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 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.

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.).

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 \( x = \text{Washington} \) 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 \( \text{Washington} \) 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 \( \frac{(cm_y)!}{(cm_y - m_y)!} \cdot (P_y(x_1)P_y(x_2)\cdots P_y(x_{m_y}))^c \). 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 \( P_y \). 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 \( \alpha \), 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 \( \Sigma \).

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

---

1Dictionaries 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.

2If 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 \( H_{\text{LOC}} \). It is \( P_{\text{LOC}} \) 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 \( H_{\text{LOC}} \) are far more frequent under \( P_{\text{LOC}} \), presumably because they were chosen as the names of actual, significant places. These place names were randomly drawn from \( H_{\text{LOC}} \) 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 \( \alpha \) 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 \( x = \text{Washington} \) 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 places. These place names were randomly drawn under \( H_{\text{LOC}} \), presumably because they are merely plausible under \( H_{\text{LOC}} \), and because they are far more efficient much as in equation (1).

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

\[
H_y(a_{1:j}) = \left( \prod_{j=1}^{J} H_y(a_j | a_{1:j-1}) \right) H_y(\$ | a_{1:j})
\]

\[
= \left( \prod_{j=1}^{J} H_y(a_{1:j-1}(a_j)) \right) H_y.a_{1:j} (\$) \tag{2}
\]

where \( H_y, u(a) \) is the conditional probability of character \( a \) given the left context \( u \in \Sigma^* \). Each \( H_y, u \) is a distribution over \( \Sigma \), defined recursively as

\[
H_y, \epsilon \sim \text{PYP}(d, \alpha, \epsilon, U_{\Sigma})
\]

\[
H_y, u \sim \text{PYP}(d_{|u|}, \alpha_{|u|}, H_y, \sigma(u))
\]

where \( \epsilon \) is the empty sequence, \( U_{\Sigma} \) is the uniform distribution over \( \Sigma \cup \{\$\} \), and \( \sigma(u) \) drops the first symbol from \( u \). The discount and concentration parameters \( (d_{|u|}, \alpha_{|u|}) \) are associated with the lengths of the contexts \( |u| \), 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 \( y = y_{1:T} \in \mathcal{Y}^* \).⁴ The observations

³We 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.

⁴The label sequence is terminated by a distinguished end-of-sequence label, again written as \$.
$x_{1:T}$ 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 $y$ 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,

$$p(y_{1:T}) := \left( \prod_{t=1}^{T} G_{y_{t-1}}(y_t) \right) G_{y_{1:T}}(§)$$

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

$$G_e \sim \text{PYP}(d_e, \alpha_e, U_\mathcal{Y})$$

$$G_e \sim \text{PYP}(d_v, \alpha_v, G_\sigma(v))$$

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

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 $x_t = \text{Washington}$ could have been emitted from the label $y_t = \text{LOC}$, or from $y_t = \text{PER}$, or from $y_t = \text{Washington}$ 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 $\mathcal{O}$. Of course, most labels are impossible at most positions (e.g., $y_t$ cannot be Washington unless $x_t = \text{Washington}$). 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 $y$ can learn that spoke to $\text{PER}$ yesterday is a common 4-gram in the label sequence $y$, and thus we are more likely to label Washington as a person if $x = \ldots \text{spoke to Washington yesterday} \ldots$.

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 $\mathcal{Y} = \mathcal{Y}' \cup \mathcal{O}$. But no uniform distribution exists over the infinite set $\mathcal{O}$, so how should we replace the base distribution $U_y$ over labels in equation (5)? Answer: To draw from the new base distribution, sample $y \sim U_{\mathcal{Y}' \cup \mathcal{O}}$. If $y = \mathcal{O}$, however, then “expand” it by resampling $y \sim H_{\mathcal{O}}$. Here $H_{\mathcal{O}}$ 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 $Y$ to be a random variable, we are interested in the posterior distribution $p(Y = y \mid x)$ over label sequences $y$ given the emitted word sequence $x$. Our model does not admit an efficient dynamic programming algorithm, owing to the dependencies introduced among the $Y_t$ 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 $y_{1:t-1}$, for two reasons: (1) The transition distribution $p(Y_t = y \mid y_{1:t-1})$ 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 $x$ has been observed. However, we are interested in the inference (or smoothing) problem that conditions on all of $x$ (Dubin and Blunsom, 2012; Tripuraneni et al., 2015).

SMC employs a proposal distribution $q(y \mid x)$.
whose definition decomposes as follows:

\[ q(y_1 \mid x_1) \prod_{t=2}^T q(y_t \mid y_{1:t-1}, x_{1:t}) \]

for \( T = \|x\| \). 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 \( q_t(\cdot \mid y_{1:t-1}, x_{1:t}) \) for \( t = 2, \ldots, T \). The final sampled sequence \( y \) is called a particle, and is given an unnormalized importance weight of

\[ \tilde{w} = \tilde{w}_T \cdot p(\$ \mid y_{1:T}) \]

where \( \tilde{w}_T \) was built up via

\[ \tilde{w}_t := \tilde{w}_{t-1} \cdot \frac{p(y_{1:t}, x_{1:t})}{p(y_{1:t-1}, x_{1:t-1})} q(y_t \mid y_{1:t-1}, x_{1:t}) \]

(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 \( \tilde{w}^{(m)} := \frac{\tilde{w}^{(m)}}{\sum_{m=1}^M \tilde{w}^{(m)}} \). As \( M \to \infty \), SMC provides a consistent estimate of the marginal likelihood \( p(x) \) 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 \( p(y \mid x) \) (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 \( y \) of a sentence \( x \) 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 \( y'_{1:T} \) and observations \( x_{1:T} \), 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 \( p(Y_t \mid y_{1:t-1}) \) 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

\[ q(y_t \mid y_{1:t-1}, x_{1:t}) := \frac{p(y_t \mid y_{1:t-1}) p(x_t \mid y_t)}{p(x_{t-1})} \]

where

\[ p(x_t \mid y_{1:t-1}) = \sum_{y_t \in \mathcal{Y}} p(y_t \mid y_{1:t-1}) p(x_t \mid y_t) \]

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

\[ \tilde{w}_t := \tilde{w}_{t-1} \cdot p(x_t \mid y_{1:t-1}) \]

(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:

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

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($x_{1:T}$, $y'_{1:T}$, $M$)  
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 $\tilde{w}^{(m)} = \tilde{w}_{T}^{(m)} p(\$|y_{1:T})$ for $m \in [1, M]$  
10: Draw index $k$ where $p(k = m) \propto \tilde{w}^{(m)}$  
11: return $y_{1:T}^{(k)}$

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 $p(y_{t} | y_{1:t-1})$, which determines the probability of a label in a given context, and an emission distribution $p(x_{t} | y_{t})$ (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 $\_ \in \Sigma$ 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, $x_{1:2} = \text{New York}$ would be annotated as a location segment by defining $y_{1:2} = I-\text{LOC}$ $E-\text{LOC}$. This says that $y_{1:2}$ has jointly emitted $x_{1:2}$, an event with prior probability $P_{\text{LOC}}(\text{New York})$. 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 $x_{3} = \text{hosted}$ then it should be tagged with $y_{3} = \text{hosted}$ 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 $p(y_{t} | y_{1:t-1}) = 1$, since then $y_{t}$ merely continues an existing segment. Otherwise $y_{t}$ starts a new segment, and we take $p(y_{t} | y_{1:t-1}) = 1$ to be defined by the PYP’s probability $G_{y_{1:t-1}}(y_{t})$ as usual, but where we interpret the subscript $y_{1:t-1}$ 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 $p(x_{t} | y_{t})$, since the choice of $x_{t}$ also depends on any words emitted earlier in the segment. Let $s \leq t$ be the starting position of the segment that contains $t$. If $y_{t} = E-y$, then the emission probability is proportional to $P_{y}(x_{s+1} x_{s+1} \cdots x_{t})$. 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 $\Sigma^{*}$ that have $x_{s} x_{s+1} \cdots x_{t}$ 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, 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-

---

https://www.wiktionary.org/
of-speech. In our setting, however, the dictionaries are not constraints but evidence. If \textit{monthly} is listed in (only) the adjective lexicon, this tells us that \( P_{\text{ADJ}} \) sometimes generates \textit{monthly} and therefore that \( H_{\text{ADJ}} \) may also tend to generate other words that end with \textit{-ly}. However, for us, \( P_{\text{ADV}}(\text{monthly}) > 0 \) as well, allowing us to still correctly treat \textit{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 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.

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 competitive, 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 \( y \in \{ \text{PER, LOC, ORG, GPE, CONTEXT} \} \), 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.

\(^7\)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.
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). 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.

---

9We also experimented with neural models, but found that the CRF outperformed them in low-data conditions.

---

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

<table>
<thead>
<tr>
<th>Model</th>
<th>Danish</th>
<th>German</th>
<th>Greek</th>
<th>English</th>
<th>Italian</th>
<th>Portuguese</th>
<th>Spanish</th>
<th>Swedish</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wiktionary</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SHMM-ME</td>
<td>83.3</td>
<td>85.8</td>
<td>79.2</td>
<td>87.1</td>
<td>86.5</td>
<td>84.5</td>
<td>86.4</td>
<td>86.1</td>
<td>84.9</td>
</tr>
<tr>
<td>HSM</td>
<td>83.7</td>
<td>90.7</td>
<td>81.7</td>
<td>84.0</td>
<td>86.7</td>
<td>85.5</td>
<td>87.6</td>
<td>86.8</td>
<td>85.8</td>
</tr>
<tr>
<td>Supervised</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SHMM-ME</td>
<td>93.9</td>
<td>97.4</td>
<td>95.1</td>
<td>95.8</td>
<td>93.8</td>
<td>95.5</td>
<td>93.8</td>
<td>95.5</td>
<td>95.1</td>
</tr>
<tr>
<td>HSM</td>
<td>95.2</td>
<td>97.4</td>
<td>97.4</td>
<td>95.2</td>
<td>94.5</td>
<td>96.0</td>
<td>95.6</td>
<td>92.2</td>
<td>95.3</td>
</tr>
</tbody>
</table>
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 $F1_{\text{model}} - F1_{\text{baseline}}$, 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


Supplementary Material

Figure 3: Version of Figure 1 for Uzbek.

Figure 4: Version of Figure 2 for Uzbek.