Bayesian Modeling of Lexical Resources for Low-Resource Settings

Nicholas Andrews and **Mark Dredze** and **Benjamin Van Durme** and **Jason Eisner**
Department of Computer Science and Human Language Technology Center of Excellence
Johns Hopkins University
3400 N. Charles St., Baltimore, MD 21218 USA
{noa,mdredze,vandurme,eisner}@jhu.edu

Abstract

Lexical resources such as dictionaries and gazetteers are often used as auxiliary data for tasks such as part-of-speech induction and named-entity recognition. However, discriminative training with lexical features requires annotated data to reliably estimate the lexical feature weights and may result in overfitting the lexical features at the expense of features which generalize better. In this paper, we investigate a more robust approach: we stipulate that the lexicon is the result of an assumed generative process. Practically, this means that we may treat the lexical resources as *observations* under the proposed generative model. The lexical resources provide training data for the generative model without requiring separate data to estimate lexical feature weights. We evaluate the proposed approach in two settings: part-of-speech induction and low-resource named-entity recognition.

1 Introduction

Dictionaries and gazetteers are useful in many natural language processing tasks. These lexical resources may be derived from freely available sources (such as Wikidata and Wiktionary) or constructed for a particular domain. Lexical resources are typically used to complement existing annotations for a given task (Ando and Zhang, 2005; Collobert et al., 2011). In this paper, we focus instead on low-resource settings where task annotations are unavailable or scarce. Specifically, we use lexical resources to guide part-of-speech induction (§4) and to bootstrap named-entity recognizers in low-resource languages (§5).

Given their success, it is perhaps surprising that incorporating gazetteers or dictionaries into discriminative models (e.g. conditional random fields) may sometimes *hurt performance*. This phenomena is called *weight under-training*, in which lexical features—which detect whether a name is listed in the dictionary or gazetteer—are given excessive weight at the expense of other useful features such as spelling features that would generalize to unlisted names (Smith et al., 2005; Sutton et al., 2006; Smith and Osborne, 2006). Furthermore, discriminative training with lexical features requires sufficient annotated training data, which poses challenges for the unsupervised and low-resource settings we consider here.

Our observation is that Bayesian modeling provides a principled solution. The lexicon is itself a dataset that was generated by some process. Practically, this means that lexicon entries (words or phrases) may be treated as additional observations. As a result, these entries provide information about how names are spelled. The presence of the lexicon therefore now *improves* training of the spelling features, rather than *competing with* the spelling features to help explain the labeled corpus.

A downside is that generative models are typically less feature-rich than their globally normalized discriminative counterparts (e.g. conditional random fields). In designing our approach—the *hierarchical sequence memoizer (HSM)*—we aim to be reasonably expressive while retaining practically useful inference algorithms. We propose a Bayesian nonparametric model to serve as a generative distribution responsible for both lexicon and corpus data. The proposed model *memoizes* previously used lexical entries (words or phrases) but backs off to a character-level distribution when generating novel types (Teh, 2006; Mochihashi et al., 2009). We propose an efficient inference algorithm for the proposed model using particle Gibbs sampling (§3). Our code is available at `https://github.com/noa/bayesner`.

2 Model

Our goal is to fit a model that can automatically annotate text. We observe a supervised or unsupervised training corpus. For each label y in the annotation scheme, we also observe a lexicon of strings of type y. For example, in our tagging task (§4), a dictionary provides us with a list of words for each part-of-speech tag y. (These lists need not be disjoint.) For named-entity recognition (NER, §5), we use a list of words or phrases for each named-entity type y (PER, LOC, ORG, etc.).[1]

2.1 Modeling the lexicon

We may treat the lexicon for type y, of size m_y, as having been produced by a set of m_y IID draws from an unknown distribution P_y over the words or named entities of type y. It therefore provides some evidence about P_y. We will later assume that P_y is also used when generating mentions of these words or entities in text. Thanks to this sharing of P_y, if $x = \texttt{Washington}$ is listed in the gazetteer of locations ($y = \text{LOC}$), we can draw the same conclusions as if we had seen a LOC-labeled instance of `Washington` in a supervised corpus.

Generalizing this a bit, we may suppose that one observation of string x in the lexicon is equivalent to c labeled tokens of x in a corpus, where the constant $c > 0$ is known as a *pseudocount*. In other words, observing a lexicon of m_y distinct types $\{x_1, \ldots, x_{m_y}\}$ is equivalent to observing a labeled *pseudocorpus* of cm_y tokens. Notice that given such an observation, the prior probability of any candidate distribution P_y is reweighted by the likelihood $\frac{(cm_y)!}{(c!)^{m_y}} \cdot (P_y(x_1)P_y(x_2)\cdots P_y(x_{m_y}))^c$. Therefore, this choice of P_y can have relatively high posterior probability only to the extent that it assigns high probability to all of the lexicon types.

2.2 Discussion

We employ the above model because it has reasonable qualitative behavior and because computationally, it allows us to condition on observed lexicons as easily as we condition on observed corpora. However, we caution that as a generative model of the lexicon, it is deficient, in the sense that it allocates probability mass to events that cannot actually correspond to any lexicon. After all, drawing cm_y IID tokens from P_y is highly unlikely to result in exactly c tokens of each of m_y different types, and yet a run of our system will always assume that precisely this happened to produce each observed lexicon! To avoid the deficiency, one could assume that the lexicon was generated by rejection sampling: that is, the gazetteer author repeatedly drew samples of size cm_y from P_y until one was obtained that had this property, and then returned the set of distinct types in that sample as the lexicon for y. But this is hardly a realistic description of how gazetteers are actually constructed. Rather, one imagines that the gazetteer author simply harvested a lexicon of frequent types from P_y or from a corpus of tokens generated from P_y. For example, a much better generative story is that the lexicon was constructed as the first m_y distinct types to appear $\geq c$ times in an unbounded sequence of IID draws from P_y. When $c = 1$, this is equivalent to modeling the lexicon as m_y draws *without replacement* from P_y.[2] Unfortunately, draws without replacement are no longer IID or exchangeable: order matters. It would therefore become difficult to condition inference and learning on an observed lexicon, because we would need to explicitly sum or sample over the possibilities for the latent *sequence* of tokens (or stick segments). We therefore adopt the simpler deficient model.

A version of our lexicon model (with $c = 1$) was previously used by Dreyer and Eisner (2011, Appendix C), who observed a list of verb paradigm types rather than word or entity-name types.

2.3 Prior distribution over P_y

We assume *a priori* that P_y was drawn from a Pitman-Yor process (PYP) (Pitman and Yor, 1997). Both the lexicon and the ordinary corpus are observations that provide information about P_y. The PYP is defined by three parameters: a concentration parameter α, a discount parameter d, and a base distribution H_y. In our case, H_y is a distribution over $\mathcal{X} = \Sigma^*$, the set of possible strings over a finite character alphabet Σ.

For example, H_{LOC} is used to choose new place names, so it describes what place names tend to

[1]Dictionaries and knowledge bases provide more information than we use in this paper. For instance, Wikidata also provides a wealth of attributes and other metadata for each entity s. In principle, this additional information could also be helpful in estimating $P_y(s)$; we leave this intriguing possibility for future work.

[2]If we assume that P_y was drawn from a Pitman-Yor process prior (as in §2.3) using the stick-breaking method (Pitman, 1996), it is also equivalent to modeling the lexicon as the set of labels of the *first* m_y stick segments (which tend to have high probability).

look like in the language. The draw $P_{\text{LOC}} \sim \text{PYP}(d, \alpha, H_{\text{LOC}})$ is an "adapted" version of H_{LOC}. It is P_{LOC} that determines how often each name is mentioned in text (and whether it is mentioned in the lexicon). Some names such as `Washington` that are merely plausible under H_{LOC} are far more frequent under P_{LOC}, presumably because they were chosen as the names of actual, significant places. These place names were randomly drawn from H_{LOC} as part of the procedure for drawing P_y.

The expected value of P_y is H (i.e., H is the mean of the PYP distribution), but if α and d are small, then a typical draw of P_y will be rather different from H, with much of the probability mass falling on a subset of the strings.

At training or test time, when deciding whether to label a corpus token of $x =$ `Washington` as a place or person, we will be interested in the relative values of $P_{\text{LOC}}(x)$ and $P_{\text{PER}}(x)$. In practice, we do not have to represent the unknown infinite object P_y, but can integrate over its possible values. When $P_y \sim \text{PYP}(d, \alpha, H_y)$, then a sequence of draws $X_1, X_2, \ldots \sim P_y$ is distributed according to a Chinese restaurant process, via

$$\begin{aligned} &P_y(X_{i+1} = x \mid X_1, \ldots, X_i) \quad (1)\\ &\quad = \frac{\text{customers}(x) - d \cdot \text{tables}(x)}{\alpha + i} \\ &\quad + \frac{\alpha + d \cdot \sum_{x'} \text{tables}(x')}{\alpha + i} H_y(x) \end{aligned}$$

where $\text{customers}(x) \leq i$ is the number of times that x appeared among $X_1, \ldots, X_i$, and $\text{tables}(x) \leq \text{customers}(x)$ is the number of those times that x was drawn from H_y (where each $P_y(X_i \mid \cdots)$ defined by (1) is interpreted as a mixture distribution that *sometimes* uses H_y).

2.4 Form of the base distribution H_y

By fitting H_y on corpus and lexicon data, we learn what place names or noun strings tend to look like in the language. By simultaneously fitting P_y, we learn which ones are commonly mentioned. Recall that under our model, tokens are drawn from P_y but the underlying types are drawn from H_y, e.g., H_y is responsible for (at least) the first token of each type.

A simple choice for H_y is a Markov process that emits characters in $\Sigma \cup \{\$\}$, where \$ is a distinguished stop symbol that indicates the end of the string. Thus, the probability of producing \$ controls the typical string length under H_y.

We use a more sophisticated model of strings—a sequence memoizer (SM), which is a (hierarchical) Bayesian treatment of variable-order Markov modeling (Wood et al., 2009). The SM allows dependence on an unbounded history, and the probability of a given sequence (string) can be found efficiently much as in equation (1).

Given a string $x = a_1 \cdots a_J \in \Sigma^*$, the SM assigns a probability to it via

$$\begin{aligned} H_y(\boldsymbol{a}_{1:J}) &= \Big(\prod_{j=1}^{J} H_y(a_j \mid \boldsymbol{a}_{1:j-1}) \Big) H_y(\$ \mid \boldsymbol{a}_{1:J}) \\ &= \Big(\prod_{j=1}^{J} H_{y,\boldsymbol{a}_{1:j-1}}(a_j) \Big) H_{y,\boldsymbol{a}_{1:J}}(\$) \quad (2) \end{aligned}$$

where $H_{y,\boldsymbol{u}}(a)$ denotes the conditional probability of character a given the left context $\boldsymbol{u} \in \Sigma^*$. Each $H_{y,\mathbf{u}}$ is a distribution over Σ, defined recursively as

$$\begin{aligned} H_{y,\epsilon} &\sim \text{PYP}(d_\epsilon, \alpha_\epsilon, \mathcal{U}_\Sigma) \quad (3) \\ H_{y,\boldsymbol{u}} &\sim \text{PYP}(d_{|\mathbf{u}|}, \alpha_{|\mathbf{u}|}, H_{y,\sigma(\mathbf{u})}) \end{aligned}$$

where ϵ is the empty sequence, $\mathcal{U}_\Sigma$ is the uniform distribution over $\Sigma \cup \{\$\}$, and $\sigma(\mathbf{u})$ drops the first symbol from $\mathbf{u}$. The discount and concentration parameters $(d_{|\mathbf{u}|}, \alpha_{|\mathbf{u}|})$ are associated with the lengths of the contexts $|\boldsymbol{u}|$, and should generally be larger for longer (more specific) contexts, implying stronger backoff from those contexts.[3]

Our inference procedure is largely indifferent to the form of H_y, so the SM is not the only option. It would be possible to inject more assumptions into H_y, for instance via structured priors for morphology or a grammar of name structure. Another possibility is to use a parametric model such as a neural language model (e.g., Jozefowicz et al. (2016)), although this would require an inner-loop of gradient optimization.

2.5 Modeling the sequence of tags y

We now turn to modeling the corpus. We assume that each sentence is generated via a sequence of latent labels $\boldsymbol{y} = \boldsymbol{y}_{1:T} \in \mathcal{Y}^*$.[4] The observations

[3]We fix these hyperparameters using the values suggested in (Wood et al., 2009; Gasthaus and Teh, 2010), which we find to be quite robust in practice. One could also resample their values (Blunsom and Cohn, 2010); we experimented with this but did not observe any consistent advantage to doing so in our setting.

[4]The label sequence is terminated by a distinguished end-of-sequence label, again written as \$.

$\boldsymbol{x}_{1:T}$ are then generated conditioned on the label sequence via the corresponding P_y distribution (defined in §2.3). All observations with the *same* label y are drawn from the same P_y, and thus this *subsequence* of observations is distributed according to the Chinese restaurant process (1).

We model $\boldsymbol{y}$ using another sequence memoizer model. This is similar to other hierarchical Bayesian models of latent sequences (Goldwater and Griffiths, 2007; Blunsom and Cohn, 2010), but again, it does not limit the Markov order (the number of preceding labels that are conditioned on). Thus, the probability of a sequence of latent types is computed in the same way as the base distribution in §2.4, that is,

$$p(\boldsymbol{y}_{1:T}) := \Big(\prod_{t=1}^{T} G_{\boldsymbol{y}_{1:t-1}}(y_t)\Big) G_{\boldsymbol{y}_{1:T}}(\$) \qquad (4)$$

where $G_{\boldsymbol{v}}(y)$ denotes the conditional probability of latent label $y \in \mathcal{Y}$ given the left context $\boldsymbol{v} \in \mathcal{Y}^*$. Each $G_{\mathbf{v}}$ is a distribution over $\mathcal{Y}$, defined recursively as

$$\begin{aligned} G_\epsilon &\sim \mathrm{PYP}(d_\epsilon, \alpha_\epsilon, \mathcal{U}_{\mathcal{Y}}) \\ G_{\boldsymbol{v}} &\sim \mathrm{PYP}(d_{|\mathbf{v}|}, \alpha_{|\mathbf{v}|}, G_{\sigma(\mathbf{v})}) \end{aligned} \qquad (5)$$

The probability of transitioning to label y_t depends on the assignments of all previous labels $y_1 \cdots y_{t-1}$.

For part-of-speech induction, each label y_t is the part-of-speech associated with the corresponding word x_t. For named-entity recognition, we say that each word token is labeled with a named entity type (LOC, PER, ...),[5] or with *itself* if it is not a named entity but rather a "context word." For example, the word token x_t = `Washington` could have been emitted from the label y_t = LOC, or from y_t = PER, or from y_t = `Washington` itself (in which case $p(x_t \mid y_t) = 1$). This uses a much larger set of labels $\mathcal{Y}$ than in the traditional setup where all context words are emitted from the same latent label type O. Of course, most labels are impossible at most positions (e.g., y_t cannot be `Washington` *unless* x_t = `Washington`). This scheme makes our generative model sensitive to specific contexts (which is accomplished in discriminative NER systems by contextual features). For example, the SM for $\boldsymbol{y}$ can learn that `spoke to PER yesterday` is a common 4-gram in the label sequence $\boldsymbol{y}$, and thus we are more likely to label `Washington` as a person if $\boldsymbol{x}$ = `...spoke to Washington yesterday...`.

We need one change to make this work, since now $\mathcal{Y}$ must include not only the standard NER labels $\mathcal{Y}' = \{\text{PER}, \text{LOC}, \text{ORG}, \text{GPE}\}$ but also words like `Washington`. Indeed, now $\mathcal{Y} = \mathcal{Y}' \cup \Sigma^*$. But no uniform distribution exists over the infinite set Σ^*, so how should we replace the base distribution $\mathcal{U}_{\mathcal{Y}}$ over labels in equation (5)? Answer: To draw from the new base distribution, sample $y \sim \mathcal{U}_{\mathcal{Y}' \cup \{\text{CONTEXT}\}}$. If y = CONTEXT, however, then "expand" it by resampling $y \sim H_{\text{CONTEXT}}$. Here H_{CONTEXT} is the base distribution over spellings of context words, and is learned just like the other H_y distributions in §2.4.

3 Inference via particle Markov chain Monte Carlo

3.1 Sequential sampler

Taking $\boldsymbol{Y}$ to be a random variable, we are interested in the posterior distribution $p(\boldsymbol{Y} = \boldsymbol{y} \mid \boldsymbol{x})$ over label sequences $\boldsymbol{y}$ given the emitted word sequence $\boldsymbol{x}$. Our model does not admit an efficient dynamic programming algorithm, owing to the dependencies introduced among the $\boldsymbol{Y}_t$ when we marginalize over the unknown G and P distributions that govern transitions and emissions, respectively. In contrast to tagging with a hidden Markov model tagging, the distribution of each label Y_t depends on *all* previous labels $\boldsymbol{y}_{1:t-1}$, for two reasons: ① The transition distribution $p(Y_t = y \mid \boldsymbol{y}_{1:t-1})$ has unbounded dependence because of the PYP prior (4). ② The emission distribution $p(x_t \mid Y_t = y)$ depends on the emissions observed from any earlier tokens of y, because of the Chinese restaurant process (1). When ② is the only complication, block Metropolis-Hastings samplers have proven effective (Johnson et al., 2007). However, this approach uses dynamic programming to sample from a proposal distribution efficiently, which ① precludes in our case. Instead, we use sequential Monte Carlo (SMC)—sometimes called *particle filtering*—as a proposal distribution. Particle filtering is typically used in online settings, including word segmentation (Börschinger and Johnson, 2011), to make decisions before all of $\boldsymbol{x}$ has been observed. However, we are interested in the inference (or *smoothing*) problem that conditions on all of $\boldsymbol{x}$ (Dubbin and Blunsom, 2012; Tripuraneni et al., 2015).

SMC employs a *proposal distribution* $q(\boldsymbol{y} \mid \boldsymbol{x})$

[5] In §3.2, we will generalize this labeling scheme to allow multi-word named entities such as `New York`.

whose definition decomposes as follows:

$$q(y_1 \mid x_1) \prod_{t=2}^{T} q(y_t \mid \boldsymbol{y}_{1:t-1}, \boldsymbol{x}_{1:t}) \tag{6}$$

for $T = |\boldsymbol{x}|$. To sample a sequence of latent labels, first sample an initial label y_1 from q_1, then proceed incrementally by sampling y_t from $q_t(\cdot \mid \boldsymbol{y}_{1:t-1}, \boldsymbol{x}_{1:t})$ for $t = 2, \ldots, T$. The final sampled sequence $\boldsymbol{y}$ is called a *particle*, and is given an *unnormalized importance weight* of $\tilde{w} = \tilde{w}_T \cdot p(\$ \mid \boldsymbol{y}_{1:T})$ where $\tilde{w}_T$ was built up via

$$\tilde{w}_t := \tilde{w}_{t-1} \cdot \frac{p(\boldsymbol{y}_{1:t}, \boldsymbol{x}_{1:t})}{p(\boldsymbol{y}_{1:t-1}, \boldsymbol{x}_{1:t-1})\, q(y_t \mid \boldsymbol{y}_{1:t-1}, \boldsymbol{x}_{1:t})} \tag{7}$$

The SMC procedure consists of generating a system of M weighted particles whose unnormalized importance weights $\tilde{w}^{(m)} : 1 \leq m \leq M$ are normalized into $w^{(m)} := \tilde{w}^{(m)} / \sum_{m=1}^{M} \tilde{w}^{(m)}$. As $M \to \infty$, SMC provides a consistent estimate of the marginal likelihood $p(\boldsymbol{x})$ as $\frac{1}{M} \sum_{m=1}^{M} \tilde{w}^{(m)}$, and samples from the weighted particle system are distributed as samples from the desired posterior $p(\boldsymbol{y} \mid \boldsymbol{x})$ (Doucet and Johansen, 2009).

Particle Gibbs. We employ SMC as a kernel in an MCMC sampler (Andrieu et al., 2010). In particular, we use a block Gibbs sampler in which we iteratively resample the hidden labeling $\boldsymbol{y}$ of a sentence $\boldsymbol{x}$ conditioned on the current labelings for all other sentences in the corpus. In this context, the algorithm is called *conditional* SMC since one particle is always fixed to the previous sampler state for the sentence being resampled, which ensures that the MCMC procedure is ergodic. At a high level, this procedure is analogous to other Gibbs samplers (e.g. for topic models), except that the conditional SMC (CSMC) kernel uses auxiliary variables (particles) in order to generate the new block variable assignments. The procedure is outlined in Algorithm 1. Given a previous latent state assignment $\boldsymbol{y}'_{1:T}$ and observations $\boldsymbol{x}_{1:T}$, the CSMC kernel produces a new latent state assignment via M auxiliary particles where one particle is fixed to the previous assignment. For ergodicity, $M \geq 2$, where larger values of M may improve mixing rate at the expense of increased computation per step.

Proposal distribution. The choice of proposal distribution q is crucial to the performance of SMC methods. In the case of continuous latent variables, it is common to propose y_t from the transition probability $p(Y_t \mid \boldsymbol{y}_{1:t-1})$ because this distribution usually has a simple form that permits efficient sampling. However, it is possible to do better in the case of discrete latent variables. The optimal proposal distribution is the one which minimizes the variance of the importance weights, and is given by

$$\begin{aligned} q(y_t \mid \boldsymbol{y}_{1:t-1}, \boldsymbol{x}_{1:t}) &:= p(y_t \mid \boldsymbol{y}_{1:t-1}, \boldsymbol{x}_{1:t}) \\ &= \frac{p(y_t \mid \boldsymbol{y}_{1:t-1}) p(x_t \mid y_t)}{p(x_t \mid \boldsymbol{y}_{1:t-1})} \end{aligned} \tag{8}$$

where

$$p(x_t \mid \boldsymbol{y}_{1:t-1}) = \sum_{y_t \in \mathcal{Y}} p(y_t \mid \boldsymbol{y}_{1:t-1}) p(x_t \mid y_t) \tag{9}$$

Substituting this expression in equation (7) and simplifying yields the incremental weight update:

$$\tilde{w}_t := \tilde{w}_{t-1} \cdot p(x_t \mid \boldsymbol{y}_{1:t-1}) \tag{10}$$

Resampling. In filtering applications, it is common to use resampling operations to prevent weight degeneracy. We do not find resampling necessary here for three reasons. First, note that we resample hidden label sequences that are only as long as the number of words in a given sentence. Second, we use a proposal which minimizes the variance of the weights. Finally, we use SMC as a kernel embedded in an MCMC sampler; asymptotically, this procedure yields samples from the desired posterior regardless of degeneracy (which only affects the mixing rate). Practically speaking, one can diagnose the need for resampling via the effective sample size (ESS) of the particle system:

$$\text{ESS} := \frac{1}{\sum_{m=1}^{M} (\tilde{w}^{(m)})^2} = \frac{(\sum_{m=1}^{M} w^{(m)})^2}{\sum_{m=1}^{M} (w^{(m)})^2}$$

In our experiments, we find that ESS remains high (a significant fraction of M) even for long sentences, suggesting that resampling is not necessary to enable mixing of the the Gibbs sampler.

Decoding. In order to obtain a single latent variable assignment for evaluation purposes, we simply take the state of the Markov chain after a fixed number of iterations of particle Gibbs. In principle, one could collect many samples during particle Gibbs and use them to perform minimum Bayes risk decoding under a given loss function. However, this approach is somewhat slower and did not appear to improve performance in preliminary experiments

Algorithm 1 Conditional SMC

```
1:  procedure CSMC($\boldsymbol{x}_{1:T}$, $\boldsymbol{y}'_{1:T}$, $M$)
2:      Draw $y_1^{(m)}$ (eqn. 8) for $m \in [1, M-1]$
3:      Set $y_1^{(M)} = y'_1$
4:      Set $\tilde{w}_1^{(m)}$ (eqn. 10) for $m \in [1, M]$
5:      for $t = 2$ to $T$ do
6:          Draw $y_t^{(m)}$ (eqn. 8) for $m \in [1, M-1]$
7:          Set $y_t^M = y'_t$
8:          Set $\tilde{w}_t^{(m)}$ (eqn. 10) for $m \in [1, M]$
9:      Set $\tilde{w}^{(m)} = \tilde{w}_T^{(m)} p(\$|\boldsymbol{y}_{1:T})$ for $m \in [1, M]$
10:     Draw index $k$ where $p(k = m) \propto \tilde{w}^{(m)}$
11:     return $\boldsymbol{y}_{1:T}^{(k)}$
```

3.2 Segmental sampler

We now present an sampler for settings such as NER where each latent label emits a *segment* consisting of 1 or more words. We make use of the same transition distribution $p(y_t \mid \boldsymbol{y}_{1:t-1})$, which determines the probability of a label in a given context, and an emission distribution $p(x_t \mid y_t)$ (namely P_{y_t}); these are assumed to be drawn from hierarchical Pitman-Yor processes described in §2.5 and §2.1, respectively. To allow the x_t to be a multi-word string, we simply augment the character set with a distinguished space symbol $\llcorner\!\lrcorner \in \Sigma$ that separates words within a string. For instance, New York would be generated as the 9-symbol sequence New␣York\$.

Although the *model* emits New␣York all at once, we still formulate our *inference procedure* as a particle filter that proposes one tag for each word. Thus, for a given segment label type y, we allow two tag types for its words:

- I-y corresponds to a non-final word in a segment of type y (in effect, a word with a following ␣ attached).
- E-y corresponds to the final word in a segment of type y.

For instance, $\boldsymbol{x}_{1:2}$ = New York would be annotated as a location segment by defining $\boldsymbol{y}_{1:2}$ = I-Loc E-Loc. This says that $\boldsymbol{y}_{1:2}$ has *jointly* emitted $\boldsymbol{x}_{1:2}$, an event with prior probability P_{LOC}(New␣York). Each word that is not part of a named entity is considered to be a single-word segment. For example, if the next word were x_3 = hosted then it should be tagged with y_3 = hosted as in §2.5, in which case x_3 was emitted with probability 1.

To adapt the sampler described in §3.1 for the segmental case, we need only to define the transition and emission probabilities used in equation (8) and its denominator (9).

For the transition probabilities, we want to model the sequence of segment labels. If y_{t-1} is an I- tag, we take $p(y_t \mid \boldsymbol{y}_{1:t-1}) = 1$, since then y_t merely continues an existing segment. Otherwise y_t starts a new segment, and we take $p(y_t \mid \boldsymbol{y}_{1:t-1}) = 1$ to be defined by the PYP's probability $G_{\boldsymbol{y}_{1:t-1}}(y_t)$ as usual, but where we interpret the subscript $\boldsymbol{y}_{1:t-1}$ to refer to the possibly shorter sequence of segment labels implied by those $t-1$ tags.

For the emission probabilities, if y_t has the form I-y or E-y, then its associated emission probability no longer has the form $p(x_t \mid y_t)$, since the choice of x_t also depends on any words emitted earlier in the segment. Let $s \le t$ be the starting position of the segment that contains t. If y_t = E-y, then the emission probability is proportional to $P_y(x_s \llcorner\!\lrcorner x_{s+1} \llcorner\!\lrcorner \ldots \llcorner\!\lrcorner x_t)$. If y_t = I-y then the emission probability is proportional to the *prefix probability* $\sum_x P_y(x)$ where x ranges over all strings in Σ^* that have $x_s \llcorner\!\lrcorner x_{s+1} \llcorner\!\lrcorner \ldots \llcorner\!\lrcorner x_t \llcorner\!\lrcorner$ as a proper prefix. Prefix probabilities in H_y are easy to compute because H_y has the form of a language model, and prefix probabilities in P_y are therefore also easy to compute (using a prefix tree for efficiency).

This concludes the description of the segmental sampler. Note that the particle Gibbs procedure is unchanged.

4 Inducing parts-of-speech with type-level supervision

Automatically inducing parts-of-speech from raw text is a challenging problem (Goldwater et al., 2005). Our focus here is on the easier problem of type-supervised part-of-speech induction, in which (partial) dictionaries are used to guide inference (Garrette and Baldridge, 2012; Li et al., 2012). Conditioned on the unlabeled corpus and dictionary, we use the MCMC procedure described in §3.1 to impute the latent parts-of-speech.

Since dictionaries are freely available for hundreds of languages,[6] we see this as a mild additional requirement in practice over the purely unsupervised setting.

In prior work, dictionaries have been used as constraints on possible parts-of-speech: words appearing in the dictionary take one of their known parts-

[6] https://www.wiktionary.org/

of-speech. In our setting, however, the dictionaries are not constraints but evidence. If `monthly` is listed in (only) the adjective lexicon, this tells us that P_{ADJ} sometimes generates `monthly` and therefore that H_{ADJ} may also tend to generate other words that end with `-ly`. However, for us, $P_{\text{ADV}}(\texttt{monthly}) > 0$ as well, allowing us to still correctly treat `monthly` as a possible adverb if we later encounter it in a training or test corpus.

4.1 Experiments

We follow the experimental procedure described in Li et al. (2012), and use their released code and data to compare to their best model: a second-order maximum entropy Markov model parametrized with log-linear features (SHMM-ME). This model uses hand-crafted features designed to distinguish between different parts-of-speech, and it has special handling for rare words. This approach is surprisingly effective and outperforms alternate approaches such as cross-lingual transfer (Das and Petrov, 2011). However, it also has limitations, since words that do not appear in the dictionary will be unconstrained, and spurious or incorrect lexical entries may lead to propagation of errors.

The lexicons are taken from the Wiktionary project; their size and coverage are documented by (Li et al., 2012). We evaluate our model on multi-lingual data released as part of the CoNLL 2007 and CoNLL-X shared tasks. In particular, we use the same set of languages as Li et al. (2012).[7] For our method, we impute the parts-of-speech by running particle Gibbs for 100 epochs, where one epoch consists of resampling the states for a each sentence in the corpus. The final sampler state is then taken as a 1-best tagging of the unlabeled data.

Results. The results are reported in Table 1. We find that our hierarchical sequence memoizer (HSM) matches or exceeds the performance of the baseline (SHMM-ME) for nearly all the tested languages, particularly for morphologically rich languages such as German where the spelling distributions H_y may capture regularities. It is interesting to note that our model performs worse relative to the baseline for English; one possible explanation is that the baseline uses hand-engineered features whereas ours does not, and these features may have been tuned using English data for validation.

[7]With the exception of Dutch. Unlike the other CoNLL languages, Dutch includes phrases, and the procedure by which these were split into tokens was not fully documented.

Our generative model is supposed to exploit lexicons well. To see what is lost from using a generative model, we also compared with Li et al. (2012) on standard supervised tagging without any lexicons. Even here our generative model is very competive, losing only on English and Swedish.

5 Boostrapping NER with type-level supervision

Name lists and dictionaries are useful for NER particularly when in-domain annotations are scarce. However, with little annotated data, discriminative training may be unable to reliably estimate lexical feature weights and may overfit. In this section, we are interested in evaluating our proposed Bayesian model in the context of low-resource NER.

5.1 Data

Most languages do not have corpora annotated for parts-of-speech, named-entities, syntactic parses, or other linguistic annotations. Therefore, rapidly deploying natural language technologies in a new language may be challenging. In the context of facilitating relief responses in emergencies such as natural disasters, the DARPA LORELEI (Low Resource Languages for Emergent Incidents) program has sponsored the development and release of representative “language packs” for Turkish and Uzbek with more languages planned (Strassel and Tracey, 2016). We use the named-entity annotations as part of these language packs which include persons, locations, organizations, and geo-political entities, in order to explore bootstrapping named-entity recognition from small amounts of data. We consider two types of data: ① in-context annotations, where sentences are fully annotated for named-entities, and ② lexical resources.

The LORELEI language packs lack adequate in-domain lexical resources for our purposes. Therefore, we simulate in-domain lexical resources by holding out portions of the annotated development data and deriving dictionaries and name lists from them. For each label $y \in \{\text{PER}, \text{LOC}, \text{ORG}, \text{GPE}, \text{CONTEXT}\}$, our lexicon for y lists all distinct y-labeled strings that appear in the held-out data. This setup ensures that the labels associated with lexicon entries correspond to the annotation guidelines used in the data we use for evaluation. It avoids possible problems that might arise when leveraging noisy out-of-domain knowledge bases, which we may explore in future.

	Model	Danish	German	Greek	English	Italian	Portuguese	Spanish	Swedish	**Mean**
Wiktionary	SHMM-ME	83.3	85.8	79.2	87.1	86.5	84.5	86.4	86.1	84.9
	HSM	83.7	90.7	81.7	84.0	86.7	85.5	87.6	86.8	85.8
Supervised	SHMM-ME	93.9	97.4	95.1	95.8	93.8	95.5	93.8	95.5	95.1
	HSM	95.2	97.4	97.4	95.2	94.5	96.0	95.6	92.2	95.3

Table 1: Part-of-speech induction results in multiple languages.

5.2 Evaluation

In this section we report supervised NER experiments on two low-resource languages: Turkish and Uzbek. We vary both the amount of supervision as well as the size of the lexical resources. A challenge when evaluating the performance of a model with small amounts of training data is that there may be high-variance in the results. In order to have more confidence in our results, we perform bootstrap resampling experiments in which the training set, evaluation set, and lexical resources are randomized across several replications of the same experiment (for each of the data conditions). We use 10 replications for each of the data conditions reported in Figures 1–2, and report both the mean performance and 95% confidence intervals.

Baseline. We use the Stanford NER system with a standard set of language-independent features (Finkel et al., 2005).[8]. This model is a conditional random field (CRF) with feature templates which include character n-grams as well as word shape features. Crucially, we also incorporate lexical features. The CRF parameters are regularized using an L1 penalty and optimized via Orthant-wise limited-memory quasi-Newton optimization (Andrew and Gao, 2007). For both our proposed method and the discriminative baseline, we use a fixed set of hyperparameters (i.e. we do not use a separate validation set for tuning each data condition). In order to make a fair comparison to the CRF, we use our sampler for forward inference only, without resampling on the test data.

Results. We show learning curves as a function of supervised training corpus size. Figure 1 shows that our generative model strongly beats the baseline in this low-data regime. In particular, when there is little annotated training data, our proposed generative model can compensate by exploiting the lexicon, while the discriminative baseline scores terribly. The performance gap decreases with larger supervised corpora, which is consistent with prior results comparing generative and discriminative training (Ng and Jordan, 2002).

In Figure 2, we show the effect of the lexicon's size: as expected, larger lexicons are better. The generative approach significantly outperforms the discriminative baseline at any lexicon size, although its advantage drops for smaller lexicons or larger training corpora.

In Figure 1 we found that increasing the pseudocount c consistently *decreases* performance, so we used $c = 1$ in our other experiments.[9]

6 Conclusion

This paper has described a generative model for low-resource sequence labeling and segmentation tasks using lexical resources. Experiments in semi-supervised and low-resource settings have demonstrated its applicability to part-of-speech induction and low-resource named-entity recognition. There are many potential avenues for future work. Our model may be useful in the context of active learning where efficient re-estimation and performance in low-data conditions are important. It would also be interesting to explore more expressive parameterizations, such recurrent neural networks for H_y. In the space of neural methods, differentiable memory (Santoro et al., 2016) may be more flexible than the PYP prior, while retaining the ability of the model to cache strings observed in the gazetteer.

Acknowledgments

This work was supported by the JHU Human Language Technology Center of Excellence, DARPA LORELEI, and NSF grant IIS-1423276. Thanks to Jay Feldman for early discussions.

[8]We also experimented with neural models, but found that the CRF outperformed them in low-data conditions.

[9]Why? Even a pseudocount of $c = 1$ is enough to ensure that $P_y(s) \gg H_y(s)$, since the prior probability $H_y(s)$ is rather small for most strings in the lexicon. Indeed, perhaps $c < 1$ would have increased performance, particularly if the lexicon reflects out-of-domain data. This could be arranged, in effect, by using a hierarchical Bayesian model in which the lexicon and corpus emissions are not drawn from the identical distribution P_y but only from similar (coupled) distributions.

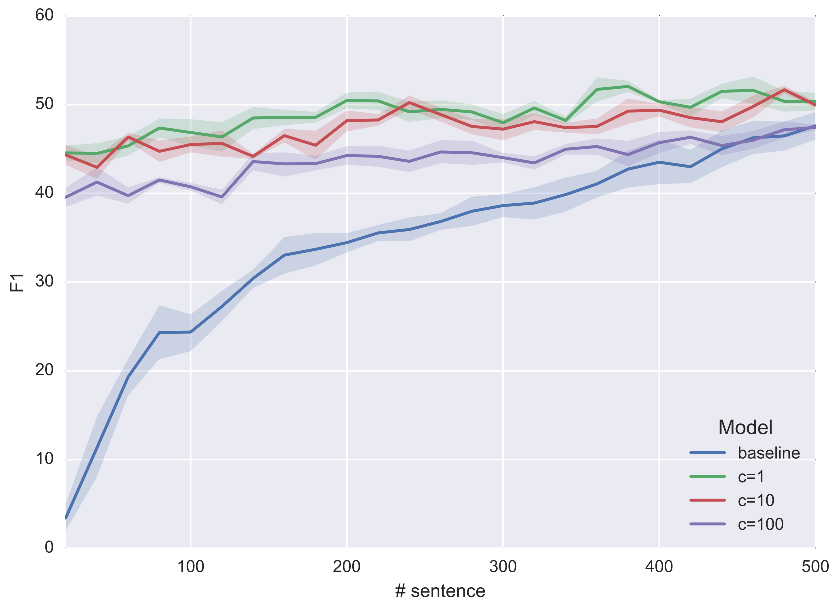

Figure 1: **Absolute** NER performance for Turkish (y-axis) as a function of corpus size (x-axis). The y-axis gives the F1 score on a held-out evaluation set (averaged over 10 bootstrap replicates, with error bars showing 95% confidence intervals). Our generative approach is compared to a baseline discriminative model with lexicon features (lowest curve). 500 held-out sentences were used to create the lexicon for both methods. Note that increasing the pseudocount c for lexicon entries (upper curves) tends to *decrease* performance for the generative model; we therefore take $c = 1$ in all other experiments. This graph shows Turkish; the corresponding Uzbek figure is available as supplementary material.

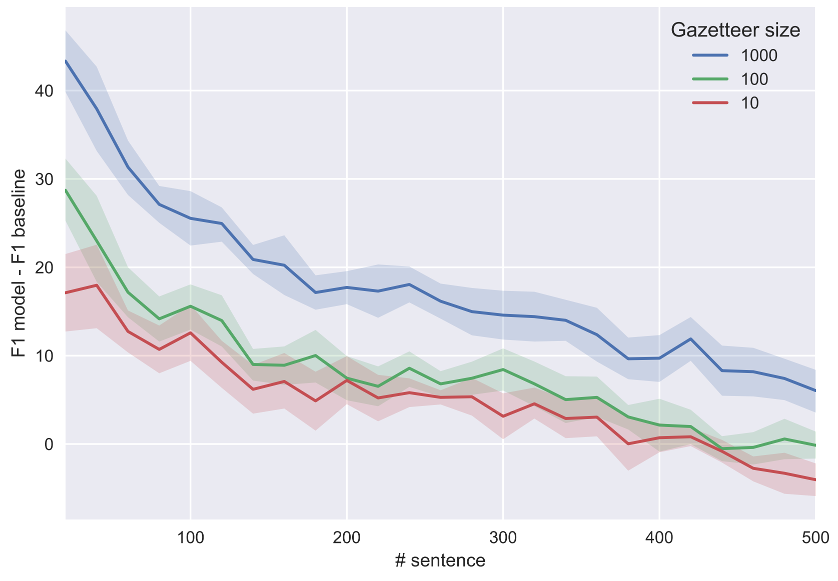

Figure 2: **Relative** NER performance for Turkish (y-axis) as a function of corpus size (x-axis). In this graph, $c = 1$ is constant and the curves instead compare different lexicon sizes derived from 10, 100, and 1000 held-out sentences. The y-axis now gives the *difference* $\mathrm{F1}_{\mathrm{model}} - \mathrm{F1}_{\mathrm{baseline}}$, so positive values indicate *improvement over the baseline* due to the proposed model. Gains are highest for large lexicons and for small corpora. Again, the corresponding Uzbek figure is available as supplementary material.

References

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research* 6:1817–1853.

Galen Andrew and Jianfeng Gao. 2007. Scalable training of L1-regularized log-linear models. In *Proceedings of the 24th International Conference on Machine Learning*. pages 33–40.

Christophe Andrieu, Arnaud Doucet, and Roman Holenstein. 2010. Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 72(3):269–342.

Phil Blunsom and Trevor Cohn. 2010. A hierarchical Pitman-Yor process HMM for unsupervised part-of-speech induction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*.

Benjamin Börschinger and Mark Johnson. 2011. A particle filter algorithm for Bayesian wordsegmentation. In *Proceedings of the Australasian Language Technology Association Workshop 2011*. Canberra, Australia, pages 10–18.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12:2493–2537.

Dipanjan Das and Slav Petrov. 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. pages 600–609.

Arnaud Doucet and Adam M. Johansen. 2009. A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of Nonlinear Filtering* 12:656–704.

Markus Dreyer and Jason Eisner. 2011. Discovering morphological paradigms from plain text using a Dirichlet process mixture model. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Edinburgh, pages 616–627.

Gregory Dubbin and Phil Blunsom. 2012. Unsupervised Bayesian part of speech inference with particle Gibbs. In *Proceedings of the 2012 European Conference on Machine Learning and Knowledge Discovery in Databases - Volume Part I*. Springer-Verlag, Berlin, Heidelberg, ECML PKDD'12, pages 760–773.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Stroudsburg, PA, USA, ACL '05, pages 363–370.

Dan Garrette and Jason Baldridge. 2012. Type-supervised hidden Markov models for part-of-speech tagging with incomplete tag dictionaries. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. pages 821–831.

Jan Gasthaus and Yee Whye Teh. 2010. Improvements to the sequence memoizer. In *Advances in Neural Information Processing Systems*. pages 685–693.

Sharon Goldwater and Thomas L. Griffiths. 2007. A fully Bayesian approach to unsupervised part-of-speech tagging. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*. Prague, Czech Republic, pages 744–751.

Sharon Goldwater, Mark Johnson, and Thomas L. Griffiths. 2005. Interpolating between types and tokens by estimating power-law generators. In *Advances in Neural Information Processing Systems*. pages 459–466.

Mark Johnson, Thomas L. Griffiths, and Sharon Goldwater. 2007. Bayesian inference for PCFGs via Markov chain Monte Carlo. In *HLT-NAACL*. pages 139–146.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *Computing Research Repository* arXiv:1602.02410.

Shen Li, João V. Graça, and Ben Taskar. 2012. Wiki-ly supervised part-of-speech tagging. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. pages 1389–1398.

Daichi Mochihashi, Takeshi Yamada, and Naonori Ueda. 2009. Bayesian unsupervised word segmentation with nested Pitman-Yor language modeling. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*. pages 100–108.

Andrew Y. Ng and Michael I. Jordan. 2002. On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes. *Advances in Neural Information Processing Systems* 2:841–848.

Jim Pitman. 1996. Some developments of the Blackwell-MacQueen urn scheme. In T. S. Ferguson, L. S. Shapley, and J. B. MacQueen, editors, *Statistics, Probability and Game Theory: Papers in Honor of David Blackwell*, Institute of Mathematical Statistics, volume 30 of *IMS Lecture Notes-Monograph series*, pages 245–267.

Jim Pitman and Marc Yor. 1997. The two-parameter Poisson-Dirichlet distribution derived from a stable subordinator. *The Annals of Probability* pages 855–900.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy P. Lillicrap. 2016. One-shot learning with memory-augmented neural networks. *Computing Research Repository* arXiv:1605.06065.

Andrew Smith, Trevor Cohn, and Miles Osborne. 2005. Logarithmic opinion pools for conditional random fields. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. pages 18–25.

Andrew Smith and Miles Osborne. 2006. Using gazetteers in discriminative information extraction. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*. pages 133–140.

Stephanie Strassel and Jennifer Tracey. 2016. Lorelei language packs: Data, tools, and resources for technology development in low resource languages. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*. European Language Resources Association (ELRA), Paris, France.

Charles Sutton, Michael Sindelar, and Andrew McCallum. 2006. Reducing weight undertraining in structured discriminative learning. In *Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*. Stroudsburg, PA, USA, HLT-NAACL '06, pages 89–95.

Yee Whye Teh. 2006. A hierarchical Bayesian language model based on Pitman-Yor processes. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*. pages 985–992.

Nilesh Tripuraneni, Shixiang Gu, Hong Ge, and Zoubin Ghahramani. 2015. Particle Gibbs for infinite hidden Markov models. In *Advances in Neural Information Processing Systems*. MIT Press, Cambridge, MA, USA, pages 2395–2403.

Frank Wood, Cédric Archambeau, Jan Gasthaus, Lancelot James, and Yee Whye Teh. 2009. A stochastic memoizer for sequence data. In *Proceedings of the 26th Annual International Conference on Machine Learning*. pages 1129–1136.

Supplementary Material

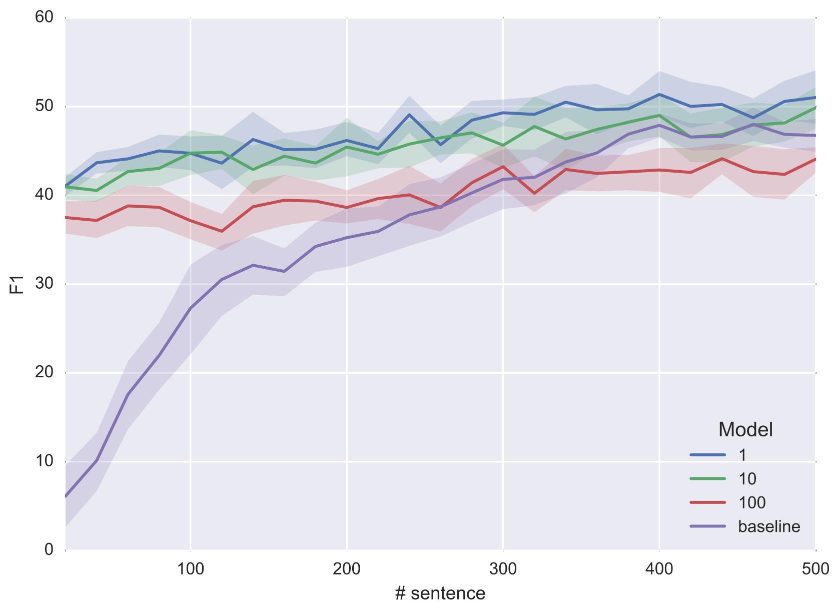

Figure 3: Version of Figure 1 for Uzbek.

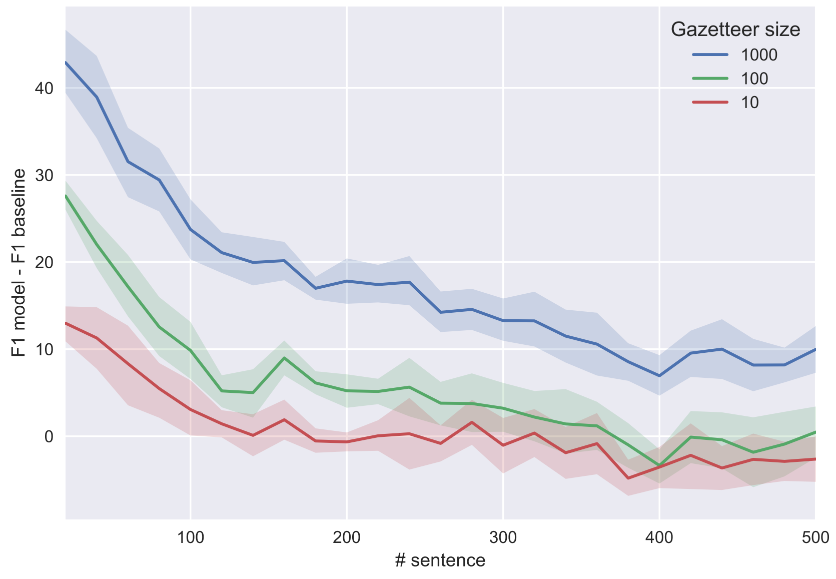

Figure 4: Version of Figure 2 for Uzbek.