Discovering Morphological Paradigms from Plain Text Using a Dirichlet Process Mixture Model

Markus Dreyer*
SDL Language Weaver
Los Angeles, CA 90045, USA
mdreyer@sdl.com

Jason Eisner
Computer Science Dept., Johns Hopkins University
Baltimore, MD 21218, USA
jason@cs.jhu.edu

Abstract

We present an inference algorithm that organizes observed words (tokens) into structured inflectional paradigms (types). It also naturally predicts the spelling of unobserved forms that are missing from these paradigms, and discovers inflectional principles (grammar) that generalize to wholly unobserved words.

Our Bayesian generative model of the data explicitly represents tokens, types, inflections, paradigms, and locally conditioned string edits. It assumes that inflected word tokens are generated from an infinite mixture of inflectional paradigms (string tuples). Each paradigm is sampled all at once from a graphical model, whose potential functions are weighted finite-state transducers with language-specific parameters to be learned. These assumptions naturally lead to an elegant empirical Bayes inference procedure that exploits Monte Carlo EM, belief propagation, and dynamic programming. Given 50–100 seed paradigms, adding a 10-million-word corpus reduces prediction error for morphological inflections by up to 10%.

1 Introduction

1.1 Motivation

Statistical NLP can be difficult for morphologically rich languages. Morphological transformations on words increase the size of the observed vocabulary, which unfortunately masks important generalizations. In Polish, for example, each lexical verb has literally 100 inflected forms (Janecki, 2000). That is, a single *lexeme* may be realized in a corpus as many different word types, which are differently inflected for person, number, gender, tense, mood, etc.

All this makes lexical features even sparser than they would be otherwise. In machine translation or text generation, it is difficult to learn *separately* how to translate, or when to generate, each of these many word types. In text analysis, it is difficult to learn lexical features (as cues to predict topic, syntax, semantics, or the next word), because one must learn a separate feature for each word form, rather than generalizing across inflections.

Our engineering goal is to address these problems by mostly-unsupervised learning of morphology. Our linguistic goal is to build a generative probabilistic model that directly captures the basic representations and relationships assumed by morphologists. This model suffices to *define* a posterior distribution over analyses of any given collection of type and/or token data. Thus we obtain scientific data interpretation as probabilistic inference (Jaynes, 2003). Our computational goal is to *estimate* this posterior distribution.

1.2 What is Estimated

Our inference algorithm jointly reconstructs *token*, *type*, and *grammar* information about a language's morphology. This has not previously been attempted.

Tokens: We will tag each word token in a corpus with (1) a *part-of-speech (POS) tag*,[1] (2) an *inflection*, and (3) a *lexeme*. A token of `broken` might be tagged as (1) a VERB and more specifically as (2) the past participle inflection of (3) the abstract lexeme *break*.[2]

Reconstructing the latent lexemes and inflections allows the features of other statistical models to consider them. A parser may care that `broken` is a past participle; a search engine or question answering system may care that it is a form of *break*; and a translation system may care about both facts.

* This research was done at Johns Hopkins University as part of the first author's dissertation work. It was supported by the Human Language Technology Center of Excellence and by the National Science Foundation under Grant No. 0347822.

[1] POS tagging may be done as part of our Bayesian model or beforehand, as a preprocessing step. Our experiments chose the latter option, and then analyzed only the verbs (see section 8).

[2] We use cursive font for abstract lexemes to emphasize that they are atomic objects that do not decompose into letters.

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 616–627,
Edinburgh, Scotland, UK, July 27–31, 2011.

		singular	plural
present	1st-person	breche	brechen
	2nd-person	brichst	brecht
	3rd-person	bricht	brechen
past	1st-person	brach	brachen
	2nd-person	brachst	bracht
	3rd-person	brach	brachen

Table 1: Part of a morphological paradigm in German, showing the spellings of some inflections of the lexeme *break* (whose lemma is brechen), organized in a grid.

Types: In carrying out the above, we will reconstruct specific *morphological paradigms* of the language. A paradigm is a grid of all the inflected forms of some lexeme, as illustrated in Table 1. Our reconstructed paradigms will include our predictions of inflected forms that were never observed in the corpus. This tabular information about the types (rather than the tokens) of the language may be separately useful, for example in translation and other generation tasks, and we will evaluate its accuracy.

Grammar: We estimate *parameters* $\vec{\theta}$ that describe general patterns in the language. We learn a prior distribution over inflectional paradigms by learning (e.g.) how a verb's suffix or stem vowel tends to change when it is pluralized. We also learn (e.g.) whether singular or plural forms are more common. Our basic strategy is Monte Carlo EM, so these parameters tell us how to guess the paradigms (Monte Carlo E step), then these reconstructed paradigms tell us how to reestimate the parameters (M step), and so on iteratively. We use a few supervised paradigms to initialize the parameters and help reestimate them.

2 Overview of the Model

We begin by sketching the main ideas of our model, first reviewing components that we introduced in earlier papers. Sections 5–7 will give more formal details. Full details and more discussion can be found in the first author's dissertation (Dreyer, 2011).

2.1 Modeling Morphological Alternations

We begin with a family of joint distributions $p(x, y)$ over string pairs, parameterized by $\vec{\theta}$. For example, to model just the semi-systematic relation between a German lemma and its 3rd-person singular present form, one could train $\vec{\theta}$ to maximize the likelihood of (x, y) pairs such as (brechen, bricht). Then, given a lemma x, one could predict its inflected form y via $p(y \mid x)$, and vice-versa.

Dreyer et al. (2008) define such a family via a log-linear model with latent alignments,

$$p(x, y) = \sum_a p(x, y, a) \propto \sum_a \exp(\vec{\theta} \cdot \vec{f}(x, y, a))$$

Here a ranges over monotonic 1-to-1 character alignments between x and y. $\propto$ means "proportional to" (p is normalized to sum to 1). $\vec{f}$ extracts a vector of local features from the aligned pair by examining trigram windows. Thus $\vec{\theta}$ can reward or penalize specific features—e.g., insertions, deletions, or substitutions in specific contexts, as well as trigram features of x and y separately.[3] Inference and training are done by dynamic programming on finite-state transducers.

2.2 Modeling Morphological Paradigms

A paradigm such as Table 1 describes how some abstract lexeme (*break*) is *expressed* in German.[4] We evaluate *whole paradigms* as linguistic objects, following word-and-paradigm or realizational morphology (Matthews, 1972; Stump, 2001). That is, we presume that some language-specific distribution $p(\pi)$ defines whether a paradigm π is a grammatical—and *a priori* likely—way for a lexeme to express itself in the language. Learning $p(\pi)$ helps us reconstruct paradigms, as described at the end of section 1.2.

Let $\pi = (x_1, x_2, \ldots)$. In Dreyer and Eisner (2009), we showed how to model $p(\pi)$ as a renormalized *product* of many pairwise distributions $p_{rs}(x_r, x_s)$, each having the log-linear form of section 2.1:

$$p(\pi) \propto \prod_{r,s} p_{rs}(x_r, x_s) \propto \exp(\sum_{r,s} \vec{\theta} \cdot \overrightarrow{f_{rs}}(x_r, x_s, a_{rs}))$$

This is an undirected graphical model (MRF) over *string-valued* random variables x_s; each factor p_{rs} evaluates the relationship between some pair of strings. Note that it is still a log-linear model, and parameters in $\vec{\theta}$ can be reused across different rs pairs.

To guess at unknown strings in the paradigm, Dreyer and Eisner (2009) show how to perform approximate inference on such an MRF by loopy belief

[3] Dreyer et al. (2008) devise additional helpful features based on enriching the aligned pair with additional latent information, but our present experiments drop those for speed.

[4] Our present experiments focus on orthographic forms, because we are learning from a written corpus. But it would be natural to use phonological forms instead, or to include both in the paradigm so as to model their interrelationships.

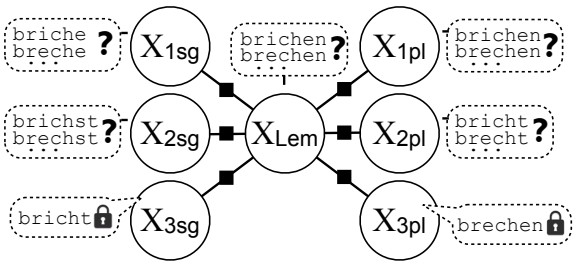

Figure 1: A distribution over paradigms modeled as an MRF over 7 strings. Random variables X_{Lem}, X_{1st}, etc., are the lemma, the 1st person form, etc. Suppose two forms are observed (denoted by the "lock" icon). Given these observations, belief propagation estimates the posterior marginals over the other variables (denoted by "?").

propagation, using finite-state operations. It is not necessary to include all rs pairs. For example, Fig. 1 illustrates the result of belief propagation on a simple MRF whose factors relate all inflected forms to a common (possibly unobserved) lemma, but not directly to one another.[5]

Our method could be used with any $p(\pi)$. To speed up inference (see footnote 7), our present experiments actually use the *directed* graphical model variant of Fig. 1—that is, $p(\pi) = p_1(x_1) \cdot \prod_{s>1} p_{1s}(x_s \mid x_1)$, where x_1 denotes the lemma.

2.3 Modeling the Lexicon (types)

Dreyer and Eisner (2009) learned $\vec{\theta}$ by partially observing some paradigms (type data). That work, while rather accurate at predicting inflected forms, sometimes erred: it predicted spellings that never occurred in text, even for forms that "should" be common. To fix this, we shall incorporate an unlabeled or POS-tagged corpus (token data) into learning.

We therefore need a model for generating tokens—a *probabilistic lexicon* that specifies which inflections of which lexemes are common, and how they are spelled. We do not know our language's probabilistic lexicon, but we assume it was generated as follows:

1. Choose parameters $\vec{\theta}$ of the MRF. This defines $p(\pi)$: which paradigms are likely *a priori*.
2. Choose a distribution over the abstract lexemes.
3. For each lexeme, choose a distribution over its inflections.
4. For each lexeme, choose a paradigm that will be used to express the lexeme orthographically.

Details are given later. Briefly, step 1 samples $\vec{\theta}$ from a Gaussian prior. Step 2 samples a distribution from a Dirichlet process. This chooses a countable number of lexemes to have positive probability in the language, and decides which ones are most common. Step 3 samples a distribution from a Dirichlet. For the lexeme *think*, this might choose to make 1st-person singular more common than for typical verbs. Step 4 just samples IID from $p(\pi)$.

In our model, each part of speech generates its own lexicon: VERBs are inflected differently from NOUNs (different parameters and number of inflections). The size and layout of (e.g.) VERB paradigms is language-specific; we currently assume it is given by a linguist, along with a few supervised VERB paradigms.

2.4 Modeling the Corpus (tokens)

At present, we use only a very simple exchangeable model of the corpus. We assume that each word was independently sampled from the lexicon given its part of speech, with no other attention to context.

For example, a token of `brechen` may have been chosen by choosing frequent lexeme *break* from the VERB lexicon; then choosing 1st-person plural given *break*; and finally looking up that inflection's spelling in *break*'s paradigm. This final lookup is deterministic since the lexicon has already been generated.

3 A Sketch of Inference and Learning

3.1 Gibbs Sampling Over the Corpus

Our job in inference is to reconstruct the lexicon that was used and how each token was generated from it (i.e., which lexeme and inflection?). We use collapsed Gibbs sampling, repeatedly guessing a reanalysis of each token in the context of all others. Gradually, similar tokens get "clustered" into paradigms (section 4).

The state of the sampler is illustrated in Fig. 2. The bottom half shows the current analyses of the verb tokens. Each is associated with a particular slot in some paradigm. We are now trying to reanalyze `brechen` at position ❼. The dashed arrows show some possible analyses.

[5] This view is adopted by some morphological theorists (Albright, 2002; Chan, 2006), although see Appendix E.2 for a caution about syncretism. Note that when the lemma is unobserved, the other forms do still influence one another indirectly.

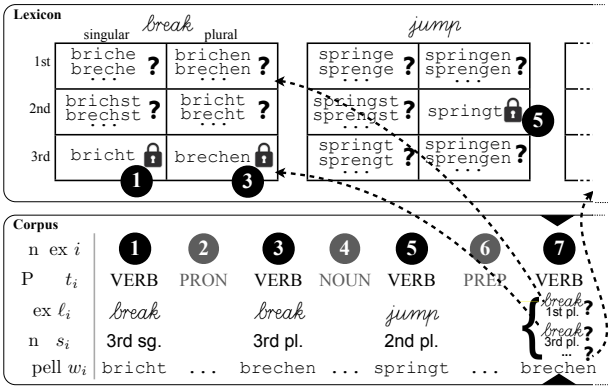

Figure 2: A state of the Gibbs sampler (note that the assumed generative process runs roughly top-to-bottom). Each corpus token i has been tagged with part of speech t_i, lexeme ℓ_i and inflection s_i. Token ❶ has been tagged as *break* and 3rd sg., which locked the corresponding type spelling in the paradigm to the spelling $w_1 =$ `bricht`; similarly for ❸ and ❺. Now w_7 is about to be reanalyzed.

The key intuition is that the current analyses of the *other* verb tokens imply a posterior distribution over the VERB lexicon, shown in the top half of the figure.

First, because of the current analyses of ❶ and ❸, the 3rd-person spellings of *break* are already constrained to match w_1 and w_3 (the "lock" icon).

Second, belief propagation as in Fig. 1 tells us which other inflections of *break* (the "?" icon) are plausibly spelled as `brechen`, and how likely they are to be spelled that way.

Finally, the fact that other tokens are associated with *break* suggest that this is a popular lexeme, making it a plausible explanation of ❼ as well. (This is the "rich get richer" property of the Chinese restaurant process; see section 6.6.) Furthermore, certain inflections of *break* appear to be especially popular.

In short, given the other analyses, we know which inflected lexemes in the lexicon are likely, *and* how likely each one is to be spelled as `brechen`. This lets us compute the relative probabilities of the possible analyses of token ❼, so that the Gibbs sampler can accordingly choose one of these analyses at random.

3.2 Monte Carlo EM Training of $\vec{\theta}$

For a given $\vec{\theta}$, this Gibbs sampler converges to the posterior distribution over analyses of the full corpus. To improve our $\vec{\theta}$ estimate, we periodically adjust $\vec{\theta}$ to maximize or increase the probability of the most recent sample(s). For example, having tagged $w_5 =$ `springt` as $s_5 =$ 2nd-person plural may strengthen our estimated probability that 2nd-person spellings tend to end in `-t`. That revision to $\vec{\theta}$, in turn, will influence future moves of the sampler.

If the sampler is run long enough between calls to the $\vec{\theta}$ optimizer, this is a Monte Carlo EM procedure (see end of section 1.2). It uses the data to optimize a language-specific prior $p(\pi)$ over paradigms—an empirical Bayes approach. (A fully Bayesian approach would resample $\vec{\theta}$ as part of the Gibbs sampler.)

3.3 Collapsed Representation of the Lexicon

The lexicon is *collapsed out* of our sampler, in the sense that we do not represent a single guess about the infinitely many lexeme probabilities and paradigms. What we store about the lexicon is information about its full posterior distribution: the top half of Fig. 2.

Fig. 2 names its lexemes as *break* and *jump* for expository purposes, but of course the sampler cannot reconstruct such labels. Formally, these labels are collapsed out, and we represent lexemes as anonymous objects. Tokens ❶ and ❸ are tagged with the *same* anonymous lexeme (which will correspond to sitting at the same table in a Chinese restaurant process).

For each lexeme ℓ and inflection s, we maintain pointers to any tokens currently tagged with the slot (ℓ, s). We also maintain an approximate marginal distribution over the spelling of that slot:[6]

1. If (ℓ, s) points to at least one token i, then we know (ℓ, s) is spelled as w_i (with probability 1).

2. Otherwise, the spelling of (ℓ, s) is not known. But if some spellings in ℓ's paradigm are known, store a truncated distribution that enumerates the 25 most likely spellings for (ℓ, s), according to loopy belief propagation within the paradigm.

3. Otherwise, we have observed nothing about ℓ: it is currently unused. All such ℓ share the same marginal distribution over spellings of (ℓ, s): the marginal of the prior $p(\pi)$. Here a 25-best list could not cover all plausible spellings. Instead we store a probabilistic finite-state language model that approximates this marginal.[7]

[6] Cases 1 and 2 below must in general be *updated* whenever a slot switches between having 0 and more than 0 tokens. Cases 2 and 3 must be updated when the parameters $\vec{\theta}$ change.

[7] This character trigram model is fast to build if $p(\pi)$ is de-

A hash table based on cases 1 and 2 can now be used to rapidly map any word w to a list of slots of existing lexemes that might plausibly have generated w. To ask whether w might instead be an inflection s of a novel lexeme, we score w using the probabilistic finite-state automata from case 3, one for each s.

The Gibbs sampler randomly chooses one of these analyses. If it chooses the "novel lexeme" option, we create an arbitrary new lexeme object in memory. The number of explicitly represented lexemes is always finite (at most the number of corpus tokens).

4 Interpretation as a Mixture Model

It is common to cluster points in $\mathbb{R}^n$ by assuming that they were generated from a *mixture of Gaussians*, and trying to reconstruct which points were generated from the same Gaussian.

We are similarly clustering word tokens by assuming that they are generated from a *mixture of weighted paradigms*. After all, each word token was obtained by randomly sampling a weighted paradigm (i.e., a cluster) and then randomly sampling a word from it.

Just as each Gaussian in a Gaussian mixture is a distribution over all points $\mathbb{R}^n$, each weighted paradigm is a distribution over all spellings Σ^* (but assigns probability > 0 to only a finite subset of Σ^*).

Inference under our model clusters words together by tagging them with the same lexeme. It tends to group words that are "similar" in the sense that the base distribution $p(\pi)$ predicts that they would tend to co-occur within a paradigm. Suppose a corpus contains several unlikely but similar tokens, such as `discombobulated` and `discombobulating`. A language might have one probable lexeme from whose paradigm all these words were sampled. It is much less likely to have several probable lexemes that all *coincidentally* chose spellings that started with `discombobulat-`. Generating `discombobulat-` only once is cheaper (especially for such a long prefix), so the former explanation has higher probability. This is like explaining nearby points in $\mathbb{R}^n$ as samples from the same Gaussian. Of course, our model is sensitive to more than shared prefixes, and it does not merely cluster words into a paradigm but assigns them to particular inflectional slots in the paradigm.

fined as at the end of section 2.2. If not, one could still try belief propagation; or one could approximate by estimating a language model from the spellings associated with slot s by cases 1 and 2.

4.1 The Dirichlet Process Mixture Model

Our mixture model uses an *infinite* number of mixture components. This avoids placing a prior bound on the number of lexemes or paradigms in the language. We assume that a natural language has an infinite lexicon, although most lexemes have sufficiently low probability that they have not been used in our training corpus or even in human history (yet).

Our specific approach corresponds to a Bayesian technique, the Dirichlet process mixture model. Appendix A (supplementary material) explains the DPMM and discusses it in our context.

The DPMM would standardly be presented as generating a distribution over countably many Gaussians or paradigms. Our variant in section 2.3 instead broke this into two steps: it first generated a distribution over countably many lexemes (step 2), and then generated a weighted paradigm for each lexeme (steps 3–4). This construction keeps distinct lexemes separate even if they happen to have identical paradigms (polysemy). See Appendix A for a full discussion.

5 Formal Notation

5.1 Value Types

We now describe our probability model in more formal detail. It considers the following types of mathematical objects. (We use consistent lowercase letters for values of these types, and consistent fonts for constants of these types.)

A **word** w, such as `broken`, is a finite string of any length, over some finite, given alphabet Σ.

A **part-of-speech tag** t, such as VERB, is an element of a certain finite set $\mathcal{T}$, which in this paper we assume to be given.

An **inflection** s,[8] such as past participle, is an element of a finite set $\mathcal{S}_t$. A token's part-of-speech tag $t \in \mathcal{T}$ determines its set $\mathcal{S}_t$ of possible inflections. For tags that do not inflect, $|\mathcal{S}_t| = 1$. The sets $\mathcal{S}_t$ are language-specific, and we assume in this paper that they are given by a linguist rather than learned. A linguist also specifies features of the inflections: the grid layout in Table 1 shows that 4 of the 12 inflections in $\mathcal{S}_{\text{VERB}}$ share the "2nd-person" feature.

[8] We denote inflections by s because they represent "slots" in paradigms (or, in the metaphor of section 6.7, "seats" at tables in a Chinese restaurant). These slots (or seats) are filled by words.

A **paradigm** for $t \in \mathcal{T}$ is a mapping $\pi : \mathcal{S}_t \to \Sigma^*$, specifying a **spelling** for each inflection in $\mathcal{S}_t$. Table 1 shows one VERB paradigm.

A **lexeme** ℓ is an abstract element of some lexical space $\mathcal{L}$. Lexemes have no internal semantic structure: the only question we can ask about a lexeme is whether it is equal to some other lexeme. There is no upper bound on how many lexemes can be discovered in a text corpus; $\mathcal{L}$ is infinite.

5.2 Random Quantities

Our generative model of the corpus is a joint probability distribution over a collection of random variables. We describe them in the same order as section 1.2.

Tokens: The corpus is represented by *token* variables. In our setting the sequence of words $\vec{w} = w_1, \ldots, w_n \in \Sigma^*$ is observed, along with n. We must recover the corresponding part-of-speech tags $\vec{t} = t_1, \ldots, t_n \in \mathcal{T}$, lexemes $\vec{\ell} = \ell_1, \ldots, \ell_n \in \mathcal{L}$, and inflections $\vec{s} = s_1, \ldots, s_n$, where $(\forall i) s_i \in \mathcal{S}_{t_i}$.

Types: The lexicon is represented by *type* variables. For each of the infinitely many lexemes $\ell \in \mathcal{L}$, and each $t \in \mathcal{T}$, the paradigm $\pi_{t,\ell}$ is a function $\mathcal{S}_t \to \Sigma^*$. For example, Table 1 shows a possible value $\pi_{\text{VERB},\textit{break}}$. The various spellings in the paradigm, such as $\pi_{\text{VERB},\textit{break}}(\text{1st-person sing. pres.})$=`breche`, are string-valued random variables that are correlated with one another.

Since the lexicon is to be probabilistic (section 2.3), $G_t(\ell)$ denotes tag t's distribution over lexemes $\ell \in \mathcal{L}$, while $H_{t,\ell}(s)$ denotes the tagged lexeme (t, ℓ)'s distribution over inflections $s \in \mathcal{S}_t$.

Grammar: Global properties of the language are captured by *grammar* variables that cut across lexical entries: our parameters $\vec{\theta}$ that describe typical inflectional alternations, plus parameters $\vec{\phi}_t, \alpha_t, \alpha'_t, \vec{\tau}$ (explained below). Their values control the overall shape of the probabilistic lexicon that is generated.

6 The Formal Generative Model

We now fully describe the generative process that was sketched in section 2. Step by step, it randomly chooses an assignment to all the random variables of section 5.2. Thus, a given assignment's probability—which section 3's algorithms consult in order to resample or improve the current assignment—is the product of the probabilities of the individual choices, as described in the sections below. (Appendix B provides a drawing of this as a graphical model.)

6.1 Grammar Variables $p(\vec{\theta}), p(\overrightarrow{\phi_t}), p(\alpha_t), p(\alpha'_t)$

First select the grammar variables from a prior. (We will see below how these variables get used.) Our experiments used fairly flat priors. Each weight in $\vec{\theta}$ or $\overrightarrow{\phi_t}$ is drawn IID from $\mathcal{N}(0, 10)$, and each α_t or α'_t from a Gamma with mode 10 and variance 1000.

6.2 Paradigms $p(\pi_{t,\ell} \mid \vec{\theta})$

For each $t \in \mathcal{T}$, let $D_t(\pi)$ denote the distribution over paradigms that was presented in section 2.2 (where it was called $p(\pi)$). D_t is fully specified by our graphical model for paradigms of part of speech t, together with its parameters $\vec{\theta}$ as generated above.

This is the linguistic core of our model. It considers spellings: D_{VERB} describes what verb paradigms typically look like in the language (e.g., Table 1).

Parameters in $\vec{\theta}$ may be shared across parts of speech t. These "backoff" parameters capture general phonotactics of the language, such as prohibited letter bigrams or plausible vowel changes.

For each possible tagged lexeme (t, ℓ), we now draw a paradigm $\pi_{t,\ell}$ from D_t. Most of these lexemes will end up having probability 0 in the language.

6.3 Lexical Distributions $p(G_t \mid \alpha_t)$

We now formalize section 2.3. For each $t \in \mathcal{T}$, the language has a distribution $G_t(\ell)$ over lexemes. We draw G_t from a Dirichlet process $\text{DP}(G, \alpha_t)$, where G is the **base distribution** over $\mathcal{L}$, and $\alpha_t > 0$ is a **concentration parameter** generated above. If α_t is small, then G_t will tend to have the property that most of its probability mass falls on relatively few of the lexemes in $\mathcal{L}_t \stackrel{\text{def}}{=} \{\ell \in \mathcal{L} : G_t(\ell) > 0\}$. A *closed-class tag* is one whose α_t is especially small.

For G to be a uniform distribution over an infinite lexeme set $\mathcal{L}$, we need $\mathcal{L}$ to be uncountable.[9] However, it turns out[10] that with probability 1, each $\mathcal{L}_t$ is *countably* infinite, and all the $\mathcal{L}_t$ are disjoint. So each lexeme $\ell \in \mathcal{L}$ is selected by at most one tag t.

[9] For example, $\mathcal{L} \stackrel{\text{def}}{=} [0, 1]$, so that *break* is merely a suggestive nickname for a lexeme such as 0.2538159.

[10] This can be seen by considering the stick-breaking construction of the Dirichlet process that (Sethuraman, 1994; Teh et al., 2006). A separate stick is broken for each G_t. See Appendix A.

6.4 Inflectional Distributions $p(H_{t,\ell} \mid \overrightarrow{\phi_t}, \alpha'_t)$

For each tagged lexeme (t, ℓ), the language specifies some distribution $H_{t,\ell}$ over its inflections.

First we construct backoff distributions H_t that are independent of ℓ. For each tag $t \in \mathcal{T}$, let H_t be some base distribution over $\mathcal{S}_t$. As $\mathcal{S}_t$ could be large in some languages, we exploit its grid structure (Table 1) to reduce the number of parameters of H_t. We take H_t to be a log-linear distribution with parameters $\overrightarrow{\phi_t}$ that refer to *features* of inflections. E.g., the 2nd-person inflections might be *systematically* rare.

Now we model each $H_{t,\ell}$ as an independent draw from a finite-dimensional Dirichlet distribution with mean H_t and concentration parameter α'_t. E.g., *think* might be biased toward 1st-person sing. present.

6.5 Part-of-Speech Tag Sequence $p(\vec{t} \mid \vec{\tau})$

In our current experiments, $\vec{t}$ is given. But in general, to discover tags and inflections simultaneously, we can suppose that the tag sequence $\vec{t}$ (and its length n) are generated by a Markov model, with tag bigram or trigram probabilities specified by some parameters $\vec{\tau}$.

6.6 Lexemes $p(\ell_i \mid G_{t_i})$

We turn to section 2.4. A lexeme token depends on its tag: draw ℓ_i from G_{t_i}, so $p(\ell_i \mid G_{t_i}) = G_{t_i}(\ell_i)$.

6.7 Inflections $p(s_i \mid H_{t_i,\ell_i})$

An inflection slot depends on its tagged lexeme: we draw s_i from H_{t_i,ℓ_i}, so $p(s_i \mid H_{t_i,\ell_i}) = H_{t_i,\ell_i}(s_i)$.

6.8 Spell-out $p(w_i \mid \pi_{t_i,\ell_i}(s_i))$

Finally, we generate the word w_i through a deterministic *spell-out* step.[11] Given the tag, lexeme, and inflection at position i, we generate the word w_i simply by looking up its spelling in the appropriate paradigm. So $p(w_i \mid \pi_{t_i,\ell_i}(s_i))$ is 1 if $w_i = \pi_{t_i,\ell_i}(s_i)$, else 0.

6.9 Collapsing the Assignment

Again, a *full* assignment's probability is the product of all the above factors (see drawing in Appendix B).

[11] To account for typographical errors in the corpus, the spell-out process could easily be made nondeterministic, with the observed word w_i derived from the correct spelling $\pi_{t_i,\ell_i}(s_i)$ by a noisy channel model (e.g., (Toutanova and Moore, 2002)) represented as a WFST. This would make it possible to analyze `brkoen` as a misspelling of a common or contextually likely word, rather than treating it as an unpronounceable, irregularly inflected neologism, which is presumably less likely.

But computationally, our sampler's state leaves the G_t *unspecified*. So its probability is the integral of $p(\text{assignment})$ over all possible G_t. As G_t appears only in the factors from headings 6.3 and 6.6, we can just integrate it out of *their* product, to get a collapsed sub-model that generates $p(\vec{\ell} \mid \vec{t}, \vec{\alpha})$ directly:

$$\int_{G_{\text{ADJ}}} \cdots \int_{G_{\text{VERB}}} d\mathbf{G} \left(\prod_{t \in \mathcal{T}} p(G_t \mid \alpha_t) \right) \left(\prod_{i=1}^{n} p(\ell_i \mid G_{t_i}) \right)$$
$$= p(\vec{\ell} \mid \vec{t}, \vec{\alpha}) = \prod_{i=1}^{n} p(\ell_i \mid \ell_1, \ldots \ell_{i-1}\ \vec{t}, \vec{\alpha})$$

where it turns out that the factor that generates ℓ_i is proportional to $|\{j < i : \ell_j = \ell_i \text{ and } t_j = t_i\}|$ if that integer is positive, else proportional to $\alpha_{t_i} G(\ell_i)$.

Metaphorically, each tag t is a *Chinese restaurant* whose *tables* are labeled with lexemes. The tokens are hungry *customers*. Each customer $i = 1, 2, \ldots, n$ enters restaurant t_i in turn, and ℓ_i denotes the label of the table she joins. She picks an occupied table with probability proportional to the number of previous customers already there, or with probability proportional to α_{t_i} she starts a new table whose label is drawn from G (it is novel with probability 1, since G gives infinitesimal probability to each old label).

Similarly, we integrate out the infinitely many lexeme-specific distributions $H_{t,\ell}$ from the product of 6.4 and 6.7, replacing it by the collapsed distribution

$$p(\vec{s} \mid \vec{\ell}, \vec{t}, \overrightarrow{\phi_t}, \overrightarrow{\alpha'}) \qquad \textit{[recall that } \overrightarrow{\phi_t} \textit{ determines } H_t\textit{]}$$
$$= \prod_{i=1}^{n} p(s_i \mid s_1, \ldots s_{i-1},\ \vec{\ell}, \vec{t}, \overrightarrow{\phi_t}, \overrightarrow{\alpha'})$$

where the factor for s_i is proportional to $|\{j < i : s_j = s_i \text{ and } (t_j, \ell_j) = (t_i, \ell_i)\}| + \alpha'_{t_i} H_{t_i}(s_i)$.

Metaphorically, each table ℓ in Chinese restaurant t has a fixed, finite set of *seats* corresponding to the inflections $s \in \mathcal{S}_t$. Each seat is really a bench that can hold any number of customers (tokens). When customer i chooses to sit at table ℓ_i, she also chooses a seat s_i at that table (see Fig. 2), choosing either an already occupied seat with probability proportional to the number of customers already in that seat, or else a random seat (sampled from H_{t_i} and not necessarily empty) with probability proportional to α'_{t_i}.

7 Inference and Learning

As section 3 explained, the learner alternates between a Monte Carlo E step that uses Gibbs sampling to

sample from the posterior of $(\vec{s}, \vec{\ell}, \vec{t})$ given $\vec{w}$ and the grammar variables, and an M step that adjusts the grammar variables to maximize the probability of the $(\vec{w}, \vec{s}, \vec{\ell}, \vec{t})$ samples given those variables.

7.1 Block Gibbs Sampling

As in Gibbs sampling for the DPMM, our sampler's basic move is to reanalyze token i (see section 3). This corresponds to making customer i invisible and then guessing where she is probably sitting—which restaurant t, table ℓ, and seat s?—given knowledge of w_i and the locations of all other customers.[12]

Concretely, the sampler guesses location (t_i, ℓ_i, s_i) with probability *proportional* to the product of

- $p(t_i \mid t_{i-1}, t_{i+1}, \vec{\tau})$ (from section 6.5)
- the probability (from section 6.9) that a new customer in restaurant t_i chooses table ℓ_i, given the *other* customers in that restaurant (and α_{t_i})[13]
- the probability (from section 6.9) that a new customer at table ℓ_i chooses seat s_i, given the *other* customers at that table (and $\overrightarrow{\phi_{t_i}}$ and α'_{t_i})[13]
- the probability (from section 3.3's belief propagation) that $\pi_{t_i,\ell_i}(s_i) = w_i$ (given $\vec{\theta}$).

We sample only from the (t_i, ℓ_i, s_i) candidates for which the last factor is non-negligible. These are found with the hash tables and FSAs of section 3.3.

7.2 Semi-Supervised Sampling

Our experiments also consider the semi-supervised case where a few **seed paradigms**—*type* data—are fully or partially observed. Our samples should also be conditioned on these observations. We assume that our supervised list of observed paradigms was generated by sampling from G_t.[14] We can modify our setup for this case: certain tables have a **host** who dictates the spelling of some seats and attracts appropriate customers to the table. See Appendix C.

7.3 Parameter Gradients

Appendix D gives formulas for the M step gradients.

[12] Actually, to improve mixing time, we choose a currently active lexeme ℓ uniformly at random, make *all* customers $\{i : \ell_i = \ell\}$ invisible, and sequentially guess where they are sitting.

[13] This is simple to find thanks to the exchangeability of the CRP, which lets us pretend that i entered the restaurant last.

[14] Implying that they are assigned to lexemes with non-negligible probability. We would learn nothing from a list of merely *possible* paradigms, since $\mathcal{L}_t$ is infinite and every conceivable paradigm is assigned to *some* $\ell \in \mathcal{L}_t$ (in fact many!).

	50 seed paradigms			100 seed paradigms		
Corpus size	0	10^6	10^7	0	10^6	10^7
Accuracy	89.9	90.6	**90.9**	91.5	92.0	**92.2**
Edit dist.	0.20	0.19	**0.18**	0.18	**0.17**	**0.17**

Table 2: Whole-word accuracy and edit distance of predicted inflection forms given the lemma. Edit distance to the correct form is measured in characters. Best numbers per set of seed paradigms in bold (statistically significant on our large test set under a paired permutation test, $p < 0.05$). Appendix E breaks down these results per inflection and gives an error analysis and other statistics.

8 Experiments

8.1 Experimental Design

We evaluated how well our model learns German verbal morphology. As *corpus* we used the first 1 million or 10 million words from WaCky (Baroni et al., 2009). For *seed and test paradigms* we used verbal inflectional paradigms from the CELEX morphological database (Baayen et al., 1995). We fully observed the seed paradigms. For each test paradigm, we observed the lemma type (Appendix C) and evaluated how well the system completed the other 21 forms (see Appendix E.2) in the paradigm.

We simplified inference by fixing the POS tag sequence to the automatic tags delivered with the WaCky corpus. The result that we evaluated for each variable was the value whose probability, averaged over the entire Monte Carlo EM run,[15] was highest. For more details, see (Dreyer, 2011).

All results are averaged over 10 different training/test splits of the CELEX data. Each split sampled 100 paradigms as seed data and used the remaining 5,415 paradigms for evaluation.[16] From the 100 paradigms, we also sampled 50 to obtain results with smaller seed data.[17]

8.2 Results

Type-based Evaluation. Table 2 shows the results of predicting verb inflections, when running with no corpus, versus with an unannotated corpus of size 10^6 and 10^7 words. Just using 50 seed paradigms, but

[15] This includes samples from before $\vec{\theta}$ has converged, somewhat like the voted perceptron (Freund and Schapire, 1999).

[16] 100 further paradigms were held out for future use.

[17] Since these seed paradigms are sampled uniformly from a set of CELEX paradigms, most of them are regular. We actually only used 90 and 40 for training, reserving 10 as development data for sanity checks and for deciding when to stop.

Bin	Frequency	# Verb Forms
1	0–9	116,776
2	10–99	4,623
3	100–999	1,048
4	1,000–9,999	95
5	10,000–	10
all	*any*	122,552

Table 3: The inflected verb forms from 5,615 inflectional paradigms, split into 5 token frequency bins. The frequencies are based on the 10-million word corpus.

no corpus, gives an accuracy of 89.9%. By adding a corpus of 10 million words we reduce the error rate by 10%, corresponding to a one-point increase in absolute accuracy to 90.9%. A similar trend can be seen when we use more seed paradigms. Simply training on 100 seed paradigms, but not using a corpus, results in an accuracy of 91.5%. Adding a corpus of 10 million words to these 100 paradigms reduces the error rate by 8.3%, increasing the absolute accuracy to 92.2%. Compared to the large corpus, the smaller corpus of 1 million words goes more than half the way; it results in error reductions of 6.9% (50 seed paradigms) and 5.8% (100 seed paradigms). Larger unsupervised corpora should help by increasing coverage even more, although Zipf's law implies a diminishing rate of return.[18]

We also tested a baseline that simply inflects each morphological form according to the basic regular German inflection pattern; this reaches an accuracy of only 84.5%.

Token-based Evaluation. We now split our results into different bins: how well do we predict the spellings of frequently expressed (lexeme, inflection) pairs as opposed to rare ones? For example, the third person singular indicative of *give* (`geben`) is used significantly more often than the second person plural subjunctive of *bask* (`aalen`);[19] they are in different frequency bins (Table 3). The more frequent a form is in text, the more likely it is to be irregular (Jurafsky et al., 2000, p. 49).

The results in Table 4 show: Adding a corpus of either 1 or 10 million words increases our prediction accuracy across *all* frequency bins, often dramatically. All methods do best on the huge number of

	50 seed paradigms			100 seed paradigms		
Bin	0	10^6	10^7	0	10^6	10^7
1	90.5	91.0	**91.3**	92.1	92.4	**92.6**
2	78.1	**84.5**	84.4	80.2	**85.5**	85.1
3	71.6	**79.3**	78.1	73.3	**80.2**	79.1
4	57.4	61.4	**61.8**	57.4	**62.0**	59.9
5	20.7	**25.0**	**25.0**	20.7	**25.0**	**25.0**
all	52.6	57.5	**57.8**	53.4	**58.5**	57.8
all (e.d.)	1.18	1.07	**1.03**	1.16	1.02	**1.01**

Table 4: Token-based analysis: Whole-word accuracy results split into different frequency bins. In the last two rows, all predictions are included, weighted by the frequency of the form to predict. Last row is edit distance.

rare forms (Bin 1), which are mostly regular, and worst on on the 10 most frequent forms of the language (Bin 5). However, adding a corpus helps most in fixing the errors in bins with more frequent and hence more irregular verbs: in Bins 2–5 we observe improvements of up to almost 8% absolute percentage points. In Bin 1, the no-corpus baseline is already relatively strong.

Surprisingly, while we always observe gains from using a corpus, the gains from the 10-million-word corpus are sometimes smaller than the gains from the 1-million-word corpus, except in edit distance. Why? The larger corpus mostly adds new infrequent types, biasing $\vec{\theta}$ toward regular morphology at the expense of irregular types. A solution might be to model irregular classes with separate parameters, using the latent conjugation-class model of Dreyer et al. (2008).

Note that, by using a corpus, we even improve our prediction accuracy for forms and spellings that are *not* found in the corpus, i.e., *novel* words. This is thanks to improved grammar parameters. In the token-based analysis above we have already seen that prediction accuracy increases for rare forms (Bin 1). We add two more analyses that more explicitly show our performance on novel words. (a) We find all paradigms that consist of novel spellings only, i.e. none of the correct spellings can be found in the corpus.[20] The whole-word prediction accuracies for the models that use corpus size 0, 1 million, and 10 million words are, respectively, 94.0%, 94.2%, 94.4% using 50 seed paradigms, and 95.1%, 95.3%, 95.2% using 100 seed paradigms. (b) Another, sim-

[18] Considering the 63,778 distinct spellings from all of our 5,615 CELEX paradigms, we find that the smaller corpus contains 7,376 spellings and the 10× larger corpus contains 13,572.

[19] See Appendix F for how this was estimated from text.

[20] This is measured on the largest corpus used in inference, the 10-million-word corpus, so that we can evaluate all models on the same set of paradigms.

pler measure is the prediction accuracy on all forms whose correct spelling cannot be found in the 10-million-word corpus. Here we measure accuracies of 91.6%, 91.8% and 91.8%, respectively, using 50 seed paradigms. With 100 seed paradigms, we have 93.0%, 93.4% and 93.1%. The accuracies for the models that use a corpus are higher, but do not always steadily increase as we increase the corpus size.

The token-based analysis we have conducted here shows the strength of the corpus-based approach presented in this paper. While the integrated graphical models over strings (Dreyer and Eisner, 2009) can learn some basic morphology from the seed paradigms, the added corpus plays an important role in correcting its mistakes, especially for the more frequent, irregular verb forms. For examples of specific errors that the models make, see Appendix E.3.

9 Related Work

Our word-and-paradigm model seamlessly handles nonconcatenative and concatenative morphology alike, whereas most previous work in morphological knowledge discovery has modeled concatenative morphology only, assuming that the orthographic form of a word can be split neatly into stem and affixes—a simplifying asssumption that is convenient but often not entirely appropriate (Kay, 1987) (how should one segment English `stopping`, `hoping`, or `knives`?).

In **concatenative** work, Harris (1955) finds morpheme boundaries and segments words accordingly, an approach that was later refined by Hafer and Weiss (1974), Déjean (1998), and many others. The unsupervised segmentation task is tackled in the annual Morpho Challenge (Kurimo et al., 2010), where ParaMor (Monson et al., 2007) and Morfessor (Creutz and Lagus, 2005) are influential contenders. The Bayesian methods that Goldwater et al. (2006b, et seq.) use to segment between words might also be applied to segment within words, but have no notion of paradigms. Goldsmith (2001) finds what he calls *signatures*—sets of affixes that are used with a given set of stems, for example (*NULL*, `-er`, `-ing`, `-s`). Chan (2006) learns sets of morphologically related words; he calls these sets *paradigms* but notes that they are not substructured entities, in contrast to the paradigms we model in this paper. His models are restricted to concatenative and regular morphology.

Morphology discovery approaches that handle **nonconcatenative** and irregular phenomena are more closely related to our work; they are rarer. Yarowsky and Wicentowski (2000) identify inflection-root pairs in large corpora without supervision. Using similarity as well as distributional clues, they identify even irregular pairs like `take`/`took`. Schone and Jurafsky (2001) and Baroni et al. (2002) extract whole conflation sets, like "`abuse`, `abused`, `abuses`, `abusive`, `abusively`, ...," which may also be irregular. We advance this work by not only extracting pairs or sets of related observed words, but whole structured inflectional paradigms, in which we can also predict forms that have never been observed. On the other hand, our present model does not yet use contextual information; we regard this as a future opportunity (see Appendix G). Naradowsky and Goldwater (2009) add simple spelling rules to the Bayesian model of (Goldwater et al., 2006a), enabling it to handle some systematically nonconcatenative cases. Our finite-state transducers can handle more diverse morphological phenomena.

10 Conclusions and Future Work

We have formulated a principled framework for simultaneously obtaining morphological annotation, an unbounded morphological lexicon that fills complete structured morphological paradigms with observed and predicted words, and parameters of a nonconcatenative generative morphology model.

We ran our sampler over a large corpus (10 million words), inferring everything jointly and reducing the prediction error for morphological inflections by up to 10%. We observed that adding a corpus increases the absolute prediction accuracy on frequently occurring morphological forms by up to almost 8%. Future extensions to the model could leverage token context for further improvements (Appendix G).

We believe that a major goal of our field should be to build full-scale explanatory probabilistic models of language. While we focus here on inflectional morphology and evaluate the results in isolation, we regard the present work as a significant step toward a larger generative model under which Bayesian inference would reconstruct other relationships as well (e.g., inflectional, derivational, and evolutionary) among the words in a family of languages.

References

A. C. Albright. 2002. *The Identification of Bases in Morphological Paradigms*. Ph.D. thesis, University of California, Los Angeles.

D. Aldous. 1985. Exchangeability and related topics. *École d'été de probabilités de Saint-Flour XIII*, pages 1–198.

C. E. Antoniak. 1974. Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems. *Annals of Statistics*, 2(6):1152–1174.

R. H Baayen, R. Piepenbrock, and L. Gulikers. 1995. The CELEX lexical database (release 2)[cd-rom]. *Philadelphia, PA: Linguistic Data Consortium, University of Pennsylvania [Distributor]*.

M. Baroni, J. Matiasek, and H. Trost. 2002. Unsupervised discovery of morphologically related words based on orthographic and semantic similarity. In *Proc. of the ACL-02 Workshop on Morphological and Phonological Learning*, pages 48–57.

M. Baroni, S. Bernardini, A. Ferraresi, and E. Zanchetta. 2009. The WaCky Wide Web: A collection of very large linguistically processed web-crawled corpora. *Language Resources and Evaluation*, 43(3):209–226.

David Blackwell and James B. MacQueen. 1973. Ferguson distributions via Pòlya urn schemes. *The Annals of Statistics*, 1(2):353–355, March.

David M. Blei and Peter I. Frazier. 2010. Distance-dependent Chinese restaurant processes. In *Proc. of ICML*, pages 87–94.

E. Chan. 2006. Learning probabilistic paradigms for morphology in a latent class model. In *Proceedings of the Eighth Meeting of the ACL Special Interest Group on Computational Phonology at HLT-NAACL*, pages 69–78.

M. Creutz and K. Lagus. 2005. Unsupervised morpheme segmentation and morphology induction from text corpora using Morfessor 1.0. *Computer and Information Science, Report A*, 81.

H. Déjean. 1998. Morphemes as necessary concept for structures discovery from untagged corpora. In *Proc. of the Joint Conferences on New Methods in Language Processing and Computational Natural Language Learning*, pages 295–298.

Markus Dreyer and Jason Eisner. 2009. Graphical models over multiple strings. In *Proc. of EMNLP*, Singapore, August.

Markus Dreyer, Jason Smith, and Jason Eisner. 2008. Latent-variable modeling of string transductions with finite-state methods. In *Proc. of EMNLP*, Honolulu, Hawaii, October.

Markus Dreyer. 2011. *A Non-Parametric Model for the Discovery of Inflectional Paradigms from Plain Text Using Graphical Models over Strings*. Ph.D. thesis, Johns Hopkins University.

T.S. Ferguson. 1973. A Bayesian analysis of some nonparametric problems. *The annals of statistics*, 1(2):209–230.

Y. Freund and R. Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296.

J. Goldsmith. 2001. Unsupervised learning of the morphology of a natural language. *Computational Linguistics*, 27(2):153–198.

S. Goldwater, T. Griffiths, and M. Johnson. 2006a. Interpolating between types and tokens by estimating power-law generators. In *Proc. of NIPS*, volume 18, pages 459–466.

S. Goldwater, T. L. Griffiths, and M. Johnson. 2006b. Contextual dependencies in unsupervised word segmentation. In *Proc. of COLING-ACL*.

P.J. Green. 1995. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82(4):711.

M. A Hafer and S. F Weiss. 1974. Word segmentation by letter successor varieties. *Information Storage and Retrieval*, 10:371–385.

Z. S. Harris. 1955. From phoneme to morpheme. *Language*, 31(2):190–222.

G.E. Hinton. 2002. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800.

Klara Janecki. 2000. *300 Polish Verbs*. Barron's Educational Series.

E. T. Jaynes. 2003. *Probability Theory: The Logic of Science*. Cambridge Univ Press. Edited by Larry Bretthorst.

D. Jurafsky, J. H. Martin, A. Kehler, K. Vander Linden, and N. Ward. 2000. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. MIT Press.

M. Kay. 1987. Nonconcatenative finite-state morphology. In *Proc. of EACL*, pages 2–10.

M. Kurimo, S. Virpioja, V. Turunen, and K. Lagus. 2010. Morpho Challenge competition 2005–2010: Evaluations and results. In *Proc. of ACL SIGMORPHON*, pages 87–95.

P. H. Matthews. 1972. *Inflectional Morphology: A Theoretical Study Based on Aspects of Latin Verb Conjugation*. Cambridge University Press.

Christian Monson, Jaime Carbonell, Alon Lavie, and Lori Levin. 2007. ParaMor: Minimally supervised induction of paradigm structure and morphological analysis. In *Proc. of ACL SIGMORPHON*, pages 117–125, June.

J. Naradowsky and S. Goldwater. 2009. Improving morphology induction by learning spelling rules. In *Proc. of IJCAI*, pages 1531–1536.

J. Pitman and M. Yor. 1997. The two-parameter Poisson-Dirichlet distribution derived from a stable subordinator. *Annals of Probability*, 25:855–900.

P. Schone and D. Jurafsky. 2001. Knowledge-free induction of inflectional morphologies. In *Proc. of NAACL*, volume 183, pages 183–191.

J. Sethuraman. 1994. A constructive definition of Dirichlet priors. *Statistica Sinica*, 4(2):639–650.

N. A. Smith, D. A. Smith, and R. W. Tromble. 2005. Context-based morphological disambiguation with random fields. In *Proceedings of HLT-EMNLP*, pages 475–482, October.

G. T. Stump. 2001. *Inflectional Morphology: A Theory of Paradigm Structure*. Cambridge University Press.

Y.W. Teh, M.I. Jordan, M.J. Beal, and D.M. Blei. 2006. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581.

Yee Whye Teh. 2006. A hierarchical Bayesian language model based on Pitman-Yor processes. In *Proc. of ACL*.

K. Toutanova and R.C. Moore. 2002. Pronunciation modeling for improved spelling correction. In *Proc. of ACL*, pages 144–151.

D. Yarowsky and R. Wicentowski. 2000. Minimally supervised morphological analysis by multimodal alignment. In *Proc. of ACL*, pages 207–216, October.

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.

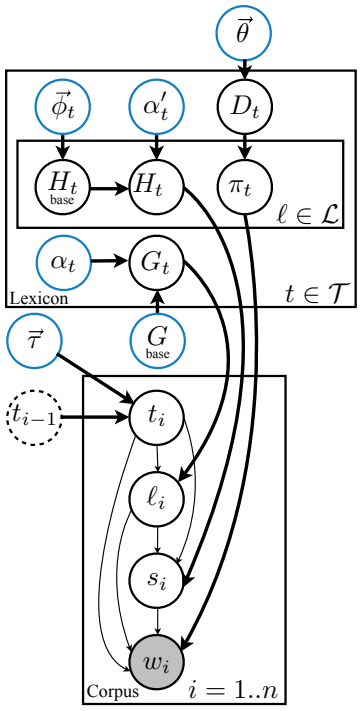

Figure 3: A graphical model drawing of the generative model. As in Fig. 2, type variables are above and token variables are below. Grammar variables are in blue. Although circles denote random variables, we label them with values (such as ℓ_i) to avoid introducing new notation (such as L_i) for the random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

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 this 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.

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).

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

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

[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

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

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 $\alpha'_t \sum_{s \in \mathcal{S}_t} (c_s - c H_t(s)) \vec{f}(s)$, 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.

[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.

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

Form	50 seed paradigms			100 seed paradigms		
	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, $p < 0.05$).

Form	50 seed paradigms			100 seed paradigms		
	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.

be 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, even for irregulars (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 errors 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 ran 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 features. 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 non-empty 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.