Productivity and Reuse in Language

Timothy J. O'Donnell (timo@wjh.harvard.edu)
Harvard University Department of Psychology

Jesse Snedeker (snedeker@wjh.harvard.edu)
Harvard University Department of Psychology

Joshua B. Tenenbaum (jbt@mit.edu)
MIT, Brain and Cognitive Science

Noah D. Goodman (ngoodman@stanford.edu)
Stanford University Department of Psychology

Abstract

We present a Bayesian model of the mirror image problems of linguistic productivity and reuse. The model, known as *Fragment Grammar*, is evaluated against several morphological datasets; its performance is compared to competing theoretical accounts including *full–parsing*, *full–listing*, and *exemplar–based* models. The model is able to learn the correct patterns of productivity and reuse for two very different systems: the English past tense which is characterized by a sharp dichotomy in productivity between regular and irregular forms and English derivational morphology which is characterized by a graded cline from very productive (*-ness*) to very unproductive (*-th*).
Keywords: Productivity;Reuse;Storage;Computation; Bayesian Model;Past Tense;Derivational Morphology

Introduction

Perhaps the most celebrated aspect of human language is its creativity. Language allows us to produce, comprehend—and perhaps even think—an unbounded number of thoughts. Creativity in language is made possible by computation. Novel expressions can be produced and understood because the linguistic system provides *productive* computational processes for generating linguistic structures. Productive processes such as syntactic and morphological rules operate via the combination of large numbers of stored, reusable units such as words, morphemes, and idioms. However, not all generalizations that are consistent with the input data are productive, nor is storage and reuse limited to a single kind of unit such as words, but rather both productivity and reuse cut across levels of linguistic structure (Di Sciullo & Williams, 1987). Therefore, a fundamental problem for linguistic and psycholinguistic theory, as well as for the language learner, is understanding which patterns in linguistic data should give rise to productive generalizations and which should give rise to stored, reusable structures.

We present a model of productivity and reuse which addresses the problem as an inference in a Bayesian framework. A productive computation is one which can give rise to novel forms. In a probabilistic setting, if a system hypothesizes that some (sub)computation is productive, it must reserve probability for hitherto unseen structures. On the other hand, if a probabilistic system hypothesizes that some sequence of computations will be frequently reused together, it must reserve probability for that particular sequence as a whole. Since there is only a finite budget of probability, this necessarily leads to a tradeoff: A probabilistic system hypothesizes reusability at the cost of generalization, and vice versa. The model advocated in this paper, *Fragment Grammars* (`FG`), can be seen as optimizing this tradeoff for a given dataset. What patterns in the data signal that some structure is likely to be reused again in the future? What patterns suggest that some class of structures will exhibit novelty and variability and therefore should be computed on the fly?

We examine two morphological systems: the English past tense and English derivational morphology. Morphology provides a domain in which questions of computation and storage have been intensely studied for decades (see, O'Donnell, 2011, for a review). The two datasets examined here are of special interest because of their very different patterns of productivity and reuse. Derivational morphology is characterized by a broad cline of affixes of differing levels of productivity. This sort of gradience is exactly the kind of structure we might expect probabilistic models to excel at capturing (Hay & Baayen, 2005). In contrast, the English past tense's regular *+ed* rule (e.g., *walk/walked*) is highly productive, while the various irregular form classes (e.g., *sing/sang*, *sleep/slept*, etc.) generalize only very rarely (e.g., Prasada & Pinker, 1993). Thus, the English past tense provides an important counterpoint to the gradient structure of derivational morphology. A model of productivity and reuse must be able to handle both kinds of linguistic systems: those with widely varying mixtures of productive computation and reuse, and those with sharp, nearly deterministic, dichotomies between the two.

Informal Overview of the Models

Over the years researchers have proposed many theories of productivity and storage (see O'Donnell, 2011, for a review). In addition to our proposal that productivity and reuse should be considered a probabilistic inference, we also consider three other approaches. Each of these models has been chosen to formalize a specific historical proposal about productivity and reuse, while keeping other dimensions maximally similar to one another.

All of the models considered here start from a underlying system which defines the space of *possible* computations—here we assume that this starting system provides the ability to generate a set of tree–shaped computations like those shown in Figure 1. Each of the approaches has been implemented as a probabilis-

tic model which is defined over this same space of trees; the models differ, however, in the strategies they use to determine which subcomputations (subtrees) can be stored and reused, and which parts of the system can productively generate novel forms.

Full–parsing: In this model, all structures are the result of computation using minimal–sized units. No larger items are stored in memory (Figure 1). In such a setting, each primitive is highly reusable. However, any computation will involve choosing many small, abstract primitives. Such an approach to reuse will be most effective when there is large amount of combinatoriality, variability, and novelty in the data.

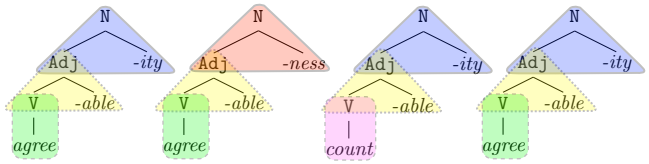

Figure 1: Full–parsing: In this model, small, reusable subcomputations can be shared across many forms, as shown by the highlighting in the same color. However, generating each form requires many random choices.

Full–listing: In this model, the first time a structure is built, it is stored in its entirety. Thus, this system can account for productive generalization, but is nevertheless very conservative—preferring to reuse previously built structures whenever possible (Figure 2). Under such a storage strategy, each stored item is extremely specific, and, therefore, can only be reused in limited contexts. Such an approach to reuse will be most effective when the language consists of a small number of specific, but frequently reused, forms.

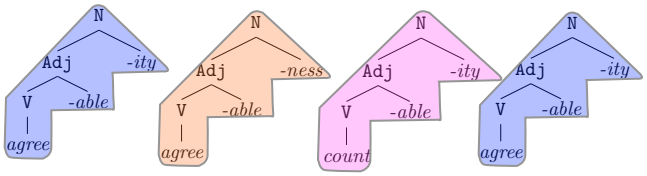

Figure 2: Full–listing: This figure shows the consequences of the full–listing model. Stored subcomputations are very specific. Their substructures cannot be shared with other forms; however, they can be reused in their entirety with high probability.

Exemplar–based Inference: This model stores *all* structures which are consistent with the data, both small and abstract, and large and specific (and all in between). This model differs from Fragment Grammars in that it does not *commit* to a single analysis of each data point, but rather hedges across many different levels of abstraction.

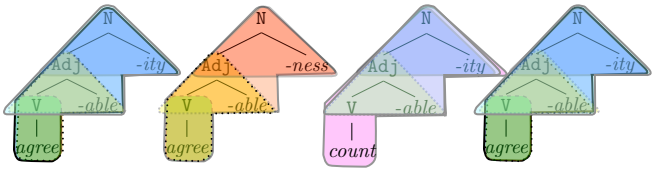

Figure 3: Exemplar–based Inference: This figure shows exemplar–based inference. Here every possible subtree consistent with the data is stored. Note that this leads to many overlapping analyses for each item.

Productivity as an Inference: The model advocated in this paper, Fragment Grammars (FG), treats as an inference the problem of which subcomputations are productive and which should be stored for later reuse.[1] Like the full–parsing approach, FG is able to store abstract structures. Like the full–listing approach, it can store specific structures, and furthermore, like the exemplar–based approach, it can store all intermediate structures. However, unlike the previous approaches, the particular solutions it finds to productivity and reuse are determined by the data itself. For each datapoint to which it is exposed, the model hypothesizes whether it is optimal to account for the structure using reusable stored items, productive computation, or some mixture of both (Figure 4).

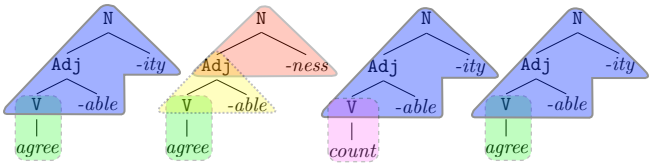

Figure 4: Productivity and Reuse as an Inference: This figure shows the consequences of inferring the set of subcomputations that best account for the data. In this example, more sharing is allowed on average than full–parsing with less computation on average than full–listing.

Formalization of the Models

In this section, we formalize each of the approaches to productivity and reuse introduced above. More detailed treatments can be found in O'Donnell et al. (2009) and O'Donnell (2011).

Full–Parsing: We formalize the full–parsing model using *Multinomial–Dirichlet Probabilistic Context–Free Grammars* (MDPCFG) (Johnson et al., 2007b). An MDPCFG is a 5-tuple, $\mathcal{G} = (N, T, P, S, \Pi)$, where N is the set of nonterminal symbols, T is the set of terminal symbols, R is the set of production rules of the form $A \longrightarrow \gamma$ where $A \in N$ and $\gamma \in (N \cup T)^*$, $S \in N$ is the distinguished

[1] Note that because all of these models are probabilistic, they are all *inferential* in a certain sense. However, only the Fragment Grammar model does inference both over the set of stored subcomputations and points of productivity.

start symbol, and Π is a set of vectors of pseudocounts for the production rules. Define $\ell(t)$ to be the function which returns the label at the root of tree t. By convention we will label the k immediate children of a top node of t as $\hat{t}_i, ..., \hat{t}_k$. The distribution over trees defined by a MDPCFG can be expressed by the simple recurrence below.

$$G_A(t) = \begin{cases} \sum_{r:A\to\ell(\hat{t}_i)\ldots\ell(\hat{t}_k)} \theta_r \prod_{i=1}^{k} G_{\ell(\hat{t}_i)}(\hat{t}_i) & \ell(t) = A \in N \\ 1 & \ell(t) = A \in T \end{cases}$$

$$\vec{\theta_A} \sim \text{DIRICHLET}(\vec{\pi_A})$$

These equations state that the probability of a tree t given by an MDPCFG $\mathcal{G}$ is just the product of the probability of the rules used to build that tree from depth–one subtrees. Note that this corresponds to Figure 1 where each form is composed of minimal fragments of structure. This model puts a Dirichlet prior on the probabilities of fragments. We set all Dirichlet pseudocounts (π's) to 1.

Full–Listing: To formalize the full–listing model we choose *Adaptor Grammars* (AG) (Johnson et al., 2007a). Adaptor Grammars add *reuse* of entire subcomputations to the MDPCFG formalism. This is achieved via *stochastic memoization*. *Memoization* refers to the widely–used technique of storing and reusing the results of function application. Stochastic memoization generalizes this idea by probabilistically mixing the reuse of previously computed results with new calls to the function.

Following Johnson et al. (2007a), we use a distribution for stochastic memoization known as the Pitman-Yor Process (PYP). Let mem$\{f\}$ be a PYP memoized version of some function f. The behavior of a PYP memoized function can be described as follows. The first time we invoke mem$\{f\}$ a new value will be computed using f. On subsequent invocations, we choose an old value i with probability $\frac{n_i - a}{N+b}$, where N is the number of values sampled so far, n_i is the number of times that value i has been used in the past, and $0 \geq a \geq 1$ and $b > -a$ are parameters of the model. Alternatively, we sample a new value with probability $\frac{aK+b}{N+b}$, where K is the number of times a *new* value was sampled in the past from the underlying function. Notice that this process induces a rich-get-richer scheme for sampling from the memoizer. The more a particular value has been reused, the more likely it is to be reused in the future. However, this rich–get–richer dynamic is tempered by a competing bias which favors new values when many new values have been sampled in the past.

Application of stochastic memoization to the basic MDPCFG recurrence leads to the Adaptor Grammars model—which was the first to use this technique (Johnson et al., 2007a).[1] The Adaptor Grammars model can be defined as shown below. Note that since the recursion G_A always returns a full tree down to terminal symbols, Adaptor Grammars can only store and reuse *complete* tree fragments as shown in Figure 2.

$$G_A(t) = \begin{cases} \sum_{r:A\to\ell(\hat{t}_i)\ldots\ell(\hat{t}_k)} \theta_r \prod_{i=1}^{k} \text{mem}\{G_{\ell(\hat{t}_i)}\}(\hat{t}_i) & \ell(t) = A \in N \\ 1 & \ell(t) = A \in T \end{cases}$$

$$\vec{\theta_A} \sim \text{DIRICHLET}(\vec{\pi_A})$$

$$\text{mem}\{G_A\} \sim \text{PYP}(a_A, b_A, G_A)$$

Productivity as an Inference: Adaptor grammars allow the reuse of complete tree fragments. However, a complete tree–fragment cannot be used for further productive computation; it does not allow for novelty or variability. To allow the reuse of productive structures, Fragment Grammars generalize Adaptor Grammars via the technique of *stochastically lazy evaluation*. Stochastically lazy evaluation allows the system to return *delayed* or partially computed items which can remain unspecified, allowing productivity and novelty. These structures can then be reused via memoization, allowing the system to *infer* which structures will exhibit productivity and future novelty.

Define the function prefix to enumerate all *tree prefixes* of a given tree—all fully connected subtrees of the given tree which include the root node. We will write the n leaves of a tree t as $t'_i, ..., t'_n$. The probability of a tree t under a fragment grammar is defined by the following recursive probability mass functions.

$$G_A(t) = \begin{cases} \sum_{s\in\text{prefix}(t)} \text{mem}\{L_A\}(s) \prod_{i=1}^{n} G_{\ell(s'_i)}(s'_i) & \ell(t) = A \in N \\ 1 & \ell(t) = A \in T \end{cases}$$

$$L_A(t) = \sum_{r:A\to\ell(\hat{t}_i)\ldots\ell(\hat{t}_k)} \theta_r \prod_{i=1}^{k} \left[\nu_{r_i} G_{\ell(\hat{t}_i)}(\hat{t}_i) + (1-\nu_{r_i})1\right]$$

$$\nu_{r_i} \sim \text{BETA}(\psi_{\texttt{continue}}, \psi_{\texttt{delay}})$$

$$\vec{\theta_A} \sim \text{DIRICHLET}(\vec{\pi_A})$$

$$\text{mem}\{L_A\} \sim \text{PYP}(a_A, b_A, L_A)$$

The stochastically lazy recurrence, L_A, generates partially evaluated tree fragments, stopping the recursion with probability ν_{r_i} at each internal node of a tree. These partially evaluated tree fragments are then stochastically memoized. Thus, the system is able to learn arbitrary mixtures of reuse and productive computation like those shown in Figure 4.

[1] While Johnson et al. (2007a) do not use the term "stochastic memoization" in describing their model, this notion has its origins in that work, and was first named as such and developed in N. D. Goodman et al. (2008).

Exemplar–Based Inference: We formalize exemplar–based models using two different variants of the *Data–Oriented Parsing* (DOP) formalism for tree–substitution grammar estimation (Bod et al., 2003). All DOP models treat the probability of a tree as a sum over all possible subtrees.

$$G_A(t) = \begin{cases} \sum_{s \in \text{prefix}(t)} \text{PROB}(s) \prod_{i=1}^{n} G_{\ell(s_i')}(s_i') & \ell(t) = A \in N \\ 1 & \ell(t) = A \in T \end{cases}$$

There are many different techniques which fall under the DOP umbrella. They differ primarily in how they estimate the expression PROB in the recurrence above—how they assign probability mass to subtrees. We will use two variants of DOP. The first, known as `DOP1`, sets the probability of a subtree proportional to its token frequency in the training corpus. Let $F_{\mathcal{C}}(s)$ be the frequency of subtree s in corpus $\mathcal{C}$. Then `DOP1` uses the following estimator: $\text{PROB}_{\texttt{DOP1}}(s) \propto F_{\mathcal{C}}(s)$.

A well–known problem with `DOP1` is that, because it counts all subtrees equally, it tends to overweight training data nodes which appear higher and in larger trees (e.g., J. Goodman, 2003). For this reason, we also explore a second DOP estimator. This estimator assigns equal weight to each training data node (Bod, 2003; J. Goodman, 2003). We will call this the *Equal Node DOP* (`ENDOP`).

There are many other variants of DOP. However, in its classical form DOP is committed to the storage of *all* subtrees. It is this idea, in particular, which we wished to explore with our exemplar–based models, and this was our main motivation in choosing the `DOP1` and `ENDOP` estimators.

Other Models: There are several models in the literature which can be seen as explicitly adopting the idea that productivity and reuse should be treated as an (optimal) inference, like the present model. Zuidema (2007) introduced *Parsimonious Data–Oriented Parsing*, a version of DOP which explicitly eschews the all–subtree approach in favor of finding a set of subtrees which best explains the data. Two other models, developed simultaneously with the current framework (Cohn et al., 2009; Post & Gildea, 2009), make use of similar Bayesian non–parametrics to define generative models over sets of subtrees. The models of (Cohn et al., 2009) and Post & Gildea (2009) differ from Fragment Grammars in that they do not define the distribution over stored structures recursively—a new stored item cannot be built out of other stored items. The three models just discussed have been evaluated primarily on syntactic datasets, and primarily in a natural language processing, rather than psychological setting.

Taatgen & Anderson (2002) present a psychologically–motivated model of the past tense which also adopts an optimization perspective (in the ACT–R framework). This model differs from the present one in that Taatgen & Anderson (2002) optimize an objective function involving processing costs, in addition to prediction accuracy (which is the only quantity optimized by Fragment Grammars). They achieve excellent performance on the past–tense domain, and we take this as converging evidence in favor of an optimization perspective in this domain.

Simulations and Inference: For the full–parsing, full–listing, and exemplar–based models, it was possible to directly compute the maximum *a posteriori* grammar from the training corpus. Fragment Grammar results were computed via selective model averaging over a large number of runs of a Markov chain Monte Carlo inference algorithm similar to that found in Johnson et al. (2007a).[1]

The English Past Tense

As discussed in the introduction, the main interest of the English past tense system for present purposes is the wide difference in productivity between the regular rule and the irregular inflectional classes. The English regular, *+ed*, past tense rule (e.g., *walk/walked*) is rampantly productive. A large number of studies have shown generalization of the regular rule to novel stems in both production and rating tasks (e.g., Prasada & Pinker, 1993). The regular rule can be applied to forms which are phonotactically odd for English, to stems derived from other morphological processes, and in a number of other rare and marginal cases (e.g., *He out-Bached Bach.* Kim et al., 1991). On the other hand, while a number of studies have shown that irregular classes can be generalized (e.g., *bring/brang*), this generalization is very rare and much more dependent on the phonological structure of the stem (e.g., Prasada & Pinker, 1993).

The simulations were performed on data extracted from the annotated version of the SwitchBoard corpus included in the Penn TreeBank (Godfrey et al., 1992; Marcus et al., 1999). All verbs excluding forms of *be*, *have* and *do* were extracted from the corpus, lemmatized and paired with the appropriate inflection for the stem.[2] The model was trained on the full English verbal paradigm (i.e., all tenses). Figure 5 shows the simple input representation used for all models. Note that this representation merely pairs stems with their correct inflectional information, it does not explicitly encode any phonological or semantic selectional restrictions. Thus,

[1] Details of the inference engine can be found in O'Donnell (2011); O'Donnell et al. (2009).

[2] These verbs were excluded to reduce the size of the training corpus and, therefore, improve inference run time. There is no theoretical reason that exclusion should have an effect on results for other verbs, and a large pilot study where they were present achieved similar results to the present simulations.

any success that any of the models has in learning contingencies between stems and inflections is the result of distributional information in the input.

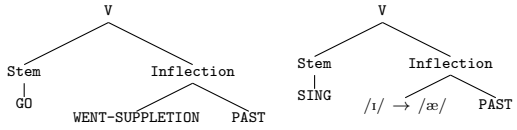

Figure 5: Example Trees for Past Tense: Examples of trees used as inputs to the past tense simulations. These trees represent the past tense forms *went* and *sang* respectively. Note that past tense inflectional classes are identified using their phonological structure.

Figure 6 shows the log odds that a past tense or past participle form sampled at random from the posterior generative model will be correctly inflected. These scores are broken down into regular (*walk*/*walked*) and irregular (*sing*/*sang*) forms that were in the training sample and a set of novel (*wug*/*wugged*) test cases.[1]

Although the full–listing model, `AG`, is able to perform well on attested forms, its generalization performance is more limited compared to `FG`. By contrast, the `DOP1` exemplar–based model performs well on the regular forms and *wug*–generalization cases but fails to learn the irregular forms. Fragment Grammars provide the best simultaneous performance across the three test sets—performing well on novel items as well as correctly learning attested regulars and irregulars.

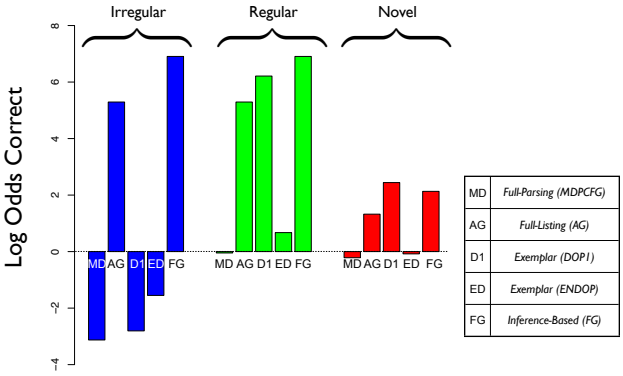

Figure 6: Performance of Models on Past Tense Dataset: The log odds that a form randomly sampled from each trained model will be inflected correctly for the three test sets.

English Derivational Morphology

Having established that the inference–based model is able to account for the sharp differences in productivity in the English past tense system, we turn to the richer and more gradient domain of derivational morphology. We focus here on two aspects of the system. 1. *Gradient Productivity:* Derivational suffixes reside on a productivity cline from very productive (e.g., *-ness*) to very unproductive (e.g., *-th*). 2. *Ordering:* Only a small fraction of the suffix combinations which are possible in principle are attested in actual words. One theory which accounts for this fact is *complexity-based ordering* (CBO; Hay, 2002; Plag & Baayen, 2009). CBO provides a direct link between productivity and ordering by proposing that on average more productive suffixes should appear outside of less productive suffixes.

The input data for our derivational morphology simulations was derived from the CELEX database which contains a large sample of English words derived from dictionaries and newswire (Baayen et al., 1993). Because they undersample low–frequency words, dictionaries tend to underrepresent high–productivity affixes (Evert & Lüdeling, 2001). For this reason, the basic set of morphological parses provided by CELEX was supplemented by applying a heuristic parsing algorithm to the remaining unparsed forms in the database. CELEX parses do not segment cases where one of the segmented parts is not a word, as is the case, for example, with bound stems. To expose this additional structure and correct errors due to automatic parsing, we selected approximately 10,000 of the combined set of forms from CELEX and automatic parsing and corrected them by hand. The resulting data set contained 338 suffixes, over 25,000 word types, and over 7.2 million word tokens. The input to the simulation consisted of trees like those shown in Figure 7.

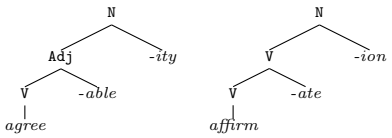

Figure 7: Example Trees for Derivational Morphology: This figure shows examples of the trees used as inputs to the derivational morphology simulations.

We first consider the productivity of suffixes as inferred by the various models. There is no gold standard measure of productivity against which the models can be evaluated. However, two widely used empirical productivity statistics are Baayen's corpus–based measures: $\mathcal{P}$ and $\mathcal{P}^*$ (e.g., Baayen, 1992). The former can be understood as an estimate of the probability that a particular suffix will be used to produce a novel form (i.e., P[NOVEL|SUFFIX]). The latter can be understood as an estimate of the probability that a novel form will use a particular suffix (i.e., P[SUFFIX|NOVEL]).

Table 1 shows the Pearson correlation between the (log) quantities P[NOVEL|SUFFIX] and P[SUFFIX|NOVEL] computed from the various (posterior) models and (log) empirical estimates of $\mathcal{P}$ and $\mathcal{P}^*$ given in Hay & Baayen (2002). Fragment Grammars provide the best fit to these quantities.

Complexity–based ordering predicts a direct link between productivity and suffix ordering—more productive

[1] Regular inflection was considered correct for *wug*–tests.

Model	FG	MDPCFG	AG	DOP1	ENDOP
$\mathcal{P}$	**0.907**	-0.0003	0.692	0.346	0.143
$\mathcal{P}^*$	**0.662**	0.480	0.568	0.402	0.500

Table 1: Correlation with Productivity Measures: The correlation between quantities computed from the trained models and empirical estimates of Baayen's $\mathcal{P}$ and $\mathcal{P}^*$ given in Hay & Baayen (2002).

suffixes should tend to appear outside of less productive suffixes. Plag & Baayen (2009) provide an empirical measure of suffix ordering, based on graph theoretic tools, which gives an estimate of the *mean rank* of a suffix. The mean rank can be understood as a measure of how easily a particular suffix appears after other suffixes in complex words. To generate predictions from the model with regard to suffix ordering, we considered the subset of forms with exactly two suffixes (i.e., *stem -suffix -suffix*) and computed the marginal probability that each suffix occurred first or second in such forms. Table 2 gives the Spearman rank correlation between the (log) ratio of the probability of appearing second to the probability of appearing first with the mean rank statistic reported in Plag & Baayen (2009). As was the case with measures of productivity, Fragment Grammars are best able to account for suffix ordering restrictions.

Model	FG	MDPCFG	AG	DOP1	ENDOP
Mean Rank	**0.568**	0.275	0.424	0.452	0.431

Table 2: Correlation of Suffix Ordering Probabilities and Ranks: The rank correlation between the log odds that a suffix will appear second and the mean rank statistic for the suffix given in Plag & Baayen (2009).

Conclusion

We have presented a model where productivity and reuse are viewed as inferences in a Bayesian framework and have systematically compared this model with four others which represent formalizations of theoretical proposals from the literature. The inference–based model, Fragment Grammars, is best able to capture the patterns of productivity and reuse in two very different sub–systems of English morphology: the past tense and derivational morphology.

In the literature, the problem of balancing productivity and reuse in sometimes discussed in terms of a tradeoff between the cost of computation (in time) and the cost of storage (in space; see, e.g., Frauenfelder & Schreuder, 1992; Yang, 2005). Here we have provided a very different perspective. The tradeoff being optimized here is not between two kinds of machine–level resources, but rather between the ability to predict *future novelty* versus *future reuse*. We believe that this perspective follows naturally from the underlying design of the linguistic system. It is a system which provides the flexibility to produce and comprehend new thoughts while maintaining the ability to specialize for commonly reencountered situations.

Acknowledgments

We would like to thank Marjorie Freedman, Manizeh Khan, and Joshua Hartshorne for detailed feedback on this paper.

References

Baayen, R. H. (1992). Quantitative aspects of morphological productivity. In G. Booij & J. van Marle (Eds.), *Yearbook of morphology 1991* (pp. 109–149). Dordrecht, The Netherlands: Kluwer Academic Publishers.

Baayen, R. H., Piepenbrock, R., & Rijn, H. van. (1993). *The CELEX lexical database.* Linguistic Data Consortium, University of Pennsylvania.

Bod, R. (2003). An efficient implementation of a new DOP model. In *Proceedings of the 10th conference of the European chapter of the association for computational linguistics* (Vol. 1, pp. 19–26). Budapest, Hungary.

Bod, R., Scha, R., & Sima'an, K. (Eds.). (2003). *Data–oriented parsing.* Stanford, CA: CSLI.

Cohn, T., Goldwater, S., & Blunson, P. (2009). Inducing compact but accurate tree–substitution grammars. In *Proceedings of the North American conference on computational linguistics.*

Di Sciullo, A. M., & Williams, E. (1987). *On the definition of word.* Cambridge, MA: MIT Press.

Evert, S., & Lüdeling, A. (2001). Measuring morphological productivity: Is automatic preprocessing sufficient? In *Proceedings of the corpus linguistics 2001 conference.*

Frauenfelder, U. H., & Schreuder, R. (1992). Constraining psycholinguistic models of morphological processing and representation: The role of productivity. In *Yearbook of morphology 1991* (pp. 165–183). Dordrecht, The Netherlands: Springer.

Godfrey, J., Holliman, E., & McDaniel, J. (1992). Switchboard: Telephone speech corpus for research and development. *IEEE ICASSP*, 517-520.

Goodman, J. (2003). Efficient parsing of DOP with PCFG–reductions. In *Data-oriented parsing.* Stanford, CA: CSLI Publications.

Goodman, N. D., Mansinghka, V. K., Roy, D., Bonawitz, K., & Tenenbaum, J. B. (2008). Church: A language for generative models. In *Uncertainty in artificial intelligence.* Helsinki, Finland: AUAI Press.

Hay, J. (2002, September). From speech perception to morphology: Affix ordering revisited. *Language*, *78*(3), 527–555.

Hay, J., & Baayen, R. H. (2002). Parsing and productivity. In *Yearbook of morphology 2001* (Vol. 35, pp. 203–236). Dordrecht, The Netherlands: Springer.

Hay, J., & Baayen, R. H. (2005). Shifting paradigms: Gradient structure in morphology. *Trends in Cognitive Sciences*, *9*(7), 342–348.

Johnson, M., Griffiths, T. L., & Goldwater, S. (2007a). Adaptor Grammars: A framework for specifying compositional nonparametric Bayesian models. In *Advances in neural information processing systems 19.* Cambridge, MA: MIT Press.

Johnson, M., Griffiths, T. L., & Goldwater, S. (2007b). Bayesian inference for PCFGs via Markov chain Monte Carlo. In *Proceedings of the North American conference on computational linguistics.* Rochester, New York.

Kim, J. J., Pinker, S., Prince, A., & Prasada, S. (1991). Why no mere mortal has ever flown out to center field. *Cognitive Science*, *15*, 173–218.

Marcus, M. P., Santorini, B., Marcinkiewicz, M. A., & Taylor, A. (1999). *Treebank–3* (Tech. Rep.). Philadelphia: Linguistic Data Consortium.

O'Donnell, T. J. (2011). *Productivity and reuse in language.* Unpublished doctoral dissertation, Harvard University.

O'Donnell, T. J., Goodman, N. D., & Tenenbaum, J. B. (2009). *Fragment grammars: Exploring computation and reuse in language* Cambridge, Ma: MIT Computer Science and Artificial Intelligence Laboratory Technical Report Series.

Plag, I., & Baayen, R. H. (2009, March). Suffix ordering and morphological processing. *Language*, *85*(1), 109–152.

Post, M., & Gildea, D. (2009). Bayesian learning of a tree substitution grammar. In *Proceedings of the Joint Conference of the 47th ACL and 4th AFNLP.*

Prasada, S., & Pinker, S. (1993). Generalisation of regular and irregular morphological patterns. *Language and Cognitive Processes*, *8*(1), 1-56.

Taatgen, N. A., & Anderson, J. R. (2002). Why do children learn to say "Broke"? A model of learning the past tense without feedback. *Cognition*, *86*(2), 123–155.

Yang, C. (2005). On productivity. In *Linguistic variation yearbook 5.* John Benjamins.

Zuidema, W. (2007). Parsimonious data-oriented parsing. In *Proceedings of EMNLP–CoNLL 2007.*