# Supplemental materials for "Types, not tokens, are relevant for determining how far to generalize"

Amy Perfors
Keith Ransom
Daniel J. Navarro

## 1 The basic model

The first step for accurately modelling this data is to specify a model of grammar learning (or, more accurately, grammar comparison). We use the model specified in Perfors, Tenenbaum, and Regier (2011), in which grammars are scored according to Bayesian probability theory. Please see Perfors et al. (2011) for the full details, but here it is in a nutshell.

The rough idea is that the posterior probability of a grammar given a corpus of sentences is proportional to the prior probability of that grammar times the likelihood of observing those sentences given that grammar. The underlying framework assumes that language is generated according to a multi-stage probabilistic process, and the Bayesian learner inverts this process to learn aspects of the generating grammar from the language data observed. A corpus is generated by first picking a type of grammar $T$ from the prior distribution $p(T)$. A specific grammar is chosen as an instance of that type with probability $p(G|T)$, and the data $D$ is then generated from the specific grammar $G$. The inferences backward are then captured by the joint posterior probability, which according to Bayes rule is:

$$p(G, T|D) \propto p(D|G)p(G|T)p(T) \tag{1}$$

### 1.1 Prior probability

The prior probability of a grammar, $p(G|T)$, reflects its complexity. It is formalized using a model under which each grammar is selected from the space of all grammars of that type. More complex grammars result from making more choices under this generating process. If one were generating a grammar from scratch, one would first have to choose a grammar type (for instance, context-free or regular). Since the model is unbiased, the prior probability of each of these, $p(T)$, is identical. One would then need to choose the number of non-terminals $n$, and for each non-terminal $k$ to generate $P_k$ productions. These $P_k$ productions, which share a left-hand side, are assigned a vector of positive, real-valued production-probability parameters $\theta_k$, which must sum to one. Each production $i$ has $N_i$ right-hand side items, and each of those items must be drawn from the grammar's vocabulary $V$ (set of non-terminals and terminals). If we assume that each right-hand side item is chosen uniformly at random from $V$, the prior probability is given by:

$$p(G|T) = p(n) \prod_{k=1}^{n} p(P_k)p(\theta_k) \prod_{i=1}^{P_k} p(N_i) \prod_{j=1}^{N} \frac{1}{V}. \tag{2}$$

For details about how the probabilities for the number of non-terminals $p(n)$, productions $p(P_k)$, items $p(N_i)$, and production-probability parameters $p(\theta_k)$ are calculated, please see Perfors et al. (2011).

### 1.2 Likelihood

The likelihood $p(D|G)$ is defined straightforwardly by assuming that each sentence in the corpus is generated independently from the grammar. It measures the probability that the corpus data $D$ would be generated by the grammar $G$, which is given by the product of the likelihoods of each sentence $S_\ell$ in the corpus. If there are $M$ unique sentence types, this is given by:

$$p(D|G) = \prod_{\ell=1}^{M} p(S_\ell|G). \tag{3}$$

We make use of the adaptor grammar framework of Johnson, Griffiths, and Goldwater (2007), which captures the intuition that a sentence may be produced either by generating the sentence directly from the grammar, or by calling up a sentence exemplar that had earlier been generated from the grammar and then stored in memory (the adaptor). Thus, there are two components to the underlying language model. The first, the grammar, assigns a probability over the potentially infinite set of syntactic forms that are accepted. The second, the adaptor, produces an observed corpus through a nonparametric stochastic process that combines draws from the generating grammar with draws from a stored memory of previously stored sentence forms – thus interpolating between types and tokens.

If all of the observed tokens are assumed to be drawn from the adaptor, the inference about the grammar is done on the basis of the sentence *types* in the corpus; we call this "type-based" inference. If everything is assumed to be drawn from the grammar, the inference about the grammar takes into account the token frequency; we call this "token-based" inference.

The central metaphor of the adaptor model conceives of the input as consisting of "tables" in a restaurant, and "customers" of the restaurant as corresponding to specific sentence tokens. In a fully type-based analysis, all of the sentence tokens of a given type are seated at the same table. Thus, in a corpus with ten examples of ten unique sentence types, there would be ten tables, each with ten customers. In a fully token-based analysis, each customer would be seated at their own table, making 100 tables, each with only one customer. The essential idea is that when language is produced, sometimes a sentence is generated directly from a grammar, and sometimes it is generated from the memory cache of previous sentences that have been spoken. If it comes from a grammar, it corresponds to adding a customer to a new table of her own; it is relevant for a learner seeking to identify the particular grammar that did the generating. If it is generated from the memory cache, this corresponds to one of the customers (tokens) sitting at an existing table; since it was not generated from the grammar directly it would be sensible for a learner to disregard this sentence token when seeking to identify the grammar.

In practice, we do not usually assume that the learner necessarily knows where sentences came from: they do not come clearly labeled as being generated from either the grammar or the memory cache. Rather, the job for the learner is to figure out how to optimally distribution tokens among tables in such a way as to maximise the probability of the observed sentences given a grammar. Because there are so many ways to distribute tokens to tables, this is a computationally extremely difficult problem. For this analysis we therefore only consider the situations in which either the learner assumes a full token-based analysis or a full type-based analysis. Interpolations between them will come closer to one or the other; this is approximated in Section 2 below.

The adaptor model assigns a probability $p(A)$ to a particular adaptor (i.e., a particular decision about which sentences were generated from the grammar and which were generated from the memory cache). In the calculation of likelihood this element is multiplied by $p(D|G)$ above. The adaptor probability, $p(A)$, is calculated as in Appendix B of Perfors et al. (2011):

$$p(A) = \left( \prod_{k=1}^{K(z)} \theta_{l_k} \right) \frac{\Gamma(K(z))}{\Gamma(N)} \alpha^{K(z)-1} \left( \prod_{k=1}^{K(z)} \frac{\Gamma(n_k^{(z)} - \alpha)}{\Gamma(1-\alpha)} \right)$$

where $z$ is the "seating assignment" (corresponding to all-types or all-tokens). When $\alpha \to 1$, $K(z) = N$, simplifying to the token-based analysis. In this case $p(A)$ simplifies to 1.0. But as $\alpha \to 0$, $K(z) = 10$, since 10 is the number of types there are. For all reported analyses we set $\alpha = 0.05$, but we did evaluate a range of values and they made little qualitative difference to the outcome.

## 1.3 Fitting grammars to input data (corpora)

It is first necessary to establish a set of grammars that we want to evaluate. This is easy enough; the training sentences in the study were generated from one grammar (the Full CFG), and there are several other grammars that generate those sentences and generalize beyond them to different extents. All of these grammars are shown in Table 1.

| Observed | Depth-limited | Full CFG | Any order |
|---|---|---|---|
| S → du gi bo du | S → du X du | S → Y S Y | S → X S |
| S → du la la gi du | S → du du X du du | S → Y X Y | S → X |
| S → du gi gi bo la du | S → du du du X du du du | X → Z X | X → du |
| S → du gi la gi bo du | S → du du du du X du du du du | X → Z | X → gi |
| S → du du bo du du | X → Y X | Y → du | X → la |
| S → du du gi bo gi du du | X → Y | Z → gi | X → bo |
| S → du du la bo gi gi du du | Y → gi | Z → la | |
| S → du du du gi la du du du | Y → la | Z → bo | |
| S → du du du bo gi la du du du | Y → bo | | |
| S → du du du du bo du du du du | | | |

Table 1: Each of the grammars evaluated in this study, in order of increasing looseness of generalization. All are capable of parsing the training corpus. The Observed grammar matches the sentence in that corpus exactly. The Depth-limited grammar is approximately the $A^n B^m A^n$ grammar, but limited to four depths of embedding. The Full CFG is just that grammar. And the Any order grammar allows any of the four vocabulary words $\{du, gi, la, bo\}$ to occur in any order.

First we fit the grammars to the training sentences. This establishes a normative standard about what one *should* conclude about the underlying grammar, given these sentences as input. Here we compare people's preferences to this standard. We can first ask what one should conclude given the corpus of 100 sentences. If the learner is doing token-based inference, then all sentences are assumed to be drawn from the grammar and therefore all sentences are relevant to the grammar. If the learner is doing type-based inference, then only the unique ten sentence types are generated from the grammar, so this is all that enters into the calculations. Posterior probabilities of which grammar is preferred for this corpus are shown in Table 2.

| Grammar | Type-based | Token-based |
|---|---|---|
| Observed | -532.9 | **-439.5** |
| Depth-limited | -442.2 | -731.0 |
| Full CFG | **-398.7** | -697.1 |
| Any order | -429.1 | -1103.3 |

Table 2: Posterior probability of the training sentences in the 10x condition under two different assumptions. Probabilities are in log form, meaning smaller absolute values are higher. In the left column, if the learner is assuming a type-based generative process, the favored grammar is the Full CFG. In the right column, if the learner is assuming a token-based process, the favored grammar is the Observed grammar. Overall, the Full CFG has the highest posterior probability, suggesting that a rational learner should favor the analysis that is closer to type-based in the 100x condition.

If the learner is assuming a type-based generative process, then they are only considering the sentence types (rather than also the frequency of the tokens) when deciding which grammar is best. In this case, the highest-scoring grammar is the Full CFG (which is $10^{13}$ times more likely than the next closest grammar). If the learner assumes a token-based generative process, then they are considering the sentence types as well as the frequency of each token. In that case, the best grammar is the Observed grammar, which does not generalize beyond the training data at all. Taken together, the Full CFG has the highest posterior probability, suggesting that a rational learner should favor the analysis that is closer to type-based in the 100x condition, and – having done so – should favor the Full CFG. The next closest grammar that learners should favor is the Any order grammar. Strikingly, these are the grammars that are most preferred by our participants in the 100x condition (see Figure 3 of the main paper).

What about the 1x condition? In this case, the posterior probabilities of the grammars are shown in Table 3 below. Overall, the Full CFG grammar is favored regardless of whether type-based or token-based inferences are assumed. This makes sense; the number of types and the number of tokens is equivalent, so there should be not qualitative differences between the assumptions in this case.[1] As before, the next

[1] The type-based models are all less probable because the learner pays a price for assuming that customers share tables. Mathematically, this is because we set $\alpha = 0.05$ for the type-based analysis, rather than $\alpha = 1$ for a token-based one.

closest grammar is the Any order one. Again, these are the two grammars favored by the participants in our study in the 1x condition.

| Grammar | Type-based | Token-based |
|---|---|---|
| Observed | -313.2 | -286.2 |
| Depth-limited | -222.5 | -195.5 |
| Full CFG | **-179.0** | **-152.0** |
| Any order | -209.4 | -182.4 |

Table 3: Posterior probability of the training sentences in the 1x condition under two different assumptions. Probabilities are in log form, meaning smaller absolute values are higher. The Full CFG is favored in both cases, followed by the Any order grammar.

Taken together, the fact that participants in both conditions endorsed the Full CFG and Any order grammars is strong evidence that they are assuming that inference should proceed on the basis of sentence types, rather than sentence tokens. If they had assumed that inference was more token-based, we should have seen a strong flip in preferences in the 10x condition to favor the Observed grammar. Intuitively, such a flip would make sense, given this assumption: if you have seen 100 sentences, but only 10 different sentences types, it begins to be an extremely strange coincidence that you haven't seen any others (especially since, in this case, the distribution of tokens to types is uniform; it's not like people saw one very frequent sentence and lots of low-frequency sentences). The fact that people did not seem to be sensitive to this suggests that they are assuming that only the sentence types are relevant to the question of what sentences are grammatical.

We can, however, try to quantify this intuition more precisely, by fitting each individual's responses to an equation capturing what proportion best captures type-based responding and which best captures token-based. This is done in the next section.

# 2 Fitting individual participants

The previous analysis suggests that most people tend to make type-based inferences. Our analysis shows that if one was making token-based inferences, the most preferred grammar in the 10x condition is the one that only can parse the Observed sentences. However, people are very willing to generalise to other sentences, so much so that most people's response patterns are better fit by the Full CFG and Any order grammars, which are favored under the type-based analysis.

How might we quantify the degree of fit of each participant to each type of analysis? We use the insight that the probability of any given sentence $S$ given the training data $D$ is given by:

$$p(S|D) = p(S|G_{any})p(G_{any}|D) + p(S|G_{cfg})p(G_{cfg}|D) + p(S|G_{lim})p(G_{lim}|D) + p(S|G_{obs})p(G_{obs}|D)$$

where each of the $G$ are the grammars of varying levels of generality. $G_{obs}$ is the Observed grammar, $G_{lim}$ corresponds to Depth-limited, $G_{cfg}$ is the Full CFG, and $G_{any}$ is the Any order grammar.

For each grammar, $p(G|D)$ is the posterior probability of the grammar given the training corpus and the adaptor. This corresponds to the probabilities reported in Table 2 and Table 3. For each sentence, $p(S|G)$ is approximated as follows. If the grammar does not parse it, it is zero. If it does, it is calculated according to the likelihood equation above.

This yields two things: $X_{type}$ is the vector of probabilities that one would acquire for each of the 15 test sentences if the learner had assumed that inference was fully type-based; $X_{token}$ is the vector that would result from assuming it was fully token-based. Thus, for each participant we can calculate the weight they put on each kind of analysis by finding the weight $w$ that minimises the following equation:

$$Y - wX_{type} + (1 - w)X_{token}$$

assuming that $Y$ is the vector of the participant responses. (Their ratings were converted to probabilities by assuming that a rating of 1 ("Strongly agree") is a probability of 100%, a rating of 7 is 0%, and scaling accordingly). Thus, $w = 1$ indicates that the person is generalizing exactly like the type-based analysis,

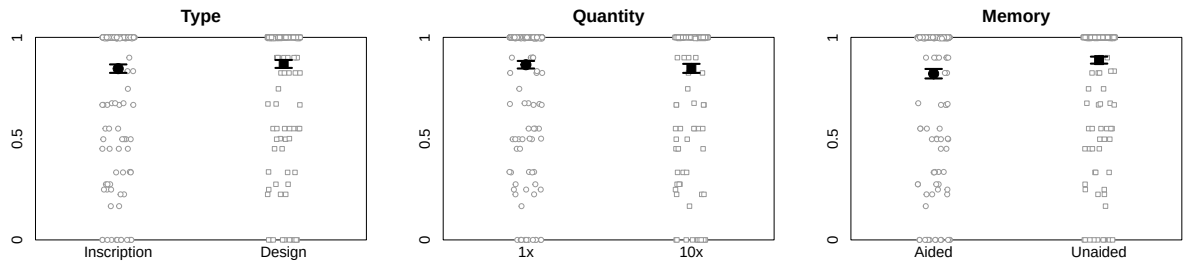


Figure 1: Best fits of of each participant to type-based or token-based analyses. The $y$ axis shows the inferred weights $w$ for each participant. 0 indicates a fully token-based analysis and 1 indicates a fully type-based analysis. On each plot, the mean and standard errors are shown in dark black. Each grey point indicates one participant.The mean $w$ is closer to the fully type-based analysis in all conditions. There is no difference according to the type of stimulus (INSCRIPTION or DESIGN) or the quantity of data (1X or 10X). There was a small difference based on whether people saw a memory aid, although even then, the mean $w$ is still much closer to one (type-based) than zero (token-based).

and $w = 0$ indicates that they are generalizing like the token-based analysis. A weight of zero would mean that they accept the sentences that were observed but no others.

As Figure 1 shows, in all conditions people tended to generalise heavily towards the type-based end of the spectrum: the mean $w$ is closer to the fully type-based analysis, and indeed the majority of people did actually have $w \to 1$. There is no difference according to the type of stimulus (INSCRIPTION or DESIGN) or the quantity of data (1X or 10X). This difference is significant according to a Kruskal-Wallis test, which we used instead of a one-way ANOVA since the distribution of data was non-normal (type of stimulus: $\chi^2(1) == 1.09, p = 0.297$; quantity: $\chi^2(1) = 0.21, p = 0.649$). Consistent with the other analyses in the paper, there was a small difference based on whether people saw a memory aid, tending to be more conservative in their generalizations if they did ($\chi^2(1) = 0.740, p = 0.007$). This is sensible given the interpretation of the adaptor as a memory cache. Even in the AIDED condition, though, the mean $w$ is much closer to type-based than token-based.

# 3 Conclusion

In sum, all of these analyses offer converging evidence of the main conclusion of the paper: people seem to be generally insensitive to token frequencies when performing inferences about what underlying grammar explains the data (or extension, if one is thinking in terms of categorization). It is of course clear that they notice the token frequencies and use them for many other things, but not in this situation, when determining how far to generalize from the hidden data.

# References

Johnson, M., Griffiths, T. L., & Goldwater, S. (2007). Adaptor grammars: A framework for specifying compositional nonparametric Bayesian models. In *NIPS 19* (pp. 641–648).

Perfors, A., Tenenbaum, J., & Regier, T. (2011). The learnability of abstract syntactic principles. *Cognition*, *118*, 306–338.