Variational Inference and Learning for a Unified Model of Syntax, Semantics and Morphology

Leonid Kontorovich   John Lafferty   David Blei*

April 2006
CMU-CALD-06-100

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

There have been recent attempts to produce trainable (unsupervised) models of human-language syntax and semantics, as well as morphology. To our knowledge, there has not been an attempt to produce a generative model that encorporates semantic, syntactic, and morphological elements. Some immediate applications of this tool are stemming, word clustering by root, and disambiguation (at the syntactic, semantic, and morphological levels). In this work, we propose a hierarchical topics-syntax-morphology model. We provide the variational inference and update rules for this model (exact inference is intractable). We show some preliminary results on segmentation tasks.

* Current address: Computer Science Department, Princeton University, Princeton, NJ 08540

Research supported in part by NSF grants IIS-0312814 and IIS- 0427206 and by the DARPA CALO project.

Keywords: syntax, semantics, morphology, variational inference

1 Introduction

There have been recent attempts to produce trainable models of human-language syntax and semantics [4, 6] as well as morphology [12].

Some immediate applications of an unsupervised syntax- semantics- morphology acquisition engine are stemming and word clustering by allomorphs. Disambiguation and inference (at all levels) can be a natural precursor to more complex tasks, such as information retrieval and machine translation. The problem of morphology is typically swept under the rug in NLP applications. One can get away with treating words as indivisible units in a language like English, with a relatively simple morphology; however, some languages are highly inflected and have many millions of possible words, necessitating some sort of morphological analysis. But even in English, one can profit by treating the words `assume`, `assuming`, `assumption` as variations on a common element as opposed to distinct and separate entities.

Previous approaches to this problem have ranged from ad-hoc heuristics [1, 11, 13] to principled approaches [4, 6]. However, to our knowledge, there has not been an attempt to produce a generative model that encorporates semantic, syntactic, and morphological elements. We claim that syntax and semantics are indispensible in acquiring a morphology, where ambgiuties abound both in the synthesis (e.g., past tense of `hang` can be `hanged` or `hung`) and analysis (`resent` can be a present tense verb, or the past tense of `resend`). Yarowsky and Wicentowski [13] indeed take syntactic information into account, but their model requires human supervision or language-specific knowledge.

This paper is organized as follows. In Section 2 we motivate paying attention to the issue of morphology and illustrate some of the difficulties involved. Section 3 reviews LDA and PSTs and describes our generative model. We present the variational inference calculations in Section 4 and give the learning updates in 5. Some preliminary results are given in Section 6, and we conclude with some future directions in Section 7.

As a notational convention, a boldfaced $\mathbf{w}$ will denote a document (sequence of words), and $|\mathbf{w}|$ its length, while w and $|w|$ denote a word (sequence of characters) and its length. The nature of the problem requires us to refer to characters i through j of the t^{th} word of the d^{th} document in the corpus; we use the (admittedly cumbersome) notation $w_{t,[i:j]}^{(d)}$ for this purpose. Mercifully, we will seldom need all the super/sub-scripts in a given expression, and will omit them if possiblele. We use $\mathbb{1}$ for the indicator function and ε for the null string. We will sometimes refer to topics as “semantic states”.

1. `batter`
   a. `[batter]` – liquidy dough
   b. `[bat][er]` – one who bats (in baseball)

2. `rung`
   a. `[rung]` – a horizontal bar in a ladder
   b. `[ring][PP]` – past participle of 'ring'

3. `dice`
   a. `[dice][PRES]` – to chop finely
   b. `[die][PLU]` – the plural form of 'die'

4. `rearrange`
   a. `[re][arrange][PRES]` – to arrange anew
   b. `[rear][range]` – the range of the rear

5. `resent`
   a. `[resent][PRES]` – to dislike
   b. `[re][send][PAST]` – past tense of 'resend' - to send again

Figure 1: Morphological ambiguity: analysis

2 Problem overview

Without getting into an involved discussion of linguistic phenomena or the computational issues that plague NLP research, we briefly illustrate the interdependence of syntax, semantics and morphology, and exhibit ambiguities at all of these levels. The problem is even more severe in the languages (of which there are many) whose morphology is richer than that of English.

Informally, to "parse" or "analyze" a word means to break it up into its semantic and morphological constituents. Figure 1 illustrates that analysis depends on the semantic and syntactic features of the word.

In the same informal sense, to "generate" or "synthesize" a word is to combine semantic components, along with grammatical features into a valid word-string. Figure 2 illustrates that the "inverse" direction of analysis – synthesis – also cannot be done independently of the syntax and semantics.

- English:
  dream + PAST → {dreamed, dreamt}
  tread + PAST → {treaded, trod}
- Russian:
  pulja + INSTR → {пулей, пулею }
  ljubov' + GENTV → {любви, любови }
- Hebrew:
  GLH + 3pers SNG MSC PAST REFL → {הִתְגַּלָּה, נִתְגַּלָּה}
  $MR + 3pers PLU FEM FUTR ACT → {יִשְׁמְרוּ, תִּשְׁמֹרְנָה}
- Latin:
  ama + 2pers SNG PERF IND ACT → {amavisti, amasti}
  ama + 3pers PLU PERF IND ACT → {amaverunt, amavere}

Figure 2: Morphological ambiguity: synthesis

3 Models

3.1 Latent Dirichlet allocation

Latent Dirichlet allocation (LDA) is the first fully generative topic model for text corpora[1]. Proposed in 2003 by Blei, Ng and Jordan [4], LDA is an elegant model that takes the assumption of word and document exchangeability (implicit in just about every approach, with the exception of [6]) to their logical conclusions. Understanding LDA is a prerequisite for making sense of our model; a detailed description is provided in [4]. In what follows, we assume a working familiarity with LDA, and limit ourselves to mapping the notation and terminology of Blei et al. to the present work.

In standard LDA, the vocabulary is a fixed set of V words, and each document is a bag of words, generated from a Dirichlet mixture of A topics. More explicitly, we equip the $(A-1)$-dimensional topic simplex with a Dirichlet distribution

$$\mathcal{D}(\boldsymbol{\lambda} \,|\, \boldsymbol{\alpha}) = \frac{\Gamma\left(\sum_{a=1}^{A} \alpha_a\right)}{\prod_{a=1}^{T} \Gamma(\alpha_a)} \lambda_1^{\alpha_1 - 1} \lambda_2^{\alpha_2 - 1} \ldots \lambda_A^{\alpha_A - 1}$$

parametrized by $\boldsymbol{\alpha} \in \mathbb{R}^A$. The other set of parameters is an $A \times V$ matrix

[1] Papadimitriou et al. [8] proposed in 1998 a generative model to explain the performance of latent semantic analysis (LSA). However, LSA [5] was in use as a heuristic method years before the post-hoc generative-model analysis became available.

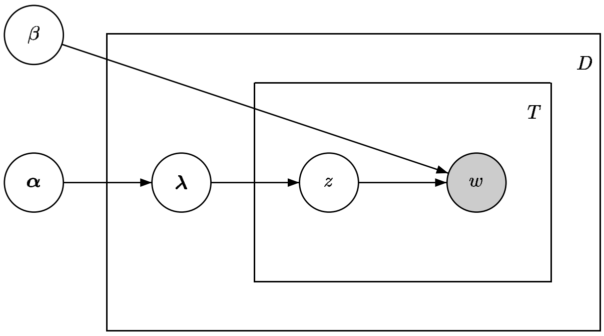


Figure 3: LDA graphical model

β for generating words from topics:

$$\beta_{aw} = \text{prob of generating word } w \text{ from topic } a.$$

For a fixed $\boldsymbol{\alpha}$, the document generation process is described below (and illustrated as a graphical model in Figure 3).

1. draw T according to some distribution on the naturals, such as $T \sim \text{Poisson}(\xi)$
2. draw $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_A) \sim \mathcal{D}(\boldsymbol{\alpha})$
3. for each of the T words w_t:
   (a) draw a topic $z_t \sim \text{Mult}(\boldsymbol{\lambda})$
   (b) draw a word $w_t \sim p(w_t \,|\, z_t, \beta)$

The corresponding marginal distribution of a document is

$$p(\mathbf{w} \,|\, \boldsymbol{\alpha}, \beta) = \int_\Delta \mathrm{d}\boldsymbol{\lambda} \mathcal{D}(\boldsymbol{\lambda} \,|\, \boldsymbol{\alpha}) \prod_{t=1}^{T} \sum_{a=1}^{A} p(z_t = a \,|\, \boldsymbol{\lambda}) p(w_t \,|\, z_t, \beta) \tag{1}$$

The inference problem for LDA is to compute the posterior $\mathbf{P}\{a_t = a \,|\, \mathbf{w}, \boldsymbol{\alpha}, \beta\}$, which requires computing the intractable integral

$$p(\mathbf{w} \,|\, \boldsymbol{\alpha}, \beta) = \frac{\Gamma\left(\sum_{a=1}^{A} \alpha_a\right)}{\prod_{a=1}^{T} \Gamma(\alpha_a)} \int_\Delta \mathrm{d}\boldsymbol{\lambda} \left(\prod_{a=1}^{A} \lambda_a^{\alpha_a - 1} \right) \left(\prod_{t=1}^{T} \sum_{a=1}^{A} \lambda_a \beta_{a w_t} \right). \tag{2}$$

One way of getting around the intractability is via a mean-field approximation; this, in fact, is the approach taken by Blei et al. This leads directly to a simple learning algorithm:

1. compute
$$q^{(d)}(a \mid w_t) \approx \mathbf{P}\left\{a_t = a \mid w_t^{(d)}\right\}$$
via mean-field for each document d

2. update β (normalized pseudo-counts):
$$\beta_{aw}^{\text{new}} \propto \sum_{d=1}^{D} \sum_{t=1}^{|\mathbf{w}^{(d)}|} q^{(d)}(a \mid w_t) \mathbb{1}_{\left\{w_t^{(d)}=w\right\}}$$

3. update $\boldsymbol{\alpha}$ (maximize lower bound on $\prod_d p(\mathbf{w}^{(d)})$):
$$\boldsymbol{\alpha}^{\text{new}} = \underset{\boldsymbol{\alpha}}{\operatorname{argmax}} \prod_d \frac{\prod_a \Gamma(\alpha_a + \sum_t q^{(d)}(a \mid w_t))}{\Gamma(\sum_a \alpha_a + |\mathbf{w}^{(d)}|)} \frac{\Gamma(\sum_a \alpha_a)}{\prod_a \Gamma(\alpha_a)}$$
(via Newton-Raphson)

3.2 Adding syntax

The top-level idea for combining the "semantics" of LDA with syntax and morphology is as follows. A document is generated by first sampling a sequence of syntactic states $\boldsymbol{\ell} = \ell_1 \ldots \ell_T$ from a Markov chain. In parallel, a topic mixture $\boldsymbol{\lambda}$ is sampled from a Dirichlet prior (parametrized by $\boldsymbol{\alpha}$) over the topic simplex. Each word w_t comes from a semantic topic a and a syntactic state ℓ, according to a morphological probability $\mathrm{P}_{\texttt{morph}}(w \mid \ell, a)$ (illustrated as a graphical model in Figure 4).

The probability of a document $\mathbf{w} = w_1 w_2 \ldots w_T$ is given by
$$p(\mathbf{w}) = \int_{\Delta_A} d\boldsymbol{\lambda} \mathcal{D}(\boldsymbol{\lambda} \mid \boldsymbol{\alpha}) \sum_{\boldsymbol{\ell}=\ell_1 \ldots \ell_T} \prod_t p(\ell_t \mid \ell_{t-1}) \sum_a \lambda_a \mathrm{P}_{\texttt{morph}}(w_t \mid \ell, a). \quad (3)$$

This is a strict generalization of LDA and therefore is also intractable. In Section 4 we calculate a variational mean-field approximation to the inference problem.

3.3 Morphological model

The crux of the modeling difficulty turned out to lie in choosing a good morphological model $\mathrm{P}_{\texttt{morph}}(w \mid \ell, a)$. The intuition is that most words have both a "syntactic" and a "semantic" component. Thus, in the word `reassignment`,

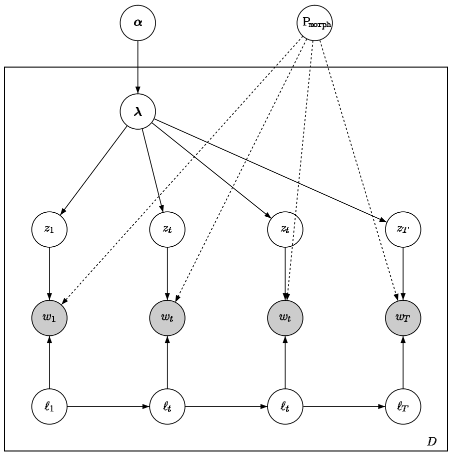


Figure 4: The graphical model for document generation; here $\mathrm{P}_{\texttt{morph}}$ is a "black box" plug-in module

the suffix `ment` can be assigned a syntactic role[2], while the root `assign` carries the semantics.

Since it is not obvious which parts of the word are semantic and which are syntactic, it makes sense to model this as a hidden Markov "microstate," taking on the binary values $m \in \{\texttt{SYN}, \texttt{SEM}\}$. This forces

$$\mathrm{P}_{\texttt{morph}}(w \mid \ell, a) = \sum_{m_1 \ldots m_{|w|}} \prod_{i=1}^{|w|} p(m_i \mid m_{i-1}) \mathrm{P}_{\texttt{chr}}(w_{[i]} \mid w_{[1:i-1]}, m_i, \ell, a). \quad (4)$$

We further assume a particular form of the character model $\mathrm{P}_{\texttt{chr}}(\cdot)$:

$$\mathrm{P}_{\texttt{chr}}(w_{[i]} \mid w_{[1:i-1]}, m_i, \ell, a) = \begin{cases} \mathrm{P}_{\texttt{chr}}(w_{[i]} \mid w_{[1:i-1]}, \ell), & m_i = \texttt{SYN} \\ \mathrm{P}_{\texttt{chr}}(w_{[i]} \mid w_{[1:i-1]}, a), & m_i = \texttt{SEM} \end{cases} . \quad (5)$$

A natural way to model bounded-memory sequences is via N-grams, which forces $\mathrm{P}_{\texttt{chr}}(w_{[i]} \mid w_{[1:i-1]}, \cdot) = \mathrm{P}_{\texttt{NGRM}}(w_{[i]} \mid w_{[i-N:i-1]}, \cdot)$. Thus, if there are L syntactic and A semantic states, we train $L + A$ different N-gram models, as described below. Given the trained models, $\mathrm{P}_{\texttt{morph}}(w \mid \ell, a)$ can be efficiently computed using forward-backward. The semantic N-grams model

[2]In English, prefixes are almost never dictated by syntax, and thus are almost entirely semantic. There are numerous languages, however, in which a prefix is obligatory, for example, in changing verb tense (cf, Russian, Latin, Greek, German). English suffixes, however, are frequently syntactic; thus, the suffix `ment` turns a verb into a noun.

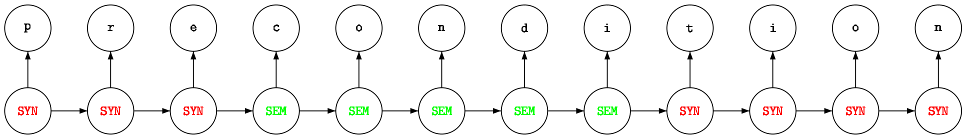


Figure 5: The word `precondition` generated from a sequence of microstates

word roots, while the syntactic N-grams model prefixes and suffixes. Observe that this model extends naturally to non-concatenative (e.g., Semitic) morphologies.

This model, though intuitively appealing, suffers from a subtle but serious drawback. N-grams always predict the next character based on a fixed-length history – there is no pressure to be "parsimonious" or sparse. Since transitions between microstates incur a probabilistic cost of $p(m_i \,|\, m_{i-1})$, a likelihood-maximizing algorithm based on N-grams will prefer not to incur this cost, and always to predict from either the syntactic or the semantic model. Nor does taking N to be small ($\sim$3) solve the problem, as this uniformly short context prevents the model from capturing word structure.

3.4 Probabilistic Suffix Trees

One way to encourage sparsity is to force the model to condition the next character on a variable-length histories, choosing a context length based on its informativeness. This desideratum lends itself naturally to a Probabilistic Suffix Tree (PST), first introduced in [10]. A PST exploits the sparsity of the data and only grows long contexts when this significantly improves next-character predictability.

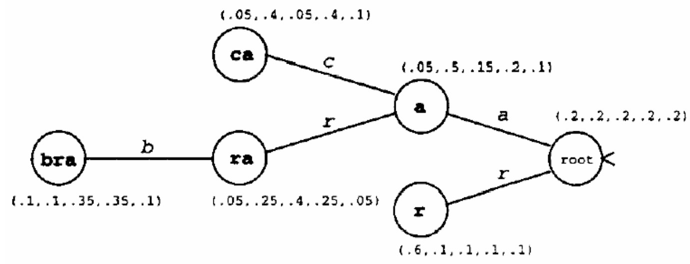
We follow the PST building algorithm of Bejerano and Yona [3]. As the details may be found in their paper, we give only a schematic presentation here. Let $\mathcal{A}$ be a finite alphabet and $S \subset \mathcal{A}^*$ a finite collection of strings. To each $s \in S$ associate a distribution $\gamma_s : \mathcal{A} \to [0,1]$. Then, modulo smoothing issues, a PST assigns probabilities to strings by

$$\mathrm{P}_{\texttt{TREE}}(w) = \prod_{i=1}^{|w|} \gamma_{s(i)}(w_{[i]}) \tag{6}$$

where $s(i)$ is the longest of $\{w_{[1:i-1]}, w_{[2:i-1]}, \ldots, w_{[i-2:i-1]}, \varepsilon\}$ s.t. $w_{[j:i-1]} \in S$; this is illustrated in Figure 6.

A rough sketch of the Bejerano-Yona PST building algorithm is as follows. We ignore all smoothing and some pruning issues here (but not in our

- $\mathcal{A} = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{r}\}$
- $S = \{\varepsilon, \mathtt{r}, \mathtt{a}, \mathtt{ca}, \mathtt{ra}, \mathtt{bra}\}$



- $\mathrm{P}_{\mathtt{TREE}}(\mathtt{abracadabra}) = \gamma_\varepsilon(\mathtt{a})\gamma_{\mathtt{a}}(\mathtt{b})\gamma_\varepsilon(\mathtt{r})\gamma_{\mathtt{r}}(\mathtt{a})\gamma_{\mathtt{bra}}(\mathtt{c})\gamma_\varepsilon(\mathtt{a})\gamma_{\mathtt{ca}}(\mathtt{d})\gamma_\varepsilon(\mathtt{a})\gamma_{\mathtt{a}}(\mathtt{b})\gamma_\varepsilon(\mathtt{r})\gamma_{\mathtt{r}}(\mathtt{a})$

Figure 6: Bejerano and Yona's illustration of how a tree assigns probabilities to words

implementation) – for details, consult [3]. The algorithm takes two main parameters: the maximal tree depth, N, and the sparsity, $r \geq 1$. This latter parameter controls tree pruning: setting $r = 1$ results in a full $|\mathcal{A}|$-ary tree of depth N, and increasing r results in sparser trees. For $s \in \mathcal{A}^*$, define χ_s to be the number of times the contiguous substring s occurs in the data. Define also

$$\chi_{s*} = \sum_{\sigma' \in \mathcal{A}} \chi_{s\sigma'}$$

and

$$\tilde{P}(\sigma \mid s) = \frac{\chi_{s\sigma}}{\chi_{s*}}.$$

Initialize S to $\emptyset$ and for each $s \in \mathcal{A}^{\leq N}$, include s in S if

$$\frac{\tilde{P}(\sigma \mid s)}{\tilde{P}(\sigma \mid s_{[2:|s|]})} \in (0, 1/r] \cup [r, \infty)$$

for some $\sigma \in \mathcal{A}$. (Bejerano and Yona do this efficiently via a pruning scheme.) The resulting character prediction probabilities, $\gamma_s(\sigma)$, are obtained by smoothing $\tilde{P}(\sigma \mid s)$.

3.5 Putting it together

Having defined $\mathrm{P}_{\mathtt{chr}}$ in terms of $\mathrm{P}_{\mathtt{TREE}}$, we've specified a generative model for documents at the character level. A document $\mathbf{w}$ is generated by the following process:

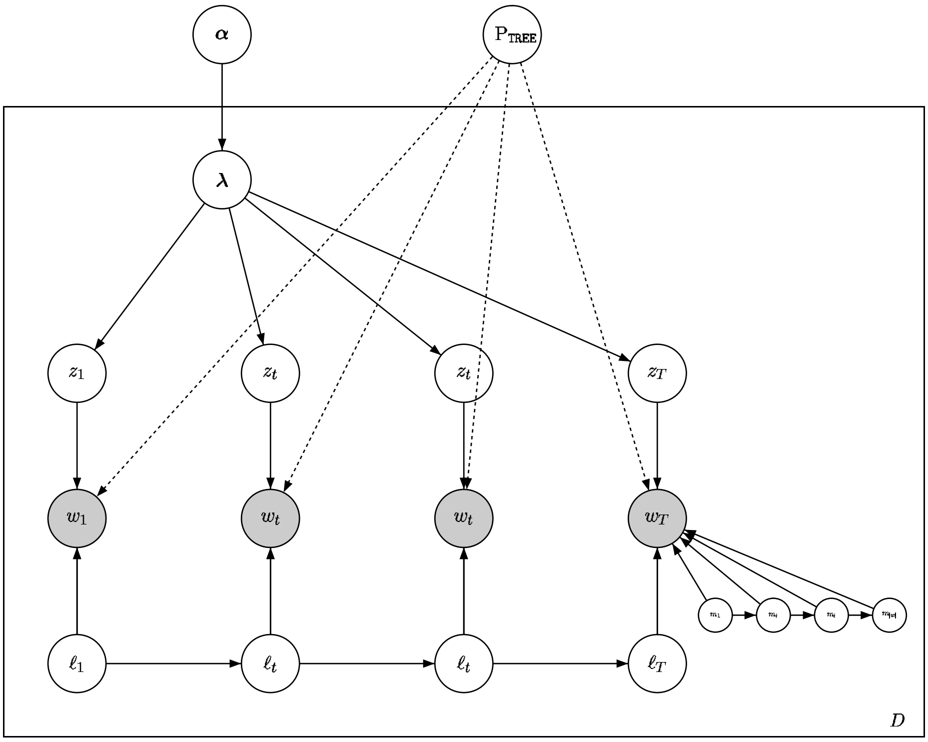


Figure 7: A graphical model representation of the document generation process

1. generate sequence of topics $a_1, \ldots, a_T$ as in LDA
2. generate sequence of syntactic states $\ell_1, \ldots, \ell_T$ Markovially
3. for each word w_t:
   (a) sample a Markov sequence of microstates $(m_i)_{i=1}^{|w|} \in \{\mathtt{SEM}, \mathtt{SYN}\}$
   (b) generate $w_{t,[i]}$ according to $\begin{cases} \mathrm{P}_{\mathtt{TREE}}(w_{[i]} \mid w_{[1:i-1]}, \ell), & m_i = \mathtt{SYN} \\ \mathrm{P}_{\mathtt{TREE}}(w_{[i]} \mid w_{[1:i-1]}, a), & m_i = \mathtt{SEM} \end{cases}$.

This process is represented graphically in Figure 7.

4 Variational Inference

Since our model is hierarchical, our inference will be also. The first step is, given a document $\mathbf{w} = w_1 w_2 \ldots w_T$, to compute the posterior $p(\ell, a \mid w_t)$

over the syntactic and semantic states.

Our likelihood $p(\mathbf{w} \,|\, \theta)$ is given by (3), where the parameters θ are

- the Dirichlet parameters α_a
- the syntactic state transition probabilities $p(\ell \,|\, \ell')$
- the microstate transition probabilities $p(m \,|\, m')$ (see (4) and (5)).
- the morphological character transition probabilities
- the character models, $\mathrm{P}_{\mathtt{chr}}(\cdot)$, derived from the smoothed string counts

Because $\boldsymbol{\lambda}$ is sampled for each document, different documents can exhibit the aspects in different proportions. However, the integral in (3) does not simplify and must be approximated; we do this by adapting the approach of Blei et al. [4]

To approximate the integral of a function, variational inference lower bounds the function and then integrates the lower bound. A simple lower bound for (3) comes from Jensen's inequality. The bound is parameterized by a vector $q(a \,|\, w)$:

$$\sum_a \lambda_a \, p(w \,|\, \ell, a) \;\geq\; \prod_a \left(\frac{\lambda_a \, p(w \,|\, \ell, a)}{q(a \,|\, w)} \right)^{q(a \,|\, w)} \tag{7}$$

$$\sum_a q(a \,|\, w) \;=\; 1 \tag{8}$$

The vector $q(a \,|\, w)$ can be interpreted as a soft assignment or "responsibility" of word w to the a^{th} aspect. While $q(a \,|\, w)$ doesn't explicitly depend on the syntactic state ℓ, it will be estimated in such a way that takes into account the posterior distribution of ℓ at w.

We will derive an approximate posterior for a document in the form $p(\boldsymbol{\lambda}, \boldsymbol{\ell}) = q(\boldsymbol{\lambda})\, q(\boldsymbol{\ell})$ which can be expressed in terms of the variational parameters $q(a \,|\, w)$. Although $\boldsymbol{\lambda}$ and $\boldsymbol{\ell} = \ell_1, \ldots, \ell_T$ are independent under this approximation, $q(\boldsymbol{\ell})$ is used to estimate $q(\boldsymbol{\lambda})$, and vice-versa.

Given bound parameters $q(a \,|\, w)$ for all a and w, the integral is now expressed as

$$p(\mathbf{w} \,|\, \theta) \;\geq \tag{9}$$
$$\left(\prod_{t,a} \left(\frac{1}{q(a \,|\, w_t)} \right)^{q(a \,|\, w_t)} \right) \sum_{\ell_1,\ldots,\ell_T} \prod_t p(\ell_t \,|\, \ell_{t-1}) \prod_a p(w_t \,|\, \ell_t, a)^{\,q(a \,|\, w_t)} D$$
$$D = \int_\Delta \mathcal{D}(\boldsymbol{\lambda} \,|\, \boldsymbol{\alpha}) \prod_a \lambda_a^{\sum_t q(a \,|\, w_t)} d\boldsymbol{\lambda} \;=\; \frac{\prod_a \Gamma(\alpha_a + \sum_t q(a \,|\, w_t))}{\Gamma(\sum_a \alpha_a + T)} \frac{\Gamma(\sum_a \alpha_a)}{\prod_a \Gamma(\alpha_a)}.$$

Now we find the best variational parameters by maximizing the value of the bound using EM. The "parameter" in the EM algorithm is $q(a \mid w)$ and the "hidden variables" are $\boldsymbol{\lambda}$ and $\boldsymbol{\ell}$.

$$\textbf{E-step:} \qquad \gamma_a \;=\; \alpha_a + \sum_t q(a \mid w_t) \tag{10}$$

$$\textbf{M-step:} \qquad q(a \mid w) \;\propto\; \exp(\Psi(\gamma_a)) \prod_{t:w_t=w} p(w_t \mid a, \ell)^{\frac{q(\ell \mid t)}{n_w}} \tag{11}$$

where Ψ is the digamma function and n_w is the number of times the word w occurs in the given document. Here $q(\ell \mid t)$ denotes the posterior probability that the state for w_t is ℓ, under the model

$$q(\mathbf{w}, \boldsymbol{\ell}) \;\propto\; \prod_t p(\ell_t \mid \ell_{t-1}) \prod_a p(w_t \mid \ell_t, a)^{\,q(a \mid w_t)}. \tag{12}$$

This posterior can be computed exactly using the forward-backward algorithm for HMMs (see [9]). The variables γ_a used in this algorithm can be interpreted as defining an approximate Dirichlet posterior $\mathcal{D}(\boldsymbol{\lambda} \mid \boldsymbol{\gamma})$ on $\boldsymbol{\lambda}$.

It remains to compute the posteriors $p(m \mid w[i], \ell, a)$ over the microstates, but this is done via the standard forward-backward algorithm.

5 Learning

Given a set of documents $\mathcal{C}$ indexed by $d = 1, ..., D$, the learning problem is to maximize the likelihood as a function of the parameters θ; the likelihood is given by $p(\mathcal{C} \mid \theta) = \prod_{d \in \mathcal{C}} p(\mathbf{w}^{(d)} \mid \theta)$, the latter probability given in (3). Notice that each document has its own integral over $\boldsymbol{\lambda}$. Standard EM, where we regard $\boldsymbol{\lambda}$ as a hidden variable for each document is impossible, since the E-step requires expectations over the posterior for $\boldsymbol{\lambda}$ and $\boldsymbol{\ell}$, which is an intractable distribution.

Using the variational estimate of the likelihood function for each document derived in the previous section, we can set the parameters to try to maximize the value of the estimate. This is the approach taken by Blei et al. in [4]. Using the variational bound (9), we apply an EM-like algorithm to estimate the parameters θ.

Let us write $q^{(d)}(\ell \mid t)$ and $q^{(d)}(a \mid t)$ for approximate posterior over syntactic and semantic states conditioned on the t^{th} word of the d^{th} document, as computed in was computed in Section 4.

The updates for the Dirichlet parameters $\boldsymbol{\alpha}$ are as in [4]: [3]

$$\boldsymbol{\alpha}^{\text{new}} = \underset{\boldsymbol{\alpha}}{\text{argmax}} \prod_d \frac{\prod_a \Gamma(\alpha_a + \sum_t q^{(d)}(a \mid t))}{\Gamma(\sum_a \alpha_a + |\mathbf{w}^{(d)}|)} \frac{\Gamma(\sum_a \alpha_a)}{\prod_a \Gamma(\alpha_a)}. \tag{13}$$

The update equation for the syntactic state transition probabilities are

$$p(\ell \mid \ell')^{\text{new}} \propto \sum_d \mathbb{E}_{q^{(d)}} \left[\sum_{t=1}^{|\mathbf{w}^{(d)}|} \mathbb{1}_{\{\ell_t = \ell)\}} \mathbb{1}_{\{\ell_{t-1} = \ell'\}} \right], \tag{14}$$

where $E_{q^{(d)}}$ is the expectation under the posterior distribution $q^{(d)}(\ell \mid t)$, computed using the forward-backward variables, as for HMMs. An analogous HMM-type update is used for the microstate transition probabilities:

$$p(m \mid m')^{\text{new}} \propto \sum_d \sum_{t=1}^{|\mathbf{w}^{(d)}|} \sum_a q^{(d)}(a \mid t) \sum_\ell q^{(d)}(\ell \mid t) \mathbb{E} \left[\sum_{i=1}^{|w_t^{(d)}|} \mathbb{1}_{\{m_i = m)\}} \mathbb{1}_{\{m_{i-1} = m'\}} \right]. \tag{15}$$

Finally, we must update the character probabilities $\mathrm{P}_{\texttt{TREE}}(\cdot \mid \ell)$ and $\mathrm{P}_{\texttt{TREE}}(\cdot \mid a)$, which are based on substring counts. We use the setup outlined in [3]. The Bejerano-Yona PST building algorithm takes as raw input χ_s – the number of times the contiguous substring s has in the data. Instead of these, we supply it with the pseudo-counts, based on the posteriors $q^{(d)}(a \mid w_t)$ and $q^{(d)}(\ell \mid w_t)$ for each document, as well as $p(m_i \mid w_t^{(d)}, \ell, a)$ for each word in the corpus. The pseudo-counts are

$$\chi_s^a = \sum_d \sum_{t=1}^{|\mathbf{w}^{(d)}|} q^{(d)}(a \mid w_t) \sum_{i=|s|+1}^{|w_t^{(d)}|} p(m_i = \texttt{SEM} \mid w_{t,[i-|s|:i-1]}^{(d)}, \ell, a) \tag{16}$$

and

$$\chi_s^\ell = \sum_d \sum_{t=1}^{|\mathbf{w}^{(d)}|} q^{(d)}(\ell \mid w_t) \sum_{i=|s|+1}^{|w_t^{(d)}|} p(m_i = \texttt{SYN} \mid w_{t,[i-|s|:i-1]}^{(d)}, \ell, a). \tag{17}$$

We then use the pseudo-counts just as if they were ordinary counts to build a semantic PST for each topic a and a syntactic PST for each state ℓ.

[3]In our experiments we did not update $\boldsymbol{\alpha}$, keeping it fixed at $\alpha_a = 1$.

As in the Blei et al. algorithm [4], once the parameters are changed, the optimal bound parameters $q(a \mid w)$ and $q(\ell \mid w)$ also change, so we simply alternate between optimizing the bound and applying these updates. Note that this "alternating maximization" algorithm is not an EM algorithm in the usual sense. It can only be understood as EM if we introduce additional hidden variables, so that the hidden variables are $\boldsymbol{\ell}$, $\boldsymbol{\lambda}$, m_i, and the "aspect assignments" to each word. An alternative approach is described in Section 5.2 of [7].

6 Preliminary results

6.1 Training data

Our corpus consisted of approximately 2000 documents of length around 200 words each, taken from part-of-speech tagged Wall Street Journal news articles. We collapsed the POS tags into 15 syntactic categories and set the number of topics at 25.

Additionally, we generated a synthetic corpus of 2000 documents with word-length 200, with 4 syntactic states and 10 topics. The topic z_t for each document were sampled from a Dirichlet mixture, just as in LDA. The syntactic states were sampled from a Markov chain. The words were generated from a simple (semantic prefix) + (syntactic suffix) morphological model, via multinomial distributions on small set of prefixes (for each topic) and a small set of suffixes (for each syntactic state).

6.2 Recovering LDA

We have not performed extensive, quantitative experiments with our model, so the results in this section will be of tentative qualitative nature. The first "sanity check" is that our model is able to recover the topics of ordinary LDA (see [4] for an illustration of the topics learned by LDA). It is indeed able to do so, with a performance indistinguishable (as subjectively judged by humans) from regular LDA, both on the WSJ and the synthetic corpus.

6.3 Word segmentation

Since the main innovation of this work is an encorporation of morphology into the generative model, it seems natural to examine our model's behavior on word segmentation tasks. We took words with common noun suffixes: `tion`, `ness`, `ism`, etc., and plotted the posterior probability $p(m_i = \texttt{SEM} \mid w)$

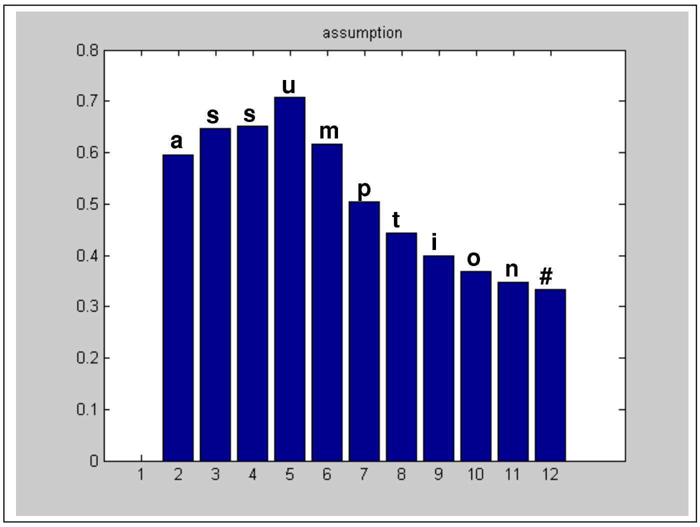


Figure 8: Posterior semantic-character probability for `assumption`

of each character coming from the semantic state. We intuitively expect this quantity to increase as more of the "semantic" prefix has been traversed and sharply plummet once the "syntactic" suffix has been reached. A representative example is given in Figure 8.

The drop in the semantic probability is not as sharp as we would like; see Section 7 for ideas on improving this.

We computed the posterior semantic probabilities for the synthetic data as well, and averaged these over the words ending in a given suffix. These averages are plotted in Figure 9, for the suffixes `ism`, `ist`, `ness`, `tion`.

There is clearly some sort of word segmentation taking place, roughly at the correct location in the word. However, the transitions from the "semantic" segment to the "syntactic" one are not as crisp as we had expected, nor can we explain the slight increase at the end of the `ness` and `tion` plots.

7 Future directions

Although the results seem to indicate that the proposed model is at least plausible, there are some immediate directions for improvement. Our tree-building procedure, borrowed from [3], is not a true EM step. Additionally, this procedure has several tuning parameters, most importantly the sparsity parameter r. The empirical trials show that the results are highly sensitive to

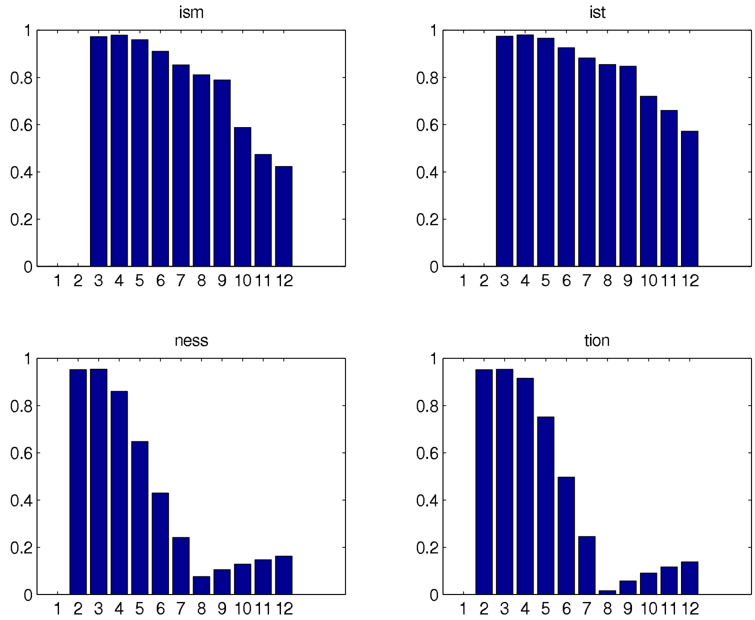


Figure 9: Averaged posterior semantic probability for the synthetic corpus

this r parameter, which was more or less hand-tuned for the results shown here. Both of these issues can be addressed by defining a Bayesian prior over the PSTs, and growing the trees in a MAP update. We look forward to attending to these in forthcoming work.

References

[1] M. Baroni, J. Matiasek, H. Trost. "Unsupervised discovery of morphologically related words based on orthographic and semantic similarity." ACL Workshop on Morphological and Phonological Learning, 2002.

[2] M. J. Beal and Z. Ghahramani. "The Variational Bayesian EM Algorithm for Incomplete Data: with Application to Scoring Graphical Model Structures." *Bayesian Statistics* 7, 2003.

[3] G. Bejerano and G. Yona. "Variations on probabilistic suffix trees: statistical modeling and the prediction of protein families." *Bioinformatics*, 17(1), pp 23-43, 2001.

[4] D. Blei, A. Ng and M. Jordan. “Latent Dirichlet allocation.” *Journal of Machine Learning Research*, 3, 993-1022, 2003.

[5] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, R. Harshman. “Indexing by latent semantic analysis.” *Journal of the American Society for Information Science* 41(6), 391-407, 1990.

[6] T. Griffiths, M. Steyvers, D. Blei, and J. Tenenbaum. “Integrating topics and syntax.” *NIPS*(17), 2005.

[7] T. Minka and J. Lafferty. “Expectation-propagation for the generative aspect model.” *Uncertainty in Artificial Intelligence*, 2002.

[8] C. Papadimitriou, H. Tamaki, P. Raghavan and S. Vempala. “Latent Semantic Indexing: A Probabilistic Analysis.” *PODS*, 1998.

[9] L. Rabiner, and B. Juang. *Fundamentals of Speech Recognition.* Prentice Hall, 1993.

[10] D. Ron, Y. Singer and N. Tishby. “The power of amnesia: learning probabilistic automata with variable memory length.” *Machine Learning*, 25(2), pp 117-150, 1996.

[11] P. Schone and D. Jurafsky. “Knowledge-Free Induction of Morphology Using Latent Semantic Analysis” Proceedings of the Fourth Conference on Computational Natural Language Learning and of the Second Learning Language in Logic Workshop, Lisbon, 2000.

[12] M. Snover and M. Brent. “A Probabilistic Model for Learning Concatenative Morphology” Proceedings of NIPS 2002.

[13] D. Yarowsky and R. Wicentowski. “Minimally Supervised Morphological Analysis by Multimodal Alignment.” Proceedings of ACL-2000, Hong Kong, pages 207-216, 2000.