Technical Appendix

Michael C. Frank, Noah D. Goodman, and Joshua B. Tenenbaum
Department of Brain and Cognitive Science
Massachusetts Institute of Technology
{mcfrank, ndg, jbt}@mit.edu

February 28, 2009

1 Introduction

This supplementary document accompanies the model of cross-situational word learning described in our paper "Using speakers' referential intentions to model early cross-situational word learning."[1] It is also meant to be read in conjunction with the source code for the models we describe in the paper. Together we hope these materials give the interested reader all of the information necessary to evaluate, replicate, or extend our work.

We focus here on the technical details of our implementation of the model, including inference, parameter setting, and experimental simulations. In section 2 we discuss the nuts and bolts of the accompanying code (which we refer to throughout). In section 3 we give the details of how scoring is accomplished in the model, including our caching scheme for speeding up inference, and then some details on the inference itself. In section 4, we describe the comparison models we used. Finally, in sections 5 and 6 we give the details of our corpus experiments and experimental simulations, respectively.

2 Guide to code base

The code provided here should be sufficient to replicate all of the simulations described in our paper. The current version of our model is provided in the form of Matlab scripts, rather than as application code. As such, it most likely cannot be applied to new datasets without modification of the source code. This decision reflects a belief on our part that our model is useful primarily as an exploratory tool and thus that readers interested in replicating our work and applying our model to new situations should become familiar with some basics of how it is implemented.

This situation also provides a useful opportunity for readers who are unfamiliar with the machinery of Bayesian models to open up the hood and get a feel for how a model operates. In the process, we suspect that distinctions which are often blurred in discussion of these models will become clearer. One such distinction is the one between the generative model, which in our case is relatively trivial to compute using a series of nested `for` loops, and inference in the model, which is highly complex and for which our current scheme is only a first possibility. Another is between prior and likelihood. Often, assumptions attributed to Bayesian models are spoken about in terms of priors. In the case of our current model, this shorthand is misleading; the prior we use is extremely simple (it is computed in one line of code that barely merits its own subfunction). Instead, the assumptions of the model are encoded in the hierarchical likelihood terms, reflecting the generative assumptions the model makes about the data.

In this section we describe the basic data structures used in our model and the organization of the code base. Because of the relative simplicity of the model, the total code base runs to only slightly more than

[1] Throughout this document we refer to the model informally as the "wurwur model" denoting the "word-world" mappings that it learns.

1000 lines, and the basic model is considerably shorter. Thus, we provide this guide to encourage readers to become familiar with its workings.

2.1 Platform

All code for our models was written in Matlab R2007b for the Macintosh, using Mac OS 10.5. There should be no dependencies to external Matlab toolboxes.

2.2 Directory structure

Code for our model is stored in four directories as well as the root model directory:

- root directory (`.`) - contains main model scripts.
- `helper_functions` - miscellaneous functions, including those for evaluation of models.
- `baseline_functions` - necessary code for baseline models.
- `wurwur_functions` - necessary code for the Bayesian model.
- `experimental_models` - code corresponding to simulations of experimental results.
- `data` - contains .MAT files for data, including the corpus, the gold-standard, and the best lexicons found for each model (useful for later experimental simulations).

2.3 Models

The four main model scripts are `wurwurModel.m` (our model), `yuModel.m` (the IBM Machine Translation Model I, used by Yu & Ballard, 2007), `yuModelReversed.m` (a reversed version of IBM Translation Model I), and `associativeModels.m` (which computes associative, conditional probability, and mutual information baseline models). Each of these scripts should simply run the appropriate model on the Rollins Corpus portion that we describe in the paper; at the end of a run (which may take anywhere from a few seconds to 30 minutes, depending on the model), a figure window will appear with the lexicon that the model learned and the precision and recall of that lexicon relative to the gold standard.

2.4 Data structures: Corpus, Lexicon, Gold-standard, and World

The core data structures involved in this project are the `corpus` from which we learn words, the `lexicon` that we learn, the `gold_standard` lexicon, and the `world` in which the learning takes place. For ease of computation in Matlab, all words and objects are transformed to numbers; the correspondences between numbers and text labels for words and objects are kept in the `world` struct, in `world.words_key` and `world.objects_key`, respectively.

2.4.1 The lexicon

The lexicon created by all of our models can be thought of as a graph with edges connecting words with objects. This is represented compactly in the variable `lex.map` by a 2xN array of words and objects (with each column indicating a link between a word and an object).

2.4.2 The corpus

The corpus (`corpus.mat`) is a Matlab struct array in which each element of the array consists of a situation—a pairing of an utterance with a set of objects that are present and visible to the learner during that utterance. For instance, the first situation of the corpus is:

```
words: [6 205 391 66 293 50 84]
objects: [4 3 20 10 18]
```

which can be translated:

```
words: ahhah look we can read books david
objects: BOOK BIRD RATTLE FACE
```

The corpus we use in all of our simulations is adapted from two files from the Rollins section of the CHILDES corpus (MacWhinney, 2000). To adapt the corpus, we first removed all punctuation and formatting. Next, for each sentence we coded the set of objects that we judged to be visible to the child during the utterance of that sentence. We included only mid-level objects, excluding referents such as the floor, the rug, the toy box, the car-seat that the child sat in, and the mother and the child's clothes (none of these referents was mentioned consistently by either mother).

2.4.3 The intents

The intents (`coded_intents.mat`) are a human coder (MCF)'s best guess as to which of the objects present in a particular situation the current utterance refers to. For simplicity and to create an objective criterion, only utterances containing nouns or pronouns were judged to refer directly to an object. Thus, e.g., "can you see it" was given an intent of PIG, but "oink oink" was given an empty intent. This set of judgments forms our gold-standard for evaluating intentional inferences by the various models.

2.4.4 The gold standard lexicon

The gold-standard lexicon (`gold_standard.mat`) is a hand-coded set of all the appropriate word-object mappings that occurred at least once in the corpus (Table 1). We included baby-talk ("moocow") as well as plurals ("moocows"); our intent was to capture all the words that were being used referentially within the corpus.

2.4.5 The world

The world (`world.mat`) is a struct which holds useful information about words, objects, their text translations, quantities, and frequencies. It also holds the same mutual information matrix described below in the section on associative baseline models. This matrix is important for inference because we use it as a proposal distribution for adding new words to the lexicon.

```
words: [1x419 double]
words_key: {1x419 cell}
num_words: 419
objects: [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23]
objects_key: {1x23 cell}
num_objects: 23
words_freq: [1x419 double]
objects_freq: [5 18 14 134 113 15 21 67 75 19 5 15 102 43 25 11 15 28 140 199
       175 12 15]
mis: [419x23 double]
```

word	**object**	**word**	**object**
baby	BABY	bear	BEAR
bigbird	BIRD	bigbirds	BIRD
bird	BIRD	book	BOOK
books	BOOK	bunny	BUNNY
bunnyrabbit	BUNNY	cow	COW
cows	COW	moocow	COW
moocows	COW	duck	DUCK
duckie	DUCK	eyes	EYES
hand	HAND	hat	HAT
kitty	KITTY	kittycat	KITTY
kittycats	KITTY	lamb	LAMB
lambie	LAMB	mirror	MIRROR
pig	PIG	piggie	PIG
piggies	PIG	rattle	RATTLE
ring	RING	rings	RING
sheep	SHEEP	bunnies	BUNNY
birdie	DUCK	bird	DUCK

Table 1: The gold-standard lexicon of word-object mappings

2.5 Other code

The folder `helper_functions` contains a number of functions that are crucial to the operation of all the models:

`computeLexiconF.m` - Computes precision, recall, and F-score for a lexicon relative to the gold standard.

`consolidateMatrix.m` - Thresholds a matrix of word-object associations and returns the highest scoring lexicon for that matrix.

`evalLexicon.m` - Evaluates and plots a lexicon graphically.

`multirnd.m` - Draws randomly from a multinomial distribution. This code by David Ross, included to avoid external dependencies.

`randint.m` - Draws a random integer, included to avoid dependencies.

3 Details of the Bayesian word learning model

The basic structure of our model is given in both the journal paper that these materials accompany and in our NIPS paper (Frank et al., 2008). Here we focus primarily on our implementation of the model. Because many terms in the model are independent of the lexicon, we can cache them at the beginning of the computation, rather than re-computing them each time the lexicon changes. This caching makes the actual inference procedure in the model less transparent than it might otherwise be, though it speeds computation dramatically.

We first describe the basic structure of how we determine the posterior score—the un-normalized product of prior and likelihood, described in Equation (1) of the paper—for a lexicon. Next, we describe our full inference scheme: how we modify lexicons during search to find the maximum a posteriori (MAP) lexicon. Finally, we index the functions in the code-base.

3.1 Scoring

The basic function responsible for scoring is `scoreLexicon.m`, which takes a lexicon and a corpus and produces a posterior score. While the prior is simple to compute, the likelihood is more challenging. Basically, for each situation in the corpus we must compute Equation (3) of the paper:

$$P(W_s|L,O_s) = \sum_{I_s} P(W_s|I_s,L)\cdot P(I_s|O_s). \tag{1}$$

We further simplify by setting $P(I_s|O_s) \propto 1$, and substitute in Equation (4) from the paper:

$$P(W_s|L,O_s) = \sum_{I_s}\prod_{w\in W_s}\left[\gamma\sum_{o\in I_s}\frac{1}{|I_s|}P_{\mathrm{R}}(w|o,L)+(1-\gamma)P_{\mathrm{NR}}(w|L)\right]. \tag{2}$$

We compute this for each sentence via accessing a series of cached data-structures. Here we walk through an example computation for the first situation in the corpus, using the gold standard lexicon. Recall that for the first situation in the corpus, we have:

```
words: ahhah look we can read books david
objects: BOOK BIRD RATTLE FACE
```

Now we begin by computing the possible intents for this sentence—the power set over the objects that are present. This will not change, so we can compute it offline and cache it in the corpus. Since the value for γ will also not change over the course of the simulation, we can multiply by the gamma factor in the first term of Equation 2. We use $\gamma = .1$. We then divide this γ value by the total size of the intention, yielding a matrix whose rows are intentions and whose columns correspond to objects (with the first column corresponding to the hypothesis of no object present):

```
corpus(1).gamma_intents =

   1.0000         0         0         0         0
   0.9000         0         0         0    0.1000
   0.9000         0         0    0.1000         0
   0.9000         0         0    0.0500    0.0500
   0.9000         0    0.1000         0         0
   0.9000         0    0.0500         0    0.0500
   0.9000         0    0.0500    0.0500         0
   0.9000         0    0.0333    0.0333    0.0333
   0.9000    0.1000         0         0         0
   0.9000    0.0500         0         0    0.0500
   0.9000    0.0500         0    0.0500         0
   0.9000    0.0333         0    0.0333    0.0333
   0.9000    0.0500    0.0500         0         0
   0.9000    0.0333    0.0333         0    0.0333
   0.9000    0.0333    0.0333    0.0333         0
   0.9000    0.0250    0.0250    0.0250    0.0250
```

For example, in the fourth row, we see .9 probability assigned to no object, with the .1 probability that is assigned to γ being split between the third and fourth objects (RATTLE and FACE). Since all of this computation need only happen once—it depends only on γ and O, which do not change during inference over the lexicon—we perform it at initialization in the function `addCorpusIntentCache.m`.

For each lexicon, we next cache the probability that words are picked referentially or non-referentially for a particular object (we call this `lex.word_costs`). We compute this over all objects and words in the function `addWordScoresCache.m` which can be called once for each lexicon. We then look up the word costs from this matrix for the particular words and objects in each situation. For the first situation, the word cost (objects x words) matrix is:

```
  0.0026    0.0026    0.0026    0.0026    0.0026    0.0001    0.0026
       0         0         0         0         0    0.5000         0
       0         0         0         0         0         0         0
       0         0         0         0         0         0         0
  0.0026    0.0026    0.0026    0.0026    0.0026    0.0001    0.0026
```

The probability of the word "book" being used to refer to the object BOOK is .5, because there are only two lexical entries under BOOK ("book" and "books"). On the other hand, the probability of the word "aah" being used referentially to refer to BOOK is 0 since there is no lexical entry linking them. The first row indicates the probability of non-referential use of each word, .0026 for each non-lexical word and .0001 for "book" since it is in the lexicon (and κ reduces its probability of being used non-referentially). The last row indicates probabilities for the object FACE which has no word for it even in the gold-standard. Thus, the probabilities for it are also those for non-referential use.

We next multiply the `gamma_intents` matrix (intents x objects) with the `word_cost` matrix (objects x words) to produce an intents x words matrix that encodes the total probability enclosed in the sum over intents and product over words in Equation 2. The only steps left are to take the product over words, then to take the the sum over intents (normalized by the uniform probability of any given intent, $\frac{1}{|I_s|}$). This calculation gives us the value of Equation 2 for this sentence; we iterate the calculation over all sentences (an efficient operation since the values for `word_cost` and `gamma_intents` are cached already) and multiply by the prior probability of the lexicon to get the full posterior score of a lexicon.

3.2 Inference details

To search for the MAP lexicon, we use stochastic search techniques. Unfortunately, finding the MAP lexicon is very difficult because the posterior distribution over lexicons is highly irregular. There are many cases in which adding a correct word-object pairing may not add to the posterior score of the lexicon because there are already several other incorrect words in the lexicon that are partially associated with that object, incorrectly "explaining away" the cases in which the word and object appear together. Thus, simple changes to the lexicon may not produce improvements in the posterior score even if they seem to be moving the current lexicon closer to the MAP.

To prevent the model from getting stuck in the pervasive local maxima of the search space we use a number of techniques. First, our search is stochastic; we use a simple rule to decide whether to take a proposed lexicon L_{new} over the current lexicon L_{old} (inspired by the standard Metropolis-Hastings rule):

$$P_{accept} = \begin{cases} \frac{P(L_{new})}{P(L_{old})} & \text{if } P(L_{new}) < P(L_{old}) \\ 1 & \text{if } P(L_{new}) > P(L_{old}) \end{cases} \tag{3}$$

This stochastic search rule is a first step towards keeping the search from becoming irremediably stuck in local maxima.

In order to produce proposals for L_{new}, we rely on three search moves (contained in the function `mutate.m`): adding a word-object pairing to the lexicon, deleting a pairing from the lexicon, and swapping one pairing for another. Rather than adding pairings at random, we instead use a strategy resembling data-driven Markov Chain Monte-Carlo (Tu & Zhu, 2002): we sample proposals for added words from the distribution created by normalizing the mutual information of words and objects in the corpus. This distribution assigns non-zero probability to all links which appear in the corpus, but it effectively biases the search towards promising pairings.

The posterior distribution over possible lexicons is very irregular, however. Therefore, in order for our search to move effectively around this space, we use simulated tempering (Marinari & Parisi, 1992). In practice, we establish five different parallel searches in which posterior scores are exponentiated to different values of $1/T$ where $T = [0.0001, 1, 10, 100, 1000]$. In conjunction with our Metropolis-Hastings rule, this establishes searches which range from very greedy (at the lowest temperature) to very permissive of lower-probabilty alternatives (at the highest temperatures). While the greedy search will only hill-climb to better alternatives in the nearby vicinity of the current lexicon, the higher-temperature searches will range more widely over the full posterior space. We then intermittently (every five steps) use the `breed.m` function to consider various ways of combining low and high-temperature searches. The breeding moves we consider are: fully exchanging lexicons between searches, adding or swapping word-object pairings, and swapping only the word or only the object in a pairing from one search lexicon to another. The overall result of this scheme is that higher-temperature searches are free to wander away from local maxima. If they stumble across innovations which result in a higher posterior score, those innovations will likely be adopted by greedier searches, which will then explore other hypotheses in the vicinity of this higher-scoring alternative. In practice, these tactics are effective enough that the greedy search in the model generally converges to its final value within 10k - 50k search moves.

3.3 Code

The code for the Bayesian model resides in the subdirectiory `wurwur_functions`.

`addCorpusIntentCache.m` - Run once to initialize caching. Adds caching of intents relative to the γ parameter.

`addWordScoresCache.m` - Run every time the corpus is scored. Caches the probabilities of words being used referentially and non-referentially (which must be recomputed since they depend on the lexicon).

`mutate.m` - Makes a proposal to change the lexicon by one move. Part of search.

`breed.m` - Makes a proposal to breed lexicons from two searches of different temperatures. Part of search.

`sampleLexicon.m` - Initialization function to seed searches.

`scoreLexicon.m` - The heart of the model, gives a posterior score (non-normalized) for a lexicon given the corpus.

4 Details of the comparison models

In this section we describe details of the implementation of each of the comparison models we used in our work. These models all have a basic feature in common: each uses some method to produce an MxN word-object matrix of association weights, which must be transformed into a lexicon for evaluation. We used the same method in each case: we renormalized the matrix to be on the unit interval, then created lexicons according to a threshold by including each word-object pairing in the lexicon for which the associative weight was greater than the threshold value. We performed this operation for thresholds on the interval `[.01:.01:.99]` and scored each one according to the gold-standard lexicon. The results for each model are those of the best lexicon found for any threshold value.

4.1 Associative baseline models

The four associative baseline models we used were all based on simple statistics which we computed over the corpus for each word-object pairing.

4.1.1 Association frequency

To compute $P(w, o)$, the association frequency of a word w and object o, we simply normalized the count of all times an object appeared in the same situation as a word, $C(w, o)$:

$$P(w, o) = \frac{C(w, o)}{\sum_i \sum_j C(w_i, o_j)} \tag{4}$$

.

4.1.2 Conditional probability

We computed conditional probability of both a word given an object, $P(w|o)$, and an object given a word, $P(o|w)$, normalizing by either $C(o)$, the count of an object, or $C(w)$, the count of a word:

$$P(w|o) = \frac{C(w, o)}{C(o)} \tag{5}$$

and

$$P(o|w) = \frac{C(w, o)}{C(w)} \tag{6}$$

.

4.1.3 Mutual information

We additionally included point-wise mutual information, a bi-directional statistic which we perceived might have some of the advantages of both directions of conditional probability (Swingley, 2005):

$$MI(w|o) = \frac{C(w, o)}{C(w)C(o)} \tag{7}$$

.

4.1.4 Code

Since the baseline models (with the exception of the Translation model) are relatively similar and relatively trivial to compute, we include code for all of them in the script `associativeModels.m`. The only helper function for this script is `consolidateMatrix.m` which computes precision, recall, and F-score across the full range of thresholds for an arbitrary matrix of summary statistics.

4.2 Translation Model

4.2.1 Model

The translation model described by Yu & Ballard (2007) is an adaptation of IBM Machine Translation Model I (Brown et al., 1994). Yu and Ballard describe using the model to optimize $P(o|w)$; we additionally used the same model to compute $P(w|o)$, for which we found differing (superior) results. Both directions are conceptually reasonable within the original translation framework—one corresponds to $P(french|english)$ and the second corresponds to $P(english|french)$. We do not give the full structure of this model, since it is described in detail elsewhere (Yu & Ballard, 2007; Brown et al., 1994).

However, two design decisions are worth discussing. First is the addition of the `NULL` object to the corpus. The `NULL` object allows the model to decide that a word is not mapped to anything in the scene. The mechanism for this decision is a positive one: the model decides that a word is most consistently associated with `NULL`—thus, even though the association of a particular word with `NULL` may not be strong, it can still be stronger than its association with other objects. We tested the model in both directions,

	Yu & Ballard		Our corpus	
	di06	me03	di06	me03
Utterances	281	321	303	316
Average words per situation	>4	>3	4.15	3.96
Average referents per situation	>2	>2	2.21	1.87
Word types	336	225	330	217
Word tokens	1236	1072	1257	1250
Object types	18	21	21	18
Object tokens	637	898	671	590

Table 2: Descriptive statistics for Yu & Ballard's and our coding of the Rollins corpus.

$P(o|w)$ and $P(w|o)$, with and without the `NULL` object and found that in both cases, the addition of `NULL` increased performance.

Second, Yu & Ballard (2007) initialize their version of the model with the association frequencies of words and objects; for convenience we use uniform initial assignments. This decision does not affect outcomes in the model since Brown et al. (1994) prove that IBM Model I (though not their later models) converges to the same solution regardless of initialization.

4.2.2 Differences in performance from Yu & Ballard (2007)

In Yu & Ballard (2007), the results reported for the translation model in Figure 8 are .75 precision and .58 recall, for an F-score of .65. In contrast, in the direction $P(o|w)$, our implementation of the translation model—run on our version of the Rollins corpus—achieves far lower performance, with precision of only .09 but a recall of .26 and an F-score of .13 (indicating a very large lexicon with many false positives). In the opposite direction, $P(w|o)$, precision was .15, recall was .38, and F was .22. What could account for these striking differences in reported results? Logically, there are only four possibilities: differences in the model, differences in the inference scheme, differences in the corpus, and differences in the gold standard.

We believe that differences in the model and inference algorithm (EM) are not to blame for the differences in performance we observed. The model reported in Yu & Ballard (2007) is essentially the same as Model I in Brown et al. (1994). We have tested our implementation of Model I against several other reference implementations of Model I and found no differences in performance. We are thus left believing that differences in the corpus and gold standard must be responsible for these differences. Table 2 (adapted from Tables 2 and 3 in Yu & Ballard, 2007) gives summary statistics for the corpus used by Yu & Ballard (2007) and for our version of the corpus. Although there are some differences (e.g., me03 has more words and fewer objects in our coding), they do not seem to be meaningfully different. The final possibility is that our gold standard differed from that used by Yu & Ballard (2007); however, we consider it unlikely that a slightly larger or smaller gold standard would lead to large changes in performance. Thus, we have no conclusive explanation for the differences we observed, though it is possible that they stem from subtler differences in the corpus than can be captured by descriptive statistics about the number of objects present in a situation.

4.2.3 Code

The code associated with the model includes four functions: wrapper functions which run the model in each direction and functions implementing the expectation and maximization steps of inference within the model.

`yuModel.m` - computes $P(o|w)$.

`yuModelReversed.m` - computes $P(w|o)$.

`computeCounts.m` - Expectation step. Computes expected counts of words and objects in the corpus based on the associative matrix created in the maximization step.

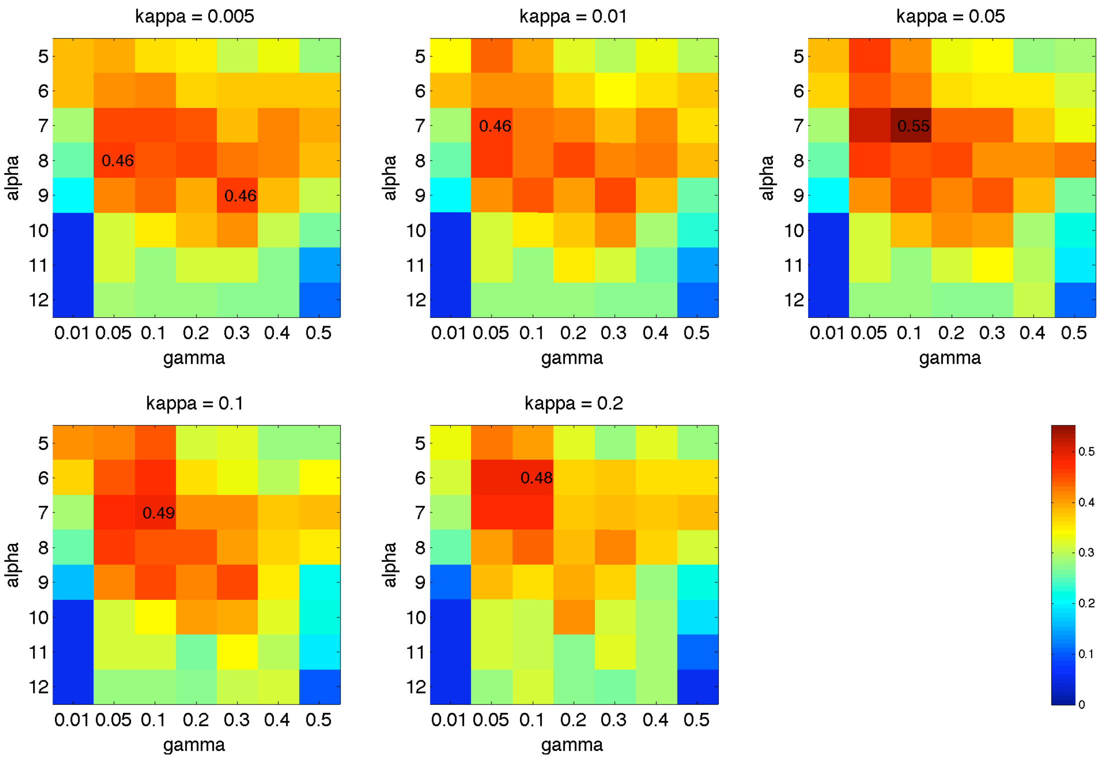

Figure 1: Results of a grid search of the parameter space of the model. Subplots are given for different values of κ with the x axis representing γ values and the y axis representing α values. Hotter colors indicate a greater F-score, with the best F-score for each subplot inscribed in the corresponding cell.

`computeAssociations.m` - Maximization step. Computes the associations between words and objects based on the corpus counts created in the expectation step.

5 Corpus experiments

5.1 Lexicon search

Our corpus results are the result of a grid parameter search over α = [5, 6, 7, 8, 9, 10, 11, 12], γ = [.01, .05, .1, .2, .3, .4, .5] and κ = [.005, .01, .05, .1, .2]. The extreme values in these ranges were determined via pilot simulations varying these parameters independently. The model was randomly initialized for each simulation. Five model runs of 20,000 search steps each were conducted for each parameter setting, resulting in 1,400 separate simulations of roughly 30 minutes each. Figure 1 shows heat maps for the full results of this search.

Overall, the model was sensitive to all three parameters, though the κ parameter did not produce major variations in F-score within the range we searched. More broadly, there was a robust range of parameter values within which the model produced results with greater F-scores than the comparison models (roughly all red and yellow regions).

In order to remove the dependence of the model on the free parameters κ and γ, we used empirical Bayesian methods (Carlin & Louis, 1997) to set these two parameters to the values which produced the highest posterior score, maximizing the F-score by fitting only the α parameter. This modification only slightly reduced the maximum F-score of the best lexicon (from .55 to .46).

5.2 Intention reading

We used our hand-coded intentions to evaluate the highest posterior-score intentions for each situation in the corpus given the MAP lexicon. Code for these simulations is in `experimental_models/scoreIntent.mat`. For each model, we found intents for each situation by looking up in that model's lexicon the objects corresponding to each word in the utterance. This simple procedure produces the same result in our model as scoring the sentence under Equation 2 and finding the value of I that receives the highest posterior score.

6 Modeling behavioral experiments

Here we give some extra details of the simulations of experimental results that we include in our paper. Again, rather than focusing on the conceptual aspects of the simulations (which we describe in the paper), we concentrate on the details of how they are implemented.

6.1 Rapid cross-situational word learning

In the script `crossSituationalLearning.m`, we use `crossSituationalCorpus.m` to generate the materials from Yu & Smith (2007). We chose the 4x4 condition—in which 4 objects are presented along with 4 words—as our example, in order to evaluate the models on the most demanding condition of that experiment. We then compute frequency counts as a proof-of-concept to show that any of the baseline models can capture this result. In fact, a basic association frequency model gets an F-score of 1 when the association matrix is thresholded properly. The association matrix corresponding to this simulation is given in Figure 2. We next run the Bayesian model. No special search techniques are needed to infer the correct result here; simple greedy search reliably converges to the correct solution. This simulation is a good way of gaining a better understanding of the workings of the Bayesian model without needing to understand the details of inference.

6.2 Mutual exclusivity

Simulations of the canonical mutual exclusivity experiment are given in `mutualExclusivity.m`. Our method in this simulation and the speakers' intentions simulation (Section 6.5) is the same. We use the MAP lexicon and the CHILDES corpus referred to throughout; we then add an extra situation to the corpus corresponding to the experiment. We evaluate a set of possible lexicons on the original corpus extended with the new situation, comparing their posterior scores. Table 3 gives the results of these simulations (they are also shown graphically in Figure 3 of the main paper). The four lexicons we evaluate are:

- The original MAP lexicon inferred over the corpus. This proposal corresponds to learning nothing from the experimental situation (participants would be at chance in a forced choice decision with the prompt "can you give me the dax?").
- A lexicon which includes the correct mapping, from "dax" to DAX.
- A lexicon which includes the incorrect mapping, from "dax" to BIRD.
- A lexicon which includes both the correct and incorrect mappings.

We review in detail the reasons that the hypothesis containing the correct mapping is preferred. As can be seen in Table 3, hypotheses which include the incorrect mapping score poorly on the corpus likelihood. Each time the word "bird" is used to refer to the object BIRD, the probability of choosing "bird" is reduced, since "dax" is now a viable option. Put another way, it is now a coincidence that "bird" is consistently used to refer to a BIRD and that "dax" was not uttered. Why is the corpus likelihood actually higher (less negative) for a hypothesis with the correct mapping over the original, learn-nothing hypothesis? Every time a word is added to the lexicon, the probability of other words being used non-referentially increases

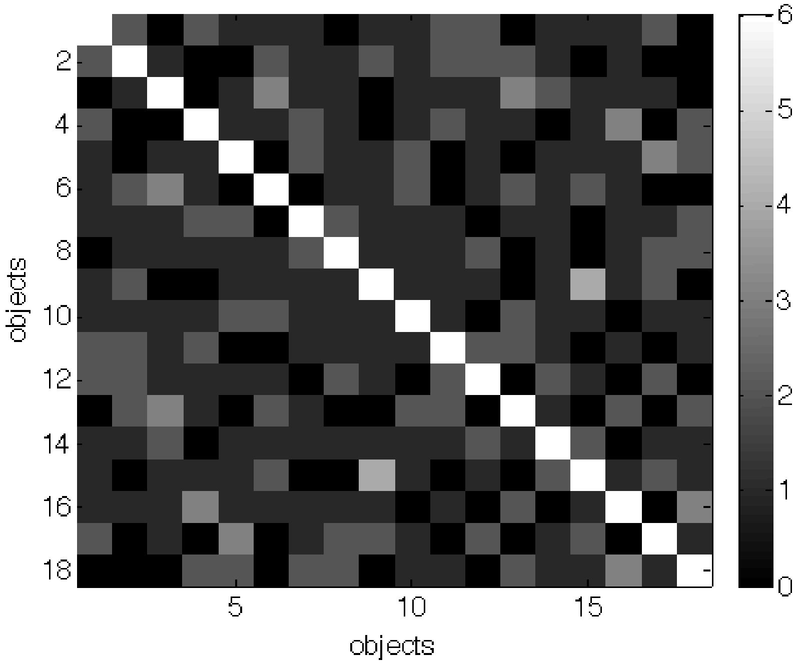

Figure 2: An example cross-situational association matrix from the 4x4 condition in Yu & Smith (2007). Squares represent the association frequency between a word and an object; the lighter the square the greater the association. The white diagonal indicates that the strongest association in this experiment is between word n and object n. Thus any associative model should be able to capture the regularities in this experiment.

Hypothesis	Corpus likelihood	ME situation likelihood	Prior	Posterior score
Learn nothing	-14760.59	-6.02	-168.00	-14934.61
Correct mapping	-14754.77	-3.28	-175.00	-14933.05
Incorrect mapping	-14773.47	-3.98	-175.00	-14952.44
Both mappings	-14773.47	-2.88	-182.00	-14958.34

Table 3: Results for mutual exclusivity simulations. All probabilities are given as natural logarithms.

very slightly (because the number of possible non-referential, non-lexicon words is smaller). Thus, under the correct mapping hypothesis, there is actually a slightly larger probability that a non-referential word like "the" will be chosen in a non-referential context, since there are fewer non-referential words to choose from in comparison with the learn-nothing hypothesis.

The likelihood of the ME situation reflects the probability of this situation coming to pass under the hypotheses about the lexicon. If "dax" refers to neither object, then it was uttered at random (non-referentially). In contrast, if it corresponds to DAX, then it was the only possible way to talk about a DAX and was very likely to be uttered. If it corresponds to BIRD, it is still relatively likely to be uttered, but "bird" could also be uttered. Finally, if both mappings are part of the lexicon it is highly likely that the word "dax" will be uttered to talk about one or the other of the objects.

Finally, prior probabilities are easily interpreted: the more mappings a lexicon contains, the lower its prior probability will be. Thus, the lexicon with both mappings has the lowest prior probability and the lexicon with neither has the highest. Note that there is some parameter dependence in our model's fit to the mutual exclusivity situation. Depending on the size of the corpus, it might be the case that the prior disadvantage of adding a word to the lexicon would not be outweighed by the increase in corpus likelihood caused by learning a new word. This fact makes a developmental prediction: in early development, when very few words are known, inferences about mutual exclusivity should be weaker.

6.3 One-trial learning

The mutual exclusivity situation described above represents one possible situation where the model (and language learners) may be able to learn words with only one or a small number of exposures. Thus, we do not include separate simulations for one-trial learning.

6.4 Object individuation

The script `objectIndividuation.m` contains our simulations of the experiments in Xu (2002). Two groups of infants participated in Xu's experiment (not including the baseline group). One group heard two words ("duck" and "ball") while the other heard only one word ("toy"). At test, each group saw two test displays, one containing one object and one containing two objects. We treat these test displays as interpretations or "construals" of the evidence given in the initial familiarization displays. Thus, in the two-word condition, the question our model answers is "how surprising is it that there were two objects behind the screen, given that I saw only one object every time but I heard two different words?" (One necessary assumption in interpreting Xu's results is that 9-month-olds are not able to remember the identities of the objects across trials).

We simulate this by creating corpora for each condition and interpretation (two-words, one-object construal; two-words, two-object construal; one-word, one-object construal; one-word, two-object construal). An example of the two situations in the two-word, one-object construal corpus is below:

```
cond1_interp2(1).objects = 1;
cond1_interp2(1).words = 1;
cond1_interp2(2).objects = 1;
cond1_interp2(2).words = 2;
```

Hypothesis	Corpus likelihood	Baldwin situation likelihood	Prior	Posterior score
Learn nothing	-14760.59	-5.98	-168.00	-14934.57
Correct mapping	-14754.77	-2.30	-175.00	-14932.07
Incorrect mapping	-14754.77	-10.58	-175.00	-14940.35

Table 4: Results for Baldwin speaker intentions simulations. All probabilities are given as natural logarithms.

In the first situation, object 1 and word 1 are both present; in the second situation, object 1 and word 2 are present.

We further assume that the infants in the experiment (who were 9 months old) do not know the meanings of "duck" and "ball." Thus, we integrate over all possible lexicons that a learner could hypothesize in each experimental situation. We list all possible lexicons using the subfunction `getAllLexicons.m`, which creates the maximally connected lexicon and then enumerates all lexicons with subsets of these connections. We then score each of these possible lexicons and sum over their posterior scores to get the lexicon-independent posterior score of each corpus (summarizing how probable each corpus would be under the model, assuming maximum uncertainty about what the particular words being used actually meant).

We then normalize these probabilities within each condition, since presumably infants in each condition are only making a comparison between the two possible test stimuli they are exposed to. Finally, to link these probabilities to looking-times, we compute the surprisal (negative log probability) for each interpretation. The use of surprisal captures an important main effect in the original data: since in the one-word condition there are fewer possible lexicons and less difference between the lexicons that are possible, the relative probabilities of the two hypotheses are both closer to .5 and hence have lower surprisal. This overall lower level of surprisal corresponds to an overall lower looking time by infants in this condition.

Note that there is one further assumption in this simulation that affects quantitative fit of the model to the looking-time data: the number of other words in the vocabulary. If this number is small, differences between conditions will be very small because P_{NR} will be very similar to P_R. On the other hand, if the vocabulary of the language is large, then P_{NR} will be very small, leading to large penalties for non-referring use of words. While this parameter does not affect the qualitative pattern (a crossover interaction is present at all values), the size of the difference between conditions depends on it.

6.5 Using speakers' intentions

In our final set of simulations, roughly approximating the experiments of Baldwin (1993) (given in the script `speakerIntentions.m`), we model the use of directly observed information about the speaker's intentions. The simplest version of this situation is one in which two objects are present and the speaker looks, points, or otherwise indicates that she is talking about one or the other object and labels that object with a novel word. In this case, the standard cross-situational learning problem is effectively circumvented, since there is only one link worth considering adding to the lexicon: the link between the word that is uttered and the object that is the focus of the speaker's attention (and most likely, her referential intention). The difference here between our approach and other approaches which focus on social attention, however, is that in directly representing the speaker's intentions we are able to extend this basic model to experimental situations like Baldwin's, in which no physical cue links a visible object with a word; instead, the linkage to be learned is between whatever object it is that is the focus of the speaker's referential intention (though this object may only be revealed some time after the word is uttered).

We use the same approach in these simulations as in the simulations of mutual exclusivity above. We create an experimental situation, which we add to the base corpus. We then evaluate three alternative hypotheses for possible lexicons: a lexicon in which no word is learned, a lexicon in which the mapping between the intended object and the word is learned, and a lexicon in which the mapping between the unintended object and the word is learned. The results of these simulations are reported in Table 4. The result is clear: as in the mutual exclusivity simulations, adding any new word increases P_{NR}, thus increasing the corpus likelihood. However, the situation likelihood is overall low in both the "learn nothing" and "incorrect mapping" cases. Thus, the correct mapping is preferred.

7 Conclusions

Within the large literature on word learning, there are likely many phenomena that our model is unable to predict. But we believe that the wide range of results predicted by even this simple model gives a good indication that many other phenomena may be amenable to our approach. The extensibility of our model is a benefit of the Bayesian paradigm: it would be easy to add additional information sources to the model without changing either the general framework or assumptions of the model. To conclude, we briefly mention three possible extensions. First, we have already explored the possibility of adding social cues like eye-gaze and hand position to the information considered by the model in inferring the speakers referential intention (Frank et al., 2008). Rather than biasing the model in favour of particular cues to attention, our model instead learns which cues are most relevant for reference, in the same way as children may, c.f. Hollich et al. (2002). Second, taking into account the hierarchical nature of object labels, a relatively trivial extension of the model would allow it to consider what other objects a particular label should extend to, a question already addressed using the Bayesian approach by Xu & Tenenbaum (2007). Finally, a further exciting direction would be the inclusion of richer representations of events and word-order cues to allow the model to begin to learn verbs as well as simple object nouns.

We hope that this longer exposition of our simulations is useful in clarifying the details of our model. However, if any questions remain, please contact the authors at the addresses given above.

References

Baldwin, D. (1993). Early referential understanding: Infants' ability to recognize acts for what they are. *Developmental Psychology*, *29*(5), 832-843.

Brown, P. F., Pietra, S. D., Pietra, V. J. D., & Mercer, R. L. (1994). The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, *19*(2), 263-311.

Carlin, B., & Louis, T. (1997). Bayes and empirical Bayes method for data analysis. *Statistics and Computing*, *7*(2), 153–154.

Frank, M. C., Goodman, N. D., & Tenenbaum, J. B. (2008). A Bayesian framework for cross-situational word learning. *Neural Information Processing Systems*, *21*.

Hollich, G., Jusczyk, P. W., & Luce, P. (2002). Lexical neighborhood effects in 17-month-old word learning. *Proceedings of the 26th Annual Boston University Conference on Language Development*, *1*.

MacWhinney, B. (2000). *The childes project: Tools for analyzing talk.* New York, NY: Lawrence Erlbaum.

Marinari, E., & Parisi, G. (1992). Simulated tempering: A new Monte Carlo scheme. *Europhysics Letters*, *19*(6), 451-455.

Swingley, D. (2005). Statistical clustering and the contents of the infant vocabulary. *Cognitive Psychology*, *50*, 86-132.

Tu, Z., & Zhu, S. (2002). Image Segmentation by Data-Driven Markov Chain Monte Carlo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 657–673.

Xu, F. (2002). The role of language in acquiring object concepts in infancy. *Cognition*, *85*, 223-250.

Xu, F., & Tenenbaum, J. B. (2007). Word learning as bayesian inference. *Psychological Review*, *114*(2), 245-272.

Yu, C., & Ballard, D. (2007). A unified model of word learning: Integrating statistical and social cues. *Neurocomputing*, *70*(13-15), 2149-2165.

Yu, C., & Smith, L. (2007). Rapid word learning under uncertainty via cross-situational statistics. *Psychological Science*, *18*(5), 414-420.