This supplementary material consists of several appendices. The main paper can be understood without them; but for the curious reader, the appendices provide additional background, details, results, analysis, and discussion (see also (Dreyer, 2011)). Each appendix is independent of the others, since different appendices may be of interest to different readers.

A Dirichlet Process Mixture Models

Section 4 noted that morphology induction is rather like the clustering that is performed by inference under a *Dirichlet process mixture model* (Antoniak, 1974). The DPMM is trivially obtained from the Dirichlet process (Ferguson, 1973), on which there are many good tutorials. Our purpose in this appendix is

- to briefly present the DPMM in the concrete setting of morphology, for the interested reader;
- to clarify why and how we introduce abstract lexemes, obtaining a minor technical variant of the DPMM (section A.5).

The DPMM provides a Bayesian approach to nonparametric clustering. It is Bayesian because it specifies a *prior* over the mixture model that might have generated the data. It is non-parametric because that mixture model uses an *infinite* number of mixture components (in our setting, an infinite lexicon).

Generating a larger data sample tends to select more of these infinitely many components to generate actual data points. Thus, inference tends to use more clusters to explain how a larger sample was generated. In our setting, the more tokens in our corpus, the more paradigms we will organize them into.

A.1 Parameters of a DPMM

Assume that we are given a (usually infinite) set $\mathcal{L}$ of possible mixture components. That is, each element of $\mathcal{L}$ is a distribution $\ell(w)$ over some space of observable objects w. Commonly $\ell(w)$ is a Gaussian distribution over points $w \in \mathbb{R}^n$. In our setting, $\ell(w)$ is a weighted paradigm, which is a distribution (one with finite support) over strings $w \in \Sigma^*$.

Notice that we are temporarily changing our notation. In the main paper, ℓ denotes an abstract lexeme that is associated with a spelling $\pi_{t,\ell}(s)$ and a probability $H_{t,\ell}(s)$ for each slot $s \in \mathcal{S}_t$. For the moment, however, in order to present the DPMM, we are instead using ℓ to denote an actual mixture component—the distribution over words obtained from these spellings and probabilities. In section A.5 we will motivate the alternative construction used in the main paper, in which ℓ is just an index into the set of possible mixture components.

A DPMM is parameterized by a *concentration parameter* $\alpha > 0$ together with a *base distribution* G over the mixture components $\mathcal{L}$. Thus, G states what typical Gaussians or weighted paradigms ought to look like.[21] (What variances σ^2 and means μ are likely? What affixes and stem changes are likely?) In our setting, this is a global property of the language and is determined by the grammar parameters $\vec{\theta}$.

A.2 Sampling a Specific Mixture Model

To draw a specific mixture model from the DPMM prior, we can use a *stick-breaking process*. The idea is to generate an infinite *sequence* of mixture components $\ell^{(1)}, \ell^{(2)}, \ldots$ as IID samples from G. In our setting, this is a sequence of weighted paradigms.

We then associate a probability $\beta_k > 0$ with each component, where $\sum_{k=1}^{\infty} \beta_k = 1$. These β_k probabilities serve as the mixture weights. They are chosen sequentially, in a way that tends to decrease. Thus, the paradigms that fall early in the $\ell^{(k)}$ sequence tend to be the high-frequency paradigms of the language. These ℓ values do not necessarily have high prior probability under G. However, a paradigm ℓ that is very unlikely under G will probably not be chosen anywhere early in the $\ell^{(k)}$ sequence, and so will end up with a low probability.

Specifically: having already chosen $\beta_1, \ldots, \beta_{k-1}$, we set β_k to be the remaining probability mass $1 - \sum_{i=1}^{k-1} \beta_i$ times a random fraction in (0,1) that is drawn IID from $\mathrm{Beta}(1, \alpha)$.[22] Metaphorically, having already broken $k - 1$ segments off a stick of length 1, representing the total probability mass, we now break off a random fraction of what is left of the stick. We label the new stick segment with $\ell^{(k)}$.

This distribution β over the integers yields a distribution G_t over mixture components: $G_t(\ell) =$

---

[21] The traditional name for the base distribution is G_0. We depart from this notation since we want to use the subscript position instead to distinguish draws from the DPMM, e.g., G_t.

[22] Equivalently, letting β'_k be the random fraction, we can define $\beta_k = \beta'_k \cdot \prod_{i=1}^{k-1}(1 - \beta'_i)$.

$\sum_{k:\ell^{(k)}=\ell} \beta_k$, the probability of selecting a stick segment labeled with ℓ. Clearly, this probability is positive if and only if ℓ is in $\{\ell^{(1)}, \ell^{(2)}, \ldots\}$, a countable subset of $\mathcal{L}$ that is countably infinite provided that G has infinite support.

As Sethuraman (1994) shows, this G_t is distributed according to the *Dirichlet process* $\mathrm{DP}(G, \alpha)$. G_t is a discrete distribution and tends to place its mass on mixture components that have high probability under G. In fact, the *average* value of G_t is exactly G. However, G_t is not identical to G, and different samples G_t will differ considerably from one another if α is small.[23] In short, G is the mean of the Dirichlet process while α is inversely related to its variance.

A.3 α for a Natural Language Lexicon

We expect α to be relatively small in our model, since a lexicon is idiosyncratic: G_t does not look too much like G. Many verb paradigms ℓ that would be *a priori* reasonable in the language (large $G(\ell)$) are in fact missing from the dictionary and are only available as neologisms (small $G_t(\ell)$). Conversely, irregular paradigms (small $G(\ell)$) are often selected for frequent use in the language (large $G_t(\ell)$).

On the other hand, small α implies that we will break off large stick segments and most of the probability mass will rapidly be used up on a small number of paradigms. This is not true: the distribution over words in a language has a long tail (Zipf's Law). In fact, regardless of α, Dirichlet processes never capture the heavy-tailed, power-law behavior that is typical of linguistic distributions. The standard solution is to switch to the Pitman-Yor process (Pitman and Yor, 1997; Teh, 2006), a variant on the Dirichlet process that has an extra parameter to control the heaviness of the tail. We have not yet implemented this simple improvement.

A.4 Sampling *from* the Mixture Model

The distribution over paradigms, G_t, gives rise to a distribution over words, $p_t(w) = \sum_\ell G_t(\ell)\, c(w)$. To sample a word w from this distribution, one can sample a mixture component $\ell \sim G_t$ and then a point $w \sim \ell$. This is what sections 6.6–6.8 do. To generate a whole corpus of n words, one must repeat these two steps n times.

For simplicity of notation, let us assume that t is fixed, so that all words are generated from the same G_t (i.e., they have the same part-of-speech tag).[24]

Thus, we need to sample a sequence $\ell_1, \ell_2, \ldots, \ell_n \sim G_t$. Is there a way to do this without explicitly representing the particular infinite mixture model $G_t \sim \mathrm{DP}(G, \alpha)$? It does not matter here that each ℓ_i is a mixture component. What matters is that it is a sample ℓ from a sample G_t from a Dirichlet process. Hence we can use standard techniques for working with Dirichlet processes.

The solution is the scheme known as a *Chinese restaurant process* (Blackwell and MacQueen, 1973; Aldous, 1985), as employed in section 6.9. Our n IID draws from G_t are conditionally independent given G_t (by definition), but they become interdependent if G_t is not observed. This is because $\ell_1, \ldots, \ell_{i-1}$ provide evidence about what G_t must have been. The next sample ℓ_i must then be drawn from the mean of this posterior over G_t (which becomes sharper and sharper as i increases and we gain knowledge of G_t).

It turns out that the posterior mean assigns to each $\ell \in \mathcal{L}$ a probability that is proportional to $t(\ell) + \alpha G(\ell)$, where $t(\ell)$ is the number of previous samples equal to ℓ. The Chinese restaurant process samples from this distribution by using the scheme described in section 6.9.

For a given ℓ, each table in the Chinese restaurant labeled with ℓ corresponds to some stick segment k that is labeled with ℓ. However, unlike the stick segment, the table does not record the value of k. It also does not represent the segment length β_k—although the fraction of customers who are sitting at the table does provide some information about β_k, and the posterior distribution over β_k gets sharper as more customers enter the restaurant. In short, the Chinese restaurant representation is more collapsed than the stick-breaking representation, since it does not record G_t nor a specific stick-breaking construction of G_t.

A.5 Lexemes

We now make lexemes and inflections into first-class variables of the model, which can be inferred (section 7), directly observed (Appendix C), modulated by additional factors (Appendix G), or associated

---

[23] As $\alpha \to 0$, the expected divergence $\mathrm{KL}(G_t || G) \to \mathrm{H}(G)$. As $\alpha \to \infty$, on the other hand, the expected $\mathrm{KL}(G_t || G) \to 0$.

[24] Recall that in reality, we switch among several G_t, one for each part-of-speech tag t. In that case, we use G_{t_i} when sampling the word at position i.

with other linguistic information. The use of first-class lexemes also permits polysemy, where two lexemes remain distinct despite having the same paradigm.

If we used the DPMM directly, our inference procedure would only assign a paradigm ℓ_i to each corpus token i. Given our goals of doing morphological analysis, this formalization is too weak on two grounds:

- We have ignored the structure of the paradigm by treating it as a mere distribution over strings (mixture component). We would like to recover not only the paradigm that generated token i but also the specific responsible slot s_i in that paradigm.
- By tagging only with a paradigm and not with a lexical item, we have failed to disambiguate homographs. For example, ℓ_i says only that `drew` is a form of `draw`. It does not distinguish between `draw`ing a picture and `draw`ing a sample, both of which use the same paradigm.[25]

Regarding the second point, one might object that the two senses of `drew` could be identified with different *weighted* paradigms, which have the same spellings but differ slightly in their weights. However, this escape hatch is not always available. For example, suppose we simplified our model by constraining $H_{t,\ell}$ to equal H_t. Then the different senses of `draw` would have identical weighted paradigms, the weights being imposed by t, and they could no longer be distinguished. To avoid such problems, we think it is prudent to design notation that refers directly to linguistic concepts like abstract lexemes.

Thus, we now switch to the variant view we presented in the main paper, in which each $\ell \in \mathcal{L}$ is an abstract lexeme rather than a mixture component. We still have $G_t \sim \mathrm{DP}(G, \alpha)$, but now G and G_t are distributions over abstract lexemes. The particular mixture components are obtained by choosing a structured paradigm $\pi_{t,\ell}$ and weights $H_{t,\ell}$ for each abstract lexeme ℓ (sections 6.3–6.4). In principle, semantic and subcategorization information could also be associated with the lexeme.

In our sampler, a newly created Chinese restaurant table ought to sample a label $\ell \in \mathcal{L}$ from G. This is straightforward but turns out to be unnecessary, since the particular value of ℓ does not matter in our model. All that matters is the information that we associated with ℓ above. In effect, ℓ is just an arbitrary pointer to the lexical entry containing this information. Thus, in practice, we collapse out the value ℓ and take the lexical entry information to be associated directly with the Chinese restaurant table instead.[26]

We can identify lexemes with Chinese restaurant tables in this way provided that G has no atoms (i.e., any particular $\ell \in \mathcal{L}$ has infinitesimal probability under G), as is also true for the standard Gaussian DPMM. Then, in the generative processes above, two tables (or two stick segments) never happen to choose the same lexeme. So there is no possibility that a single lexeme's customers are split among multiple tables. As a result, we never have to worry about combining customers from multiple tables in order to estimate properties of a lexeme (e.g., when estimating its paradigm by loopy belief propagation).

For concreteness, we can take lexeme space $\mathcal{L}$ to be the uncountable set $[0, 1]$ (see footnote 9), and let G be the uniform distribution over $[0, 1]$. However, these specific choices are not important, provided that G has no atoms and the model does not make any reference to the structure of $\mathcal{L}$.

B Graphical Model Diagram

We provide in Fig. 3 a drawing of our graphical model, corresponding to section 6.

The model is simpler than it appears. First, the variables D_t, H_t, G, and w_i are merely deterministic functions of their parents. Second, several of the edges denote only simple "switching variable" dependencies. These are the thin edges connecting corpus variables ℓ_i, s_i, w_i to the corpus variables above them. For example, ℓ_i is simply sampled from one of the G_t distributions (section 6.6)—but specifically from G_{t_i}, so it also needs t_i as a parent (thin edge). In the

---

[25] Of course, our current model is too weak to make such sense distinctions successfully, since it does not yet consider context. But we would like it to at least be able to *represent* these distinctions in its tagging. In future work (Appendix G) we would hope to consider context, without overhauling the framework and notation of the present paper.

[26] So, instead of storing the lexeme vector $\vec{\ell}$, which associates a specific lexeme with each token, we only really store a partition (clustering) that says which tokens have the same lexeme. The lexemes play the role of cluster labels, so the fact that they are not identifiable and not stored is typical of a clustering problem.

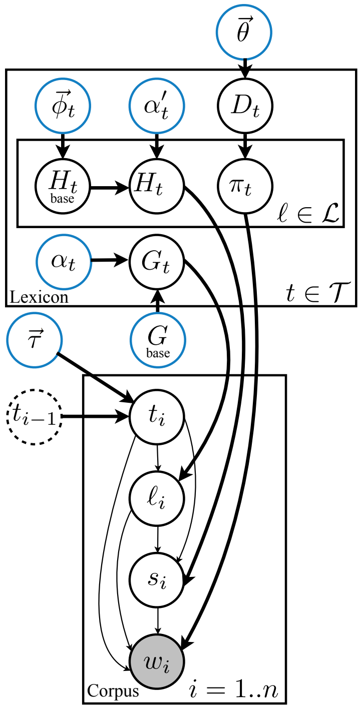

Figure 3: A graphical model drawing of the generative model. As in Fig. 2, type variables are above and token variables are below. Grammar variables are in blue. Although circles denote random variables, we label them with values (such as ℓ_i) to avoid introducing new notation (such as L_i) for the random variables.

same way, s_i is sampled from H_{t_i,ℓ_i}, and w_i deterministically equals $\pi_{t_i,\ell_i}(s)$.

w_i is observed, as shown by the shading. However, the drawing does not show some observations mentioned in section 8.1. First, $\pi_{t,\ell}$ is also observed for certain seed paradigms (t, ℓ). Second, our present experiments also constrain t_i (to a value predicted by an automatic tagger), and then consider only those i for which $t_i =$ VERB.

C Incorporating Known Paradigms

To constrain the inference and training, the method is given a small set of known paradigms of the language (for each tag t). This should help us find a reasonable initial value for $\vec{\theta}$. It can also be given a possibly larger set of *incomplete* known paradigms (section 7.2). In this appendix, we describe how this partial supervision is interpreted and how it affects inference.

Each supervised paradigm, whether complete or incomplete, comes labeled with a lexeme such as *break*. (Ordinarily these lexemes are distinct but that is not required.) This labeling will make it possible to inspect samples from the posterior and evaluate our system on how well it completed the incomplete paradigms of known lexemes (section 8.1), or used known lexemes to annotate word tokens.

In our experiments, each incomplete paradigm specifies *only* the spelling of the lemma inflection. This special inflection is assumed not to be generated in text (i.e., $H_t(\text{lemma}) = 0$). Our main reason for supplying these partial paradigms is to aid evaluation, but it does also provides some weak supervision. Additional supervision of this kind could be obtained through uninflected word lists, which are available for many languages.

Another source of incomplete paradigms would be human informants. In an active learning setting, we might show humans some of the paradigms that we have reconstructed so far, and query them about forms to which we currently assign a high entropy or a high value-of-information.

At any rate, we should condition our inference on any data that we are given. In the standard semi-supervised setting, we would be given a partially annotated corpus (*token* data), and we would run the Gibbs sampler with the observed tags constrained to their observed values. However, the present setting is unusual because we have been given semi-supervised *type* data: a finite set of (partial) paradigms, which is some subset of the infinite lexicon.

To learn from this subset, we must augument our generative story to posit a specific process for how the subset was selected. Suppose the set has k_t semi-supervised paradigms for tag t. We assume that their lexemes were sampled independently (with replacement) from the language's distribution G_t. This assumption is fairly reasonable for the CELEX database that serves as our source of data. It implies that these lexemes tend to have reasonably high probability within the infinite lexicon. Without some such assumption, the subset would provide no information, as footnote 14 explains.[27]

---

[27] The generative story also ought to say how k_t was chosen, and how it was determined which inflectional slots would be observed for each lexeme. However, we assume that these choices are independent of the variables that we are trying to recover, in which case they do not affect our inference. In particular,

It is easy to modify the generative process of section 6 to account for these additional observed samples from G_t. Once we have generated all tokens in the corpus, our posterior estimate of the distribution G_t over lexemes is implicit in the state of the Chinese restaurant t. To sample k_t additional lexemes from G_t, which will be used for the supervised data, we simply see where the *next* k_t customers would sit.

The exchangeability of the Chinese restaurant process means that these additional k_t customers can be treated as the first customers rather than the last ones. We call each of these special customers a **host** because it is at a table without actually taking any particular inflectional seat. It just stands by the table—reserving it, requiring its label to be a particular lexeme such as *break*,[28] and welcoming any future customer that is consistent with the complete or incomplete supervised paradigm at this table. In other words, just as an ordinary customer in a seat constrains a single spelling in the table's paradigm, a host standing at the table constrains the table's paradigm to be consistent with the complete or incomplete paradigm that was observed in semi-supervised data.

To modify our Gibbs sampler, we ensure that the state of a restaurant t includes a **reserved table** for each of the distinct lexemes (such as *break*) in the semi-supervised data. Each reserved table has (at least) one host who stands there permanently, thus permanently constraining some strings in its paradigm. Crucially, the host is included when counting customers at the table. Notice that without the host, ordinary customers would have only an infinitesimal chance of choosing this specific table (lexeme) from all of $\mathcal{L}$, so we would be unlikely to complete this semi-supervised paradigm with corpus words.

The M step is essentially unchanged from the version in Appendix D. Notice that $\vec{\theta}$ will have to account for the partial paradigms at the reserved tables even if only hosts are there (see footnote 30). The hosts are counted in n_t in Equation (D) when estimating α_t. However, as they are not associated with any inflectional seat, they have no effect on estimating α'_t or $\vec{\phi}$. In particular, within Equation (2), interpret $n_{t,\ell}$ as $\sum_s n_{t,\ell,s}$, which excludes hosts.

D Optimizing the Grammar Parameters

Our model has only a finite number of grammar parameters that define global properties of the language (section 6.1). We can therefore maximize their posterior probability

$$\log p(\text{observations} \mid \text{parameters}) + \log p(\text{parameters}) \tag{1}$$

as defined by section 6. This is a case of MAP estimation or empirical Bayes.

The log probability (1) uses the marginal probability of the observations, which requires summing over the possible values of missing variables. But in the case of complete data (no missing variables), (1) takes a simple form: a sum of log probabilities for the different factors in our model. It still takes a simple product form even if the data are complete only with respect to the collapsed model of section 6.9, which does not include variables G_t and $H_{t,\ell}$. This product uses the Chinese restaurant process.

When the observations are incomplete, the Monte Carlo EM method can still be used to seek a local maximum by alternating between

- **E step:** imputing more complete observations $(\vec{w}, \vec{s}, \vec{\ell}, \vec{t})$ by sampling from the posterior $p(\vec{s}, \vec{\ell}, \vec{t} \mid \vec{w}, \vec{\theta}, \ldots)$
- **M step:** optimizing $\vec{\theta}$ to locally maximize the average log-probability (1) of these samples.

Since the M step is working with complete samples, it is maximizing the log of a product. This decomposes into a set of separate supervised maximization problems, as follows.

It is straightforward to train the tag sequence model τ from samples of $\vec{T}$ (section 6.5).

For the collapsed model of lexeme sequences (section 6.9), we train the α_t values. From the probability of obtaining the lexeme sequence $\vec{\ell}$ given $\vec{t}$ under the Chinese restaurant process, it is easy to show that

$$\log p(\vec{\ell} \mid \vec{t}, \alpha_t) = r_t \log \alpha_t - \sum_{i=0}^{n_t-1} \log(i+\alpha_t) + \text{const}$$ [29]

---

we assume that the inflectional slots are missing completely at random (MCAR), just as annotations are assumed to be MCAR in the standard semi-supervised setting.

[28] Other tables created during Gibbs sampling will not have their label constrained in this way. In fact, their labels are collapsed out (Appendix A.5).

[29] The constant accounts for probability mass that does not depend on α_t.

where r_t is the number of tables in restaurant t and n_t is the number of customers in restaurant t. This quantity (and, if desired, its derivative with respect to α_t) may be easily found from $\vec{t}$ and $\hat{\ell}$ for each of our samples. Maximizing its average over our samples is a simple one-dimensional optimization problem.

For the collapsed model of inflection sequences (section 6.9), we similarly train α'_t for each t, and also $\vec{\phi}$. We see from the Chinese restaurant process in section 6.9 that

$$\begin{aligned}
&\log p(\vec{s} \mid \hat{\ell}, \vec{t}, \vec{\phi}, \alpha'_t) \\
&\quad = \sum_{\ell} \left(\sum_{s \in \mathcal{S}_t} \sum_{i=0}^{n_{t,\ell,s}-1} \log(i + \alpha'_t H_t(s)) \right. \\
&\qquad\qquad \left. - \sum_{i=0}^{n_{t,\ell}-1} \log(i + \alpha'_t) \right) + \text{const} \qquad (2)
\end{aligned}$$

where ℓ ranges over the r_t tables in restaurant t, and $n_{t,\ell}$ is the number of customers at table ℓ, of which $n_{t,\ell,s}$ are in seat s. The summation over s may be restricted to s such that $n_{t,\ell,s} > 0$. Recall that H_t is the base distribution over seats: $H_t(s) \propto \exp(\vec{\phi} \cdot \vec{f}(s))$. For a given value of $\vec{\phi}$ and hence H_t, we can easily compute quantity (2) and its gradient with respect to α_t. Its gradient with respect to $\vec{\phi}$ is $\alpha'_t \sum_{s \in \mathcal{S}_t} (c_s - cH_t(s)) \vec{f}(s)$, for $c = \sum_{s' \in \mathcal{S}_t} c_{s'}$ and

$$c_s = H_t(s) \sum_{\ell} \sum_{i=0}^{n_{t,\ell,s}-1} \frac{1}{i + \alpha'_t H_t(s)}$$

Finally, we estimate the parameters $\vec{\theta}$ of our prior distribution D_t over paradigms of the language (section 6.2), to maximize the total log-probability of the partially observed paradigms at the tables in restaurant t (averaged over samples). This can be done with belief propagation, as explained by Dreyer and Eisner (2009). Crucially, each table represents a *single* partially observed sample of D_t, regardless of how many customers chose to sit there.[30] The training formulas consider our posterior distributions over the spellings at the empty seats. In general, however, a table with many empty seats will have less influence on $\vec{\theta}$, and a completely empty table contributes 0 to our total-log-probability objective. This is because the probability of a partially observed paradigm marginalizes over its unseen spellings.

E More Detailed Experimental Results

E.1 Statistics About the Inference Process

Here we briefly list some statistics that give a feel for what is going on during inference. We have 5,415 reserved tables corresponding to the test paradigms (see Appendix C), as well as infinitely many additional tables for lexemes that may have appeared in the corpus but did not appear in test data.

We consider a single sample from inference over 10 million words, and a single sample from inference over 1 million words. The average reserved table had 88 customers in the former case and 11 customers in the latter. Many reserved tables remained completely empty (except for the host lemma)—1,516 tables and 2,978 tables respectively. Furthermore, most inflectional seats remained empty—respectively 96,758 seats and 107,040 seats, among the 113,715 total seats at the reserved tables (21 per table).

E.2 Results by Inflection

Tables 5 and 6 are additions to Table 2. They respectively report whole-word accuracy and edit distance, split by the different forms that were to be predicted.

There is an important remark to make about the inventory of forms. Notice that in cases of syncretism, where two forms are *always* identical in German (e.g., the 1st- and 3rd-person plural indicative past), CELEX uses only a single paradigm slot (e.g., 13PIA) for this shared form. This convention provides additional linguistic knowledge. It means that our $\vec{\theta}$ model does not have to learn that these forms are syncretic: when one form is irregular, then the other is forced to be irregular in exactly the same way.

Without this convention, our results would have suffered more from the simplified star-shaped graphical model given at the end of section 2.2. That model assumes that forms are conditionally independent given the lemma. So among other things, it cannot capture the fact that if a particular verb lemma has a surprising 1PIA form, then its 3PIA form will be

---

[30] In other words, $\vec{\theta}$ generates types, not tokens. Each of the uncountably many lexemes prefers to generate a paradigm that is likely under $\vec{\theta}$ (section 6.2), so the observation of *any* lexeme's paradigm provides information about $\vec{\theta}$. The fact that some tables have higher probability is irrelevant, since (at least in our model) a lexeme's probability is uncorrelated with its paradigm.

	50 seed paradigms			100 seed paradigms		
Form	0	10^6	10^7	0	10^6	10^7
13PIA	74.8	78.3	**81.5**	81.4	82.0	**84.7**
13PIE	**100.0**	99.9	99.8	**100.0**	99.9	99.8
13PKA	74.7	77.8	**82.0**	81.2	81.6	**83.6**
13PKE	**99.9**	99.9	99.7	**99.9**	99.8	99.7
13SIA	84.2	**84.8**	84.6	**85.8**	**85.7**	83.5
13SKA	83.5	87.7	**88.0**	86.2	87.5	**88.2**
13SKE	**99.8**	98.11	98.7	**99.7**	**99.7**	99.4
1SIE	**99.6**	99.3	98.2	**99.5**	**99.5**	98.6
2PIA	**84.0**	81.8	82.0	**85.9**	84.9	85.3
2PIE	98.1	**99.2**	99.2	98.1	**99.3**	99.3
2PKA	**83.0**	79.6	77.2	85.2	**85.4**	84.4
2PKE	**99.9**	**99.9**	99.8	**99.9**	**99.9**	99.9
2SIA	**83.8**	83.3	82.5	85.8	**86.0**	86.0
2SIE	91.1	91.6	**91.9**	94.2	94.5	**94.6**
2SKA	82.2	82.4	**82.7**	85.0	**85.3**	85.1
2SKE	**99.9**	**99.9**	99.9	**99.9**	**99.9**	99.9
3SIE	93.9	95.8	**95.9**	94.4	95.8	**95.9**
pA	59.8	67.8	**70.8**	63.4	69.1	**70.8**
pE	**99.4**	**99.4**	99.4	**99.4**	**99.4**	99.4
rP	97.8	**98.6**	98.3	98.1	**99.1**	99.1
rS	**98.7**	98.4	97.9	98.7	**99.0**	98.9
all	89.9	90.6	**90.9**	91.5	92.0	**92.2**

Table 5: Whole-word accuracy on recovering various inflections. Abbreviations are from CELEX (Baayen et al., 1995); for example, 13PIA means *1st or 3rd **P**lural **I**ndicative p**A**st*. The numbers 0, 10^6 and 10^7 denote the size of the corpus used. We boldface the best result from each 3-way comparison, as well as those that are not significantly worse (paired permutation test, $p < 0.05$).

	50 seed paradigms			100 seed paradigms		
Form	0	10^6	10^7	0	10^6	10^7
13PIA	0.42	0.36	**0.32**	0.34	0.32	**0.29**
13PIE	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
13PKA	0.43	0.37	**0.32**	0.35	0.34	**0.31**
13PKE	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
13SIA	0.43	0.41	**0.39**	0.40	**0.39**	0.40
13SKA	0.34	0.28	**0.27**	0.30	**0.27**	**0.27**
13SKE	**0.00**	0.01	0.01	**0.00**	**0.00**	**0.00**
1SIE	**0.01**	**0.01**	0.02	0.01	**0.00**	0.01
2PIA	**0.39**	0.42	0.44	**0.36**	0.38	0.37
2PIE	0.02	**0.00**	**0.00**	0.02	**0.00**	**0.00**
2PKA	**0.33**	0.38	0.43	0.31	**0.30**	0.34
2PKE	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
2SIA	**0.39**	**0.39**	0.42	**0.36**	**0.36**	0.37
2SIE	0.10	**0.09**	**0.09**	0.07	**0.06**	**0.06**
2SKA	**0.34**	**0.34**	**0.34**	0.31	**0.30**	0.31
2SKE	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
3SIE	0.07	**0.05**	**0.05**	0.07	**0.05**	**0.05**
pA	0.91	0.74	**0.68**	0.84	0.71	**0.69**
pE	0.01	**0.00**	**0.00**	0.01	**0.00**	**0.00**
rP	0.02	**0.01**	**0.01**	0.02	**0.00**	**0.00**
rS	**0.02**	**0.02**	0.03	0.02	**0.01**	**0.01**
all	0.20	0.19	**0.18**	0.18	**0.17**	**0.17**

Table 6: Average edit distance of the predicted morphological forms to the truth. The format is the same as in Table 5.

surprising in exactly the same way. Capturing syncretism and other correlations among inflected forms would require a more sophisticated MRF topology as studied by (Dreyer and Eisner, 2009; Dreyer, 2011). Syncretism in particular is pervasive in morphology: for example, an English verb's past-tense forms are all identical (except for *was*/*were*), and the fact that English does not make certain morphological distinctions at all (e.g., gender) could be regarded as massive syncretism of English on some universal grid for inflectional paradigms.

E.3 Error Analysis

In section 8.2, we saw that adding a corpus helps, but the model still makes some prediction errors, even for some regular verbs, which occur often in the corpus. To explain this, we look at some of these errors.

Table 7 shows some typical errors that are made in the zero-corpus model and corrected by using a corpus. The listed errors are on novel words. Most errors are due to an incorrect application of an irregular rule that was learned from the seed paradigms. The models trained on a corpus learn not to apply these rules in many cases. The seed paradigms are not very representative since they are drawn uniformly at random from all types in CELEX.[31] But from a corpus the model can learn that some phenomena are more frequent than others.

The converse pattern also exists. Even though adding a corpus to to the seed paradigms results in a higher prediction accuracy overall, it can introduce some errors, as shown in Table 8. Here, often a form that is found in the corpus is used instead of the correct one.

For example, the past participle form of *bitzeln* was predicted to be *besselt*. The correct form would be *gebitzelt*, but that does not occur in the corpus, while *besselt* does occur. The pair (*bitzeln*, *besselt*) is also morphologically somewhat plausible considering the correct pair (*sitzen*, *gesessen*) in German.[32] Simi-

---

[31] In the future, we might want to experiment with more representative seed paradigms.

[32] In both pairs, we have the changes `i`→ `e` and `tz`→ `ss`.

Form	Error (no corpus)	Correct	Explanation
aalen, 2PIA	aieltest	aaltest	*ie* as in *(halten, hieltest)*
flügeln, pA	flügelt	geflügelt	no ge- as in *(erinnern, erinnert)*
welken, pA	gewolken	gewelkt	wrong analogy to *(melken, gemolken)*
prüfen, 2SIA	prüfst	prüftest	no *-te-* as in *(rufen, riefst)*

Table 7: Novel words and typical errors that a no-corpus model makes. These errors are corrected in the model that has learned from a corpus. Most errors come from an incorrect application of some irregular rule picked up from the seed paradigms (see the *Explanation* column).

Form	Error (corpus)	Correct	Explanation
bitzeln, pA	besselt	gebitzelt	wrong analogy (see text)
ringeln, 13SIE	riegle	ring(e)le	*unclear*; incorrect rule
silieren, 13PIA	salierten	silierten	*salierten* is observed
bearbeiten, 2SIA	bearbeitest	bearbeitetest	*bearbeitest* is frequent

Table 8: Novel words and typical errors that a corpus model makes. These errors are not made by the no-corpus baseline model. Often, a spelling that can be found in the corpus was preferred instead of the correct spelling.

larly, *salierten* was predicted as a past-tense form of *silieren*. The correct form *silierten* does not occur in the corpus, while *salierten* does. *Salierten* is somewhat plausible due to the common *i*→ *a* change, as in (*bitten*, *baten*), so the morphological grammar did not give a very strong signal to prevent *salierten* from being (mis-)placed in the *silieren* paradigm.

Overall, the errors in Table 8 help explain why the edit distance results in Table 2 improve by only small fractions while the corresponding whole-word accuracy improvements are greater: The corpus models make fewer errors, but the erors they do make can be more severe. In some cases, the corpus component may force a corpus token into a paradigm slot where the finite-state parameters otherwise would have generated a spelling that is closer to the truth. On the other hand, we have seen that the corpus is often helpful in providing evidence on how highly to weight certain irregular constructions in the morphological grammar.

F Evaluation Details

In this section we explain how the token-based evaluation of section 8.2 was conducted. For each (lexeme, inflection) pair (ℓ, s), we needed to estimate the frequency of (ℓ, s) in our 10-million-word corpus to determine which bin it fell into. Since our corpus tokens were not morphologically tagged with (ℓ, s) analyses, we guessed the correct tags with the help of additional supervised type data.[33]

Let $\vec{w}$ denote the 10-million-word corpus. For each word w_i, we wish to guess (ℓ_i, s_i). We exploited all of the 5,615 CELEX paradigms, many more than we used when training. For 99.5% of the spellings in these paradigms, the paradigm ℓ is uniquely determined. Therefore, we simplified by only learning a model to get the distribution over the slot s.

Each verb position in the corpus has a spelling w_i that is consistent with a typically small number of different inflections, $\mathcal{S}_i$, as observed in the CELEX paradigms. For many of the spellings (37.9%), s is uniquely determined, $|\mathcal{S}_i| = 1$. On average $|\mathcal{S}_i| =$ 2.0.

We picked a simple model, since the task is almost entirely supervised and we do not need to predict the exact tags, only an estimate of how often each tag occurs.

Define a log-linear distribution over the slots, $p_{\vec{\lambda}}(s) \propto \exp(\vec{\lambda} \cdot \vec{g}(s))$, where the features extracted by $\vec{g}$ are the obvious properties of the different inflections and combinations thereof, e.g., (3rd-person); (singular); (3rd-person singular); etc. The full in-

---

[33] As described above, the morphological paradigms that we predict are taken from the CELEX morphological database. These forms in those paradigms do have frequency counts attached, but they are not useful for our purposes since they are just spelling frequencies. If various morphological forms have the same spelling they all get the same count.

flection form (e.g., 3rd-person singular indicative) is always a feature as well.

We assumed that the correct slot sequence $\{s_i\}$ was generated from this unigram model. The corpus $\{w_i\}$ together with CELEX gives us partial information about this correct slot sequence, namely that each $s_i \in \mathcal{S}_i$. We therefore fit the parameters $\vec{\lambda}$ by maximizing the regularized log-likelihood of these partial observations:

$$\operatorname*{argmax}_{\vec{\lambda}} \sum_i \log \sum_{s_i \in \mathcal{S}_i} p_{\vec{\lambda}}(s_i) - \frac{1}{2\sigma^2}||\vec{\lambda}||^2$$

We arbitrarily fixed $\sigma^2 = 10$ and run 100 iterations of stochastic gradient descent.[34] This can be implemented in less than 50 lines of Perl code.

After training $\vec{\lambda}$, we computed for each i the posterior distribution over the possible inflections, namely $p(s_i = s \mid s_i \in \mathcal{S}_i) \propto p_{\vec{\lambda}}(s)$ for $s \in \mathcal{S}_i$, and otherwise is 0. We used these posteriors together with ℓ_i to estimate the expected counts of each (ℓ, s) in the corpus. For the very few words whose possible morphological analyses allow more than one lexeme, we assumed a uniform posterior over ℓ_i.

G Future Work: Considering Context

We believe our basic model (see section 6) is a solid starting point for a principled generative account of inflectional (and derivational) morphology. This appendix sketches one important direction for future work.

Context is important for morphological disambiguation (Smith et al., 2005). It is particularly important for unsupervised learning of morphology, since there may be several types of external ("syntagmatic") as well as internal ("paradigmatic") clues to the correct analysis of a word, and these can effectively bootstrap one another during learning (Yarowsky and Wicentowski, 2000).

In particular, inflections can be predicted to some extent from surrounding inflections, lexemes, and tags. In English, for example, verbs tend to agree with their preceding nouns. We see that `broken` is a past participle because like other past participles, it is often preceded by the lexeme *have* (or simply by the particular word `has`, a surface pattern that might be easier to detect in earlier stages of learning). Immediate context is also helpful in predicting lexemes; for example, certain verb lexemes are associated with particular prepositions.

Lexemes are also influenced by wide context. `singed` is not a plausible past tense for `sing`, because it is associated with the same topics as `singe`, not `sing` (Yarowsky and Wicentowski, 2000).

How can we model context? It is easy enough to modulate the probability of the sampler state using a *finite* number of new contextual features whose weights can be learned. These features might consider inflections, tags, and common words. For example, we might learn to lower the probability of sampler states where a verb does not agree in number with the immediately preceding noun. This is simply a matter of multiplying some additional factors into our model (section 6) and renormalizing. This yields a Markov random field (MRF), some of whose factors happen to be distributions over lexemes. The sampler is essentially unchanged, although training in a globally normalized model is more difficult; we expect to use contrastive divergence for training (Hinton, 2002).

It is more difficult to incorporate lexeme-specific feature templates. These would lead to infinitely many feature weights in our non-parametric model, leading to overfitting problems that cannot be solved by regularization. Such feature weights must be integrated out. Three basic techniques seem to be available. We can use richer nonparametric processes that still allow collapsed sampling—e.g., we can use a Hierarchical Dirichlet Process (Teh et al., 2006) to make lexeme probabilities depend on a latent topic, or a Distance-Dependent CRP (Blei and Frazier, 2010) to make lexeme probabilities depend on an arbitrary notion of context. We can multiply together several simple nonparametric processes, thus generating the lexemes by a product of experts. As a last resort, we can always do uncollapsed sampling, which integrates over arbitrary lexeme-specific parameters by including their values explicitly in the state of our MCMC sampler. The sampler state only needs to represent the parameters for its finitely many nonempty tables, but reversible-jump MCMC techniques (Green, 1995) must be used to correctly evaluate the probability of moves that create or destroy tables.

---

[34] We also tried $\sigma^2 = 5$ and found it did not significantly affect the outcome.