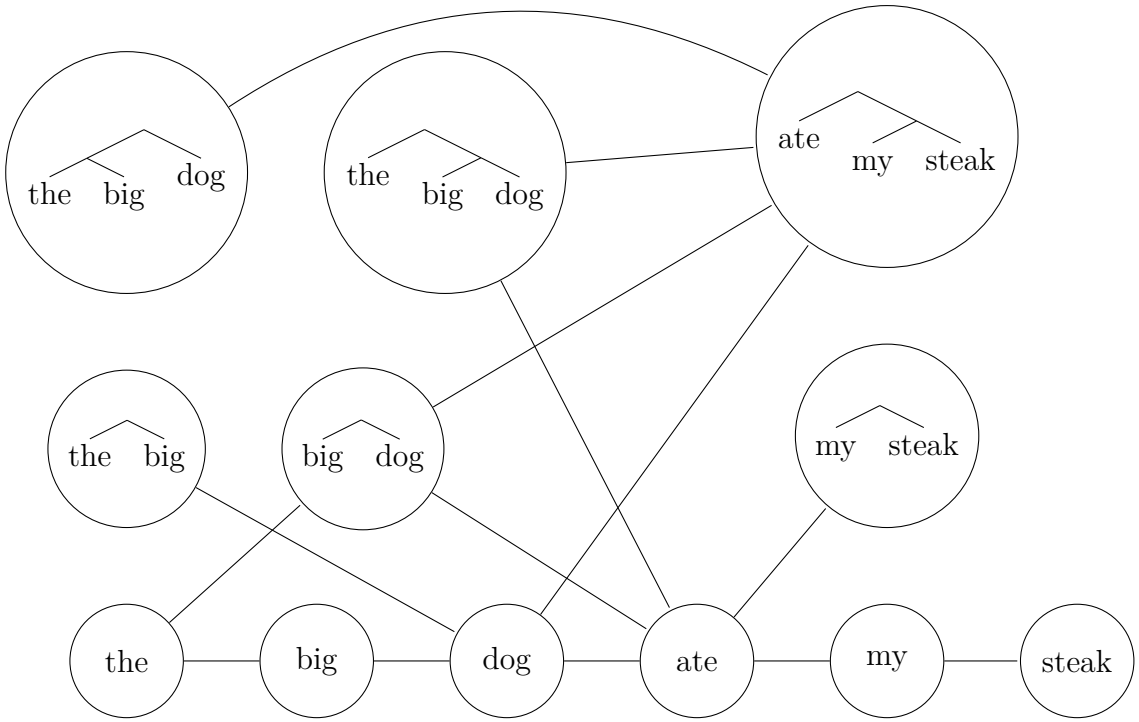

Figure 4: A simplified graph representing the utterance "the big dog ate my steak". For display, FTP and BTP edges are represented as a single undirected edge.

5.1.1 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.

5.1.2 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,) 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]

5.2 Learning

The model constructs a BindGraph 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.

5.2.1 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?]

5.2.2 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
```

6 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 of the seven languages TODO: we don't use hungarian because unicode, the input can be tokenized by word, syllable, or phoneme, giving a total of $7 \times 3 = 21$ 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.

6.1 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). The only requirement is that the model be able to quantify how well an utterance fits its knowledge of the language.

6.1.1 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 $NT \to \alpha$, 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 $\alpha \to \alpha \cdot w$, 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 $X \to X \cdot Y$ 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?]

6.1.2 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.

An ROC curve, such as the one in figure Figure 5, plots true positive rate against false positive rate. As the acceptability threshold is lowered, both values 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 a 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).

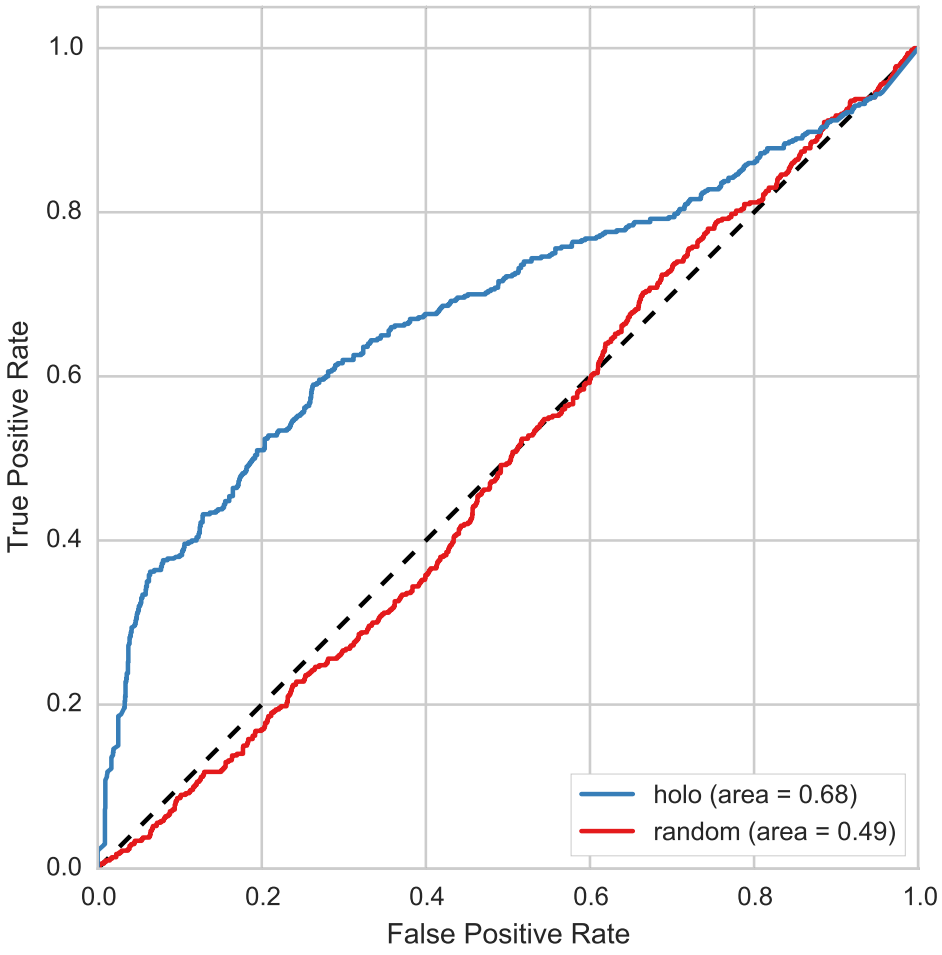

Figure 5: A Receiver Operator Characteristic curve for two models on the English word corpus.

6.1.3 Results

Overall, Nümila preforms better than chance, but no better than a bigram model. The various instantiations of Nümila all preform roughly the same, although the noise introduced by the HoloGraph has a slight negative impact. There does not appear to be any interaction with language or input type with the notable exception that the non-hierarchical model using a greedy parsing algorithm performs poorly with the phonetic input. TODO: We have no explanation for this anomaly.

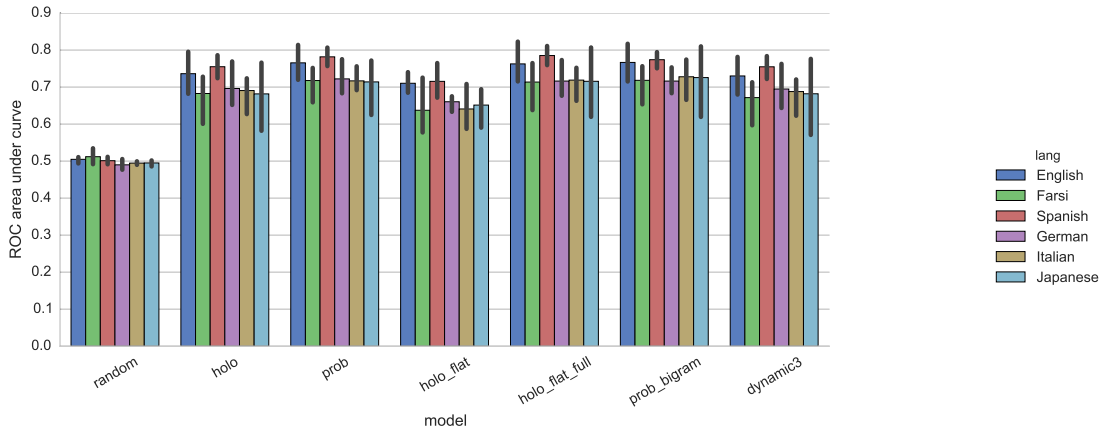

Figure 6: Area under ROC curve for different languages, collapsed across input type.

6.2 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; the syntactic task is to say the words in the right order.

6.2.1 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

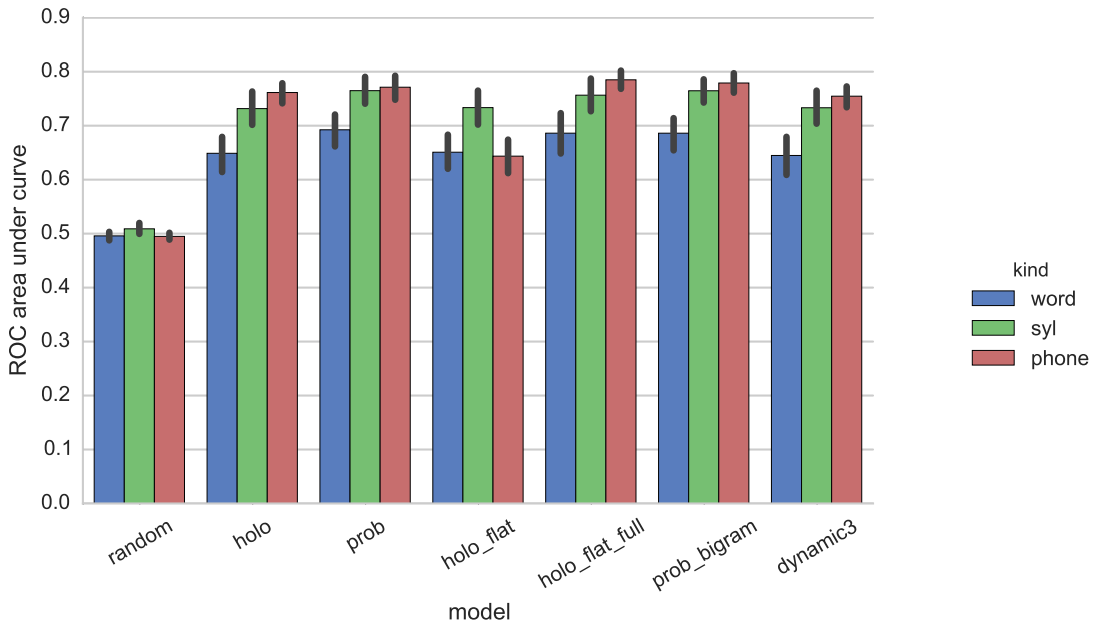

Figure 7: Area under ROC curve for different input types, collapsed across languages.

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. Kolodny, Lotem and Edelman 2015; McCauley and Christiansen 2014). Incremental production has some theoretical appeal given that words are ultimately produced in this order. However, it is not clear that utterances are planned in this way. 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
```

6.2.2 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.

6.2.3 Results

The production results are slightly more informative. Nümila outperforms the bigram model. However, the speaking algorithm was developed with chunks in mind, so this might be an unfair comparison.

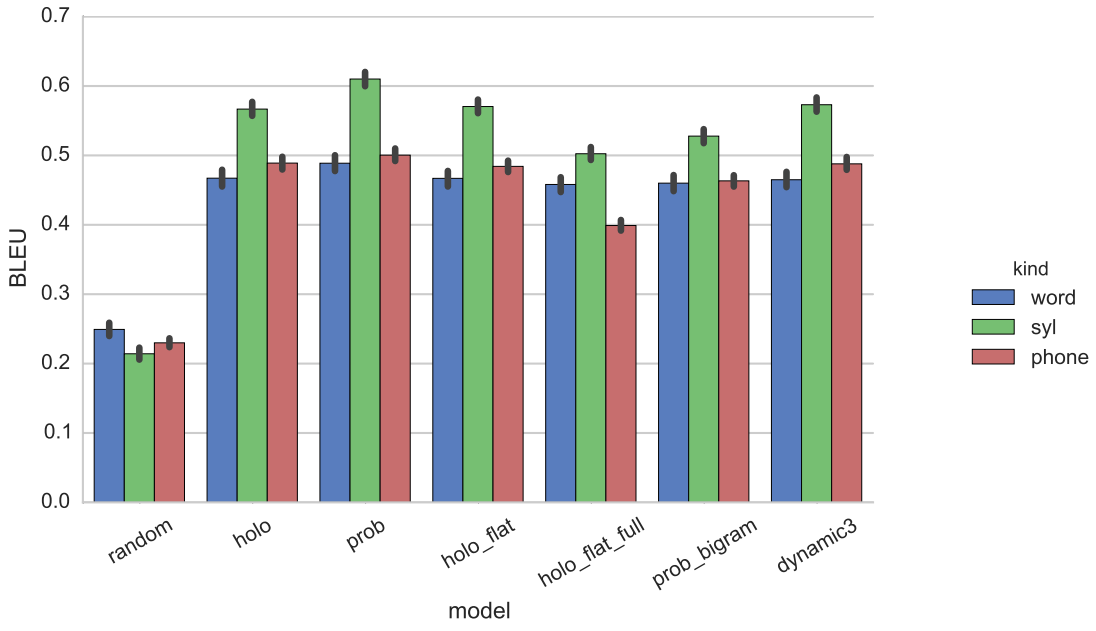

Figure 8: Production results.

7 Conclusion

Grahps are nice. Language is mean. This was fun.

References

Anderson, John R. 1991. 'The adaptive nature of human categorization.' *Psychological Review* 98 (3): 409.

Ashby, F Gregory, and Leola A Alfonso-Reese. 1995. 'Categorization as probability density estimation'. *Journal of mathematical psychology* 39 (2): 216–233.

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.

Chang, Franklin, Elena Lieven and Michael Tomasello. 2008. 'Automatic evaluation of syntactic learners in typologically-different languages'. *Cognitive Systems Research* 9 (3): 198–213.

Chomsky, Noam. 1995. *The minimalist program.* Vol. 1765. Cambridge Univ Press.

———. 2004. *The Generative Enterprise Revisited: Discussions with Riny Huybregts, Henk van Riemsdijk, Naoki Fukui and Mihoko Zushi.* Walter de Gruyter.

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

Clark, Herbert H. 1997. 'Dogmas of understanding'. *Discourse Processes* 23 (3): 567–598.

Davis, Jesse, and Mark Goadrich. 2006. 'The relationship between Precision-Recall and ROC curves'. In *Proceedings of the 23rd international conference on Machine learning,* 233–240. ACM.

Edelman, Shimon. 1995. 'Representation, similarity, and the chorus of prototypes'. *Minds and Machines* 5 (1): 45–68.

———. 2008. 'On the nature of minds, or: truth and consequences'. *Journal of Experimental*
Theoretical Artificial Intelligence 20 (3): 181–196.

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): 729–743.

Firth, John R. 1957. 'A synopsis of linguistic theory, 1930-1955'.

Fodor, Jerry A. 1975. *The language of thought.* Vol. 5. Harvard University Press.

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): 249–258.

Goodman, Joshua T. 2001. 'A bit of progress in language modeling'. *Computer Speech*
Language 15 (4): 403–434.

Griffiths, Thomas L, Nick Chater, Charles Kemp, Amy Perfors and Joshua B Tenenbaum. 2010. 'Probabilistic models of cognition: exploring representations and inductive biases'. *Trends in cognitive sciences* 14 (8): 357–364.

Griffiths, Thomas L, Mark Steyvers and Joshua B Tenenbaum. 2007. 'Topics in semantic representation.' *Psychological review* 114 (2): 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): 514–530.

Jones, Michael N, and Douglas JK Mewhort. 2007. 'Representing word meaning and order information in a composite holographic lexicon.' *Psychological review* 114 (1): 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): 400–401.

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): 227–267.

Kruschke, John K. 1992. 'ALCOVE: an exemplar-based connectionist model of category learning.' *Psychological review* 99 (1): 22.

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): 211–240.

Lund, Kevin, and Curt Burgess. 1996. 'Producing high-dimensional semantic spaces from lexical co-occurrence'. *Behavior Research Methods, Instruments, & Computers* 28 (2): 203–208.

Marr, David. 1982. *Vision: A computational investigation into the human representation and processing of visual information.*

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): 419–436.

Mitchell, Jeff, and Mirella Lapata. 2010. 'Composition in distributional models of semantics'. *Cognitive science* 34 (8): 1388–1429.

Nosofsky, Robert M. 1986. 'Attention, similarity, and the identification–categorization relationship.' *Journal of experimental psychology: General* 115 (1): 39.

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,* 311–318. 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): 244–247.

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,* 2775–2780. Citeseer.

Piantadosi, Steven Thomas. 2011. 'Learning and the language of thought'. PhD diss., Massachusetts Institute of Technology.

Plate, Tony, et al. 1995. 'Holographic reduced representations'. *Neural networks, IEEE transactions on* 6 (3): 623–641.

Pothos, Emmanuel M, and Andy J Wills. 2011. *Formal approaches in categorization.* Cambridge University Press.

Rumelhart, David E, James L McClelland, PDP Research Group et al. 1986. 'Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1-2'. *Cambridge, MA.*

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.

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): 11629–11634.

Steedman, Mark. 2000. *The syntactic process.* Vol. 24. MIT Press.

Tenenbaum, Joshua B, Charles Kemp, Thomas L Griffiths and Noah D Goodman. 2011. 'How to grow a mind: Statistics, structure, and abstraction'. *science* 331 (6022): 1279–1285.

Tomasello, Michael. 2003. 'On the different origins of symbols and grammar'. *Evolution of Language* 3:94–110.

Werning, Markus, Wolfram Hinzen and Edouard Machery. 2012. *The Oxford handbook of compositionality.* OUP Oxford.