Unsupervised Analysis of Structured Human Artifacts

by

Taylor Berg-Kirkpatrick

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Dan Klein, Chair
Professor Michael I. Jordan
Professor Line Mikkelsen

Fall 2015

Unsupervised Analysis of Structured Human Artifacts

Abstract

Unsupervised Analysis of Structured Human Artifacts

by

Taylor Berg-Kirkpatrick

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Dan Klein, Chair

The presence of hidden structure in human data—including natural language but also sources like music, historical documents, and other complex artifacts—makes this data extremely difficult to analyze. In this thesis, we develop unsupervised methods that can better cope with hidden structure across several domains of human data. We accomplish this by incorporating rich domain knowledge using two complementary approaches: (1) we develop detailed generative models that more faithfully describe how data originated and (2) we develop structured priors that create useful inductive bias.

First, we find that a variety of transcription tasks—for example, both historical document transcription and polyphonic music transcription—can be viewed as linguistic decipherment problems. By building a detailed generative model of the relationship between the input (e.g. an image of a historical document) and its transcription (the text the document contains), we are able to learn these models in a completely unsupervised fashion—without ever seeing an example of an input annotated with its transcription—effectively deciphering the hidden correspondence. The resulting systems have turned out not only to work well for both tasks—achieving state-of-the-art-results—but to outperform their supervised counterparts.

Next, for a range of linguistic analysis tasks—for example, both word alignment and grammar induction—we find that structured priors based on linguistically-motivated features can improve upon state-of-the-art generative models. Further, by coupling model parameters in a phylogeny-structured prior across multiple languages, we develop an approach to multilingual grammar induction that substantially outperforms independent learning.

To Johnny Jewell

Despite all the computations
You could just dance her to that rock 'n' roll station

Contents

List of Figures

List of Tables

Acknowledgments

I am extremely grateful to have been surrounded by such intelligent and amazing people during my time in grad school. My advisor, Dan Klein, has a particular kind of brilliance that is very unusual. It's not just that he's an exceptional researcher, mentor, teacher, writer, talker—it's the way that all these skills give the impression that you're interacting with a deep and consistent *style* of intelligence. This style, sensibility, and depth comes out in everything he touches. Perhaps most useful to me, as his student, Dan believes this sensibility can be taught. He never stopped pushing us to make the paper, the model, the talk... *better*, aiming for that almost supernatural higher standard. To me, this has always meant that he believed we could do it, and eventually it came to mean that I believed I could do it. He cares deeply about his students and I simply could not have had a better advisor.

Dan's students have been terrific compatriots and collaborators during my time in grad school—these are some of the smartest (and funniest) people I have ever known. Thank you to the old school group that taught me things when I was a undergrad: Aria Haghighi, Alex Bouchard, Percy Liang, John DeNero, Adam Pauls, John Blitzer, and David Burkett. Thanks to my contemporaries: David Hall, Mohit Bansal, Greg Durrett, Jono Kummerfeld, and Dave Golland. And thanks to the new generation: Jacob Andreas and Max Rabinovich. I was also lucky to collaborate with researchers at other institutions. I'd like thank Kevin Knight, Dirk Hovy, Megha Srivastava, Ashish Vaswani, Dan Garrette, and Hannah Alpert-Abrams.

My family and my oldest friends have supported me along the way and I want to thank them deeply for this. My mom for her constant care, love, tolerance, and surprisingly apt proof-reading of manuscripts far from her field of study. My dad for his drive, his compassion, and his unbounded enthusiasm. To both my parents: I could not have done any of this without you, it was you guys who taught me work ethic, meaning, and value, and you protected and supported me the whole way through. Also, throughout grad school, my friends Zack Deak, Sam Mullinax, Robert Demonte, and Jessica Matthews have kept me sane with their humor and care.

And of course, to my partner, Tali Hammond, whose unfaltering support has kept me going and whose opinions I trust so deeply: Thank you for being there to think through all the things with me, and for tolerating my sometimes craziness. Without Tali, I really can't imagine facing grad school at all.

Finally, I'd like to thank my dear friend Johnny Jewell. Johnny was many things to me, but most relevant to my time in grad school was Johnny's spirit of raw and absolute defiance. He taught me you never ever have to give in—no matter what.

Thanks to everyone! All the people I've encountered and known during this adventure, and thank you to the city of Berkeley, a place I will miss dearly.

Chapter 1

Introduction

1.1 Motivation

A major challenge in the analysis of structured human data—including natural language but also sources like music, historical documents, and other complex artifacts—is the presence of hidden structure between the input and almost anything you want to predict. Suppose you want a system to translate an English sentence into Chinese. The system will need to know something about the syntactic correspondence between these two languages. For example, in English, prepositional phrases follow verb phrases, but in Chinese, they precede them. Suppose a system must decipher an ancient document. Then it will need to reason about the structure of the font, blotches of ink, and page distortion.

None of this information can be simply read off the input. It has to predicted, learned, deciphered. The dominant approach in the field of natural language processing (NLP) is to use supervised systems that predict structure by learning from labeled training data. While approaches based on supervised learning have proven more robust and easier to build than the rule-based systems of the past, annotating training data is time consuming and tackling new domains requires new annotation efforts. For example, most state-of-the-art syntactic parsers are trained on the Wall Street Journal. If you want a high-performance parser for blog data you need to hire a team of linguists to annotate twenty-thousand blog posts.

In contrast, this thesis focuses on the development of unsupervised approaches—methods that learn and decipher hidden structure without hand-labeled training data. While potentially very useful, historically, unsupervised approaches have been harder to make work than their supervised counterparts. Without careful consideration of the relationship between input and output, the problem can be under-constrained. In NLP, most unsupervised approaches use carefully constructed generative models that specify this relationship and impose useful constraints on the learning problem. We now briefly give some background on the general framework using the unsupervised decipherment system of Ravi and Knight (2011) as a running example.

1.2 Unsupervised Learning with Generative Models

A generative model describes the distribution of both the observed random variables present in the input data, x, and the latent random variables posited by the type of analysis we want to carry out, z. Often, the latent variables are explanatory in nature or have a causal relationship with the observed variables. For example, borrowing from the unsupervised decipherment methods of Ravi and Knight (2011), x might be a substitution cipher written in strange symbols—which we observe—while z might be the plaintext that generated the cipher—which we would like to uncover (see Figure 1.1). A generative model is parameterized by a vector θ, which is usually generated from a prior. Learning involves estimating θ based on the observations, x. A typical approach—one which we use throughout this thesis—is to use maximum a posteriori (MAP) estimation of the parameter values, with the latent variables marginalized out[1]. The learning problem can be written as:

$$\text{[Learning]} \quad \hat{\theta}(x) = \text{argmax}_\theta \sum_z \Big[\underset{\text{[Model]}}{P(x, z; \theta)} \cdot \underset{\text{[Prior]}}{P(\theta)} \Big]$$

This optimization problem depends on the structure of both the model and prior and is usually non-convex. A standard approach for finding approximate parameter estimates corresponding to local optima is to use expectation maximization (EM) (Dempster et al. 1977), though other methods can also be effective. Once the parameter estimates, $\hat{\theta}(x)$, have been found, the desired output is attained by computing the maximum likelihood setting of the latent variables under the parameter estimates, conditioned on the observations:

$$\text{[Decoding]} \quad \hat{z}(x) = \text{argmax}_z \; P\big(x, z; \hat{\theta}(x)\big)$$

In order for this approach to work, both the model and the prior must constrain the learning problem in a way that targets the particular structure we are trying to uncover. One tactic is to design the generative model so that it encodes domain knowledge about the underlying causal process that generated the data. Returning to the example of decipherment, Ravi and Knight (2011) use a generative model (depicted in Figure 1.1) that encodes important assumptions about how substitution ciphers are generated: well-formed english plaintext is generated from a sequential language model and then each plaintext character is substituted for a cipher symbol based on a consistent mapping. In addition, their prior, $P(\theta)$, encodes a useful assumption: the mapping between plaintext characters and cipher symbols is sparse and most characters will consistently map to only one cipher symbol. Together, these modeling assumptions make the unsupervised learning problem feasible.

[1] A range of work has explored more sophisticated Bayesian methods that do not rely on point estimates of parameters. While these techniques sometimes yield improvements, they are often more computationally intensive and we opt for the simpler approach of MAP estimation throughout this thesis.

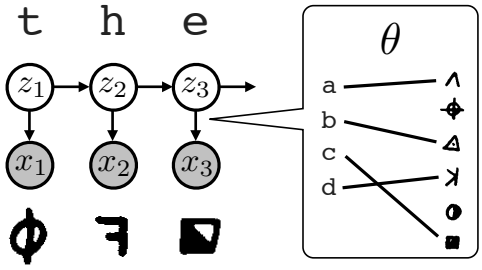


Figure 1.1: An example generative model used for an unsupervised task: decipherment.

Of course, decipherment is just one example. Similar unsupervised approaches have be used to tackle a variety of problems in NLP. For example, work on unsupervised syntactic analysis has used chain-structured models to induce parts of speech, while specialized hierarchical models have been used to target other syntactic relationships like syntactic dependency and constituency (Klein and Manning 2004). Unsupervised generative models of string transduction have been used for inducing ancient word forms in the related field of historical linguistics (Bouchard-Côté et al. 2008). In NLP, perhaps the most-widely successful application of unsupervised learning is bilingual word alignment (Och and Ney 2004), which is a core component of many modern machine translation systems. What these techniques all have in common is that instead of manually annotating structure through labeled training examples, structure is implicitly specified through careful model design that encodes inductive bias.

1.3 Broader Models and Better Priors

In this thesis, we develop unsupervised methods that can better cope with hidden structure across several domains of human data. We accomplish this by incorporating rich domain knowledge using two complementary approaches: (1) we broaden the range of data that unsupervised models can be applied to by developing detailed generative models of artifacts that are continuous in nature—like document images and musical audio, and (2) we develop linguistically-motivated priors that create useful inductive bias to make progress on core linguistic analysis tasks in NLP. The resulting systems have turned out not only to work well for a range of different tasks, but to outperform their supervised counterparts in some cases.

Outline and Contributions

First, we find that a variety of transcription tasks—for example, both historical document transcription and polyphonic music transcription—can be viewed as linguistic decipherment problems. By building a detailed generative model of the relationship between the input (e.g. an image of a historical document) and its transcription (the text the document contains), we are able to learn these models in a completely unsupervised fashion—without ever seeing an example of an input annotated with its transcription—effectively deciphering the hidden correspondence. In Chapter 2 we present the generative model for historical document transcription. In Chapter 3 we extend this model to include a more accurate representation of the historical typesetting process. In Chapter 4 we present a related generative model

for a completely different task: transcription of polyphonic piano music into a symbolic representation.

Next, for a range linguistic analysis tasks—for example, both word alignment and grammar induction—we find that structured priors based on linguistically-motivated features can improve upon state-of-the-art generative models. In Chapter 5 we demonstrate how features can easily be added to a range standard NLP models to build better priors. In Chapter 6, we extend this approach to build a phylogeny-structured prior across multiple languages and develop a system for multilingual grammar induction. This system substantially outperforms independent learning and achieves larger gains both from more articulated phylogenies and from increasing numbers of languages.

The contributions of this thesis are:

- A new generative model, inspired by historical printing processes, for transcribing images of documents from the printing press era in an unsupervised fashion (Chapter 2). This chapter is largely based on work previously published in Berg-Kirkpatrick et al. (2013).
- Richer typesetting models that extend the unsupervised historical document recognition system described in Chapter 2 (Chapter 3). This chapter is largely based on work previously published in Berg-Kirkpatrick and Klein (2014).
- A new generative model for transcribing piano music from audio to a symbolic form in an unsupervised fashion (Chapter 4). This chapter is largely based on work previously published in Berg-Kirkpatrick, Andreas, et al. (2014).
- An empirical demonstration of how features can easily be added to standard generative models for unsupervised learning in order to build linguistically-motivated priors (Chapter 5). This chapter is largely based on work previously published in Berg-Kirkpatrick, Bouchard-Côté, et al. (2010).
- A new approach to multilingual grammar induction that incorporates a phylogeny-structured prior describing parameter drift (Chapter 6). This chapter is largely based on work previously published in Berg-Kirkpatrick and Klein (2010).

Chapter 2

Transcription of Historical Documents

2.1 Overview

In this chapter we present an unsupervised method for transcribing images of historical documents into text. Standard techniques for transcribing modern documents do not work well on historical ones. For example, even state-of-the-art OCR systems produce word error rates of over 50% on the documents shown in Figure 2.1. Unsurprisingly, such error rates are too high for many research projects (Arlitsch and Herbert 2004; Shoemaker 2005; Holley 2010). We present a new, generative model specialized to transcribing printing-press era documents. Our model is inspired by the underlying printing processes and is designed to capture the primary sources of variation and noise.

One key challenge is that the fonts used in historical documents are not standard (Shoemaker 2005). For example, consider Figure 2.1a. The fonts are not irregular like handwriting – each occurrence of a given character type, e.g. `a`, will use the same underlying glyph. However, the exact glyphs are unknown. Some differences between fonts are minor, reflecting small variations in font design. Others are more severe, like the presence of the archaic long `s` character before 1804. To address the general problem of unknown fonts, our model *learns* the font in an unsupervised fashion. Font shape and character segmentation are tightly coupled, and so they are modeled jointly.

A second challenge with historical data is that the early typesetting process was noisy. Hand-carved blocks were somewhat uneven and often failed to sit evenly on the mechanical baseline. Figure 2.1b shows an example of the text's baseline moving up and down, with varying gaps between characters. To deal with these phenomena, our model incorporates random variables that specifically describe variations in vertical offset and horizontal spacing.

A third challenge is that the actual inking was also noisy. For example, in Figure 2.1c some characters are thick from over-inking while others are obscured by ink bleeds. To be robust to such rendering irregularities, our model captures both inking levels and pixel-level

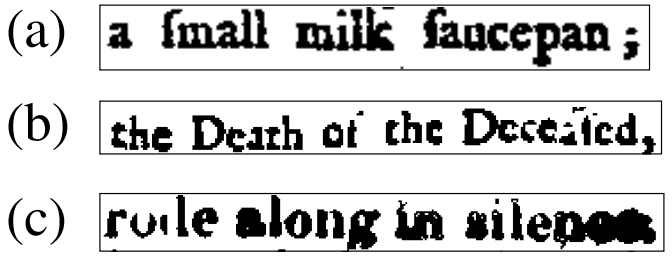


Figure 2.1: Portions of historical documents with (a) unknown font, (b) uneven baseline, and (c) over-inking.

noise. Because the model is generative, we can also treat areas that are obscured by larger ink blotches as unobserved, and let the model predict the obscured text based on visual and linguistic context.

Our system, which we call *Ocular*, operates by fitting the model to each document in an unsupervised fashion. The system outperforms state-of-the-art baselines, giving a 47% relative error reduction over Google's open source Tesseract system, and giving a 31% relative error reduction over ABBYY's commercial FineReader system, which has been used in large-scale historical transcription projects (Holley 2010).

2.2 Related Work

Relatively little prior work has built models specifically for transcribing historical documents. Some of the challenges involved have been addressed (Ho and Nagy 2000; Huang et al. 2006; Kae and Learned-Miller 2009), but not in a way targeted to documents from the printing press era. For example, some approaches have learned fonts in an unsupervised fashion but require pre-segmentation of the image into character or word regions (Ho and Nagy 2000; Huang et al. 2006), which is not feasible for noisy historical documents. Kae and Learned-Miller (2009) jointly learn the font and image segmentation but do not outperform modern baselines.

Work that has directly addressed historical documents has done so using a pipelined approach, and without fully integrating a strong language model (Vamvakas et al. 2008; Kluzner, Tzadok, Shimony, et al. 2009; Kae, Huang, et al. 2010; Kluzner, Tzadok, Chevion, et al. 2011). The most comparable work is that of Kopec and Lomelin (1996) and Kopec, Said, et al. (2001). They integrated typesetting models with language models, but did not model noise. In the NLP community, generative models have been developed specifically for correcting outputs of OCR systems (Kolak et al. 2003), but these do not deal directly with images.

A closely related area of work is automatic decipherment (Ravi and Knight 2008; Snyder, Barzilay, et al. 2010; Ravi and Knight 2011; Berg-Kirkpatrick and Klein 2011). The

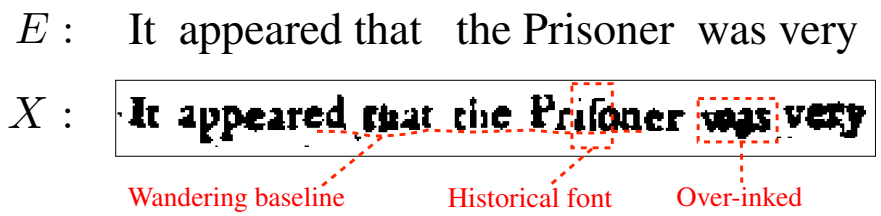


Figure 2.2: An example image from a historical document (X) and its transcription (E).

fundamental problem is similar to our own: we are presented with a sequence of symbols, and we need to learn a correspondence between symbols and letters. Our approach is also similar in that we use a strong language model (in conjunction with the constraint that the correspondence be regular) to learn the correct mapping. However, the symbols are not noisy in decipherment problems and in our problem we face a grid of pixels for which the segmentation into symbols is unknown. In contrast, decipherment typically deals only with discrete symbols.

2.3 Model

Most historical documents have unknown fonts, noisy typesetting layouts, and inconsistent ink levels, usually simultaneously. For example, the portion of the document shown in Figure 2.2 has all three of these problems. Our model must handle them jointly.

We take a generative modeling approach inspired by the overall structure of the historical printing process. Our model generates images of documents line by line; we present the generative process for the image of a single line. Our primary random variables are E (the text) and X (the pixels in an image of the line). Additionally, we have a random variable T that specifies the layout of the bounding boxes of the glyphs in the image, and a random variable R that specifies aspects of the inking and rendering process. The joint distribution is:

$$
\begin{aligned}
&P(E, T, R, X) = \\
&\quad P(E) && \text{[Language model]} \\
&\quad \cdot P(T|E) && \text{[Typesetting model]} \\
&\quad \cdot P(R) && \text{[Inking model]} \\
&\quad \cdot P(X|E, T, R) && \text{[Noise model]}
\end{aligned}
$$

We let capital letters denote vectors of concatenated random variables, and we denote the individual random variables with lower-case letters. For example, E represents the entire sequence of text, while e_i represents ith character in the sequence.

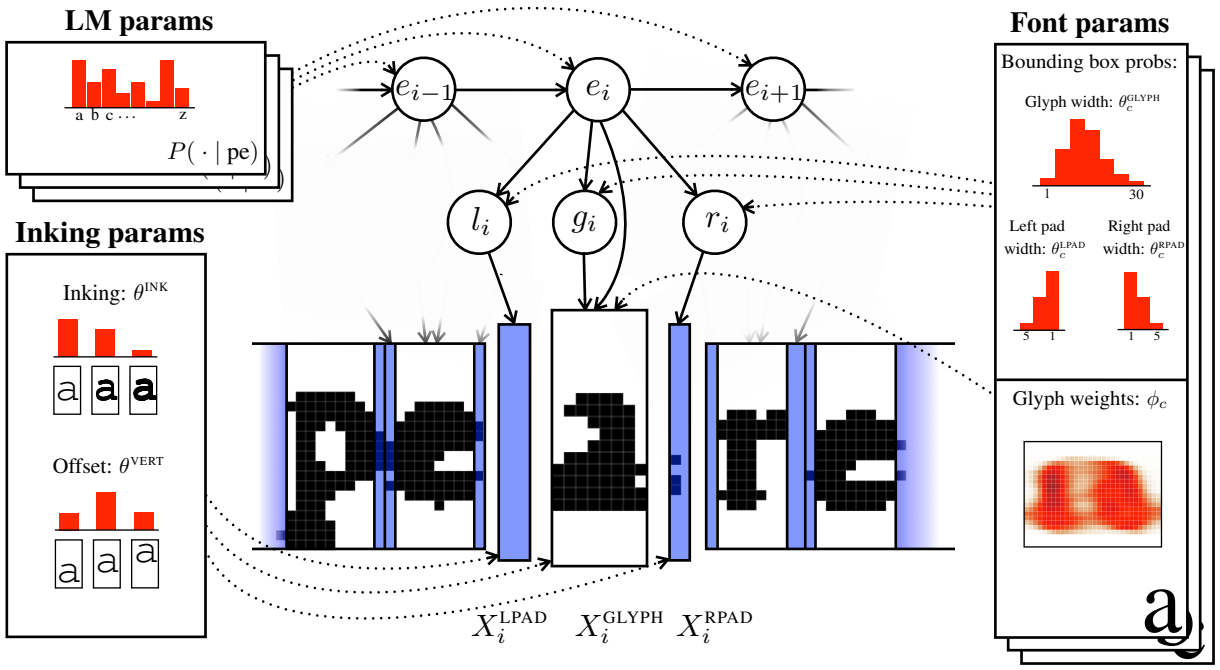


Figure 2.3: Character tokens e_i are generated by the language model. For each token index i, a glyph bounding box width g_i, left padding width l_i, and a right padding width r_i, are generated. Finally, the pixels in each glyph bounding box X_i^{GLYPH} are generated conditioned on the corresponding character, while the pixels in left and right padding bounding boxes, X_i^{LPAD} and X_i^{RPAD}, are generated from a background distribution.

Language Model $P(E)$

Our language model, $P(E)$, is a Kneser-Ney smoothed character n-gram model (Kneser and Ney 1995). We generate printed lines of text (rather than sentences) independently, without generating an explicit stop character. This means that, formally, the model must separately generate the character length of each line. We choose not to bias the model towards longer or shorter character sequences and let the line length m be drawn uniformly at random from the positive integers less than some large constant M.[1] When $i < 1$, let e_i denote a line-initial null character. We can now write:

$$P(E) = P(m) \cdot \prod_{i=1}^{m} P(e_i | e_{i-1}, \ldots, e_{i-n})$$

Typesetting Model $P(T|E)$

Generally speaking, the process of typesetting produces a line of text by first tiling bounding boxes of various widths and then filling in the boxes with glyphs. Our generative model,

[1] In particular, we do not use the kind of "word bonus" common to statistical machine translation models.

which is depicted in Figure 2.3, reflects this process. As a first step, our model generates the dimensions of character bounding boxes; for each character token index i we generate three bounding box widths: a glyph box width g_i, a left padding box width l_i, and a right padding box width r_i, as shown in Figure 2.3. We let the pixel height of all lines be fixed to h. Let $T_i = (l_i, g_i, r_i)$ so that T_i specifies the dimensions of the character box for token index i; T is then the concatenation of all T_i, denoting the full layout.

Because the width of a glyph depends on its shape, and because of effects resulting from kerning and the use of ligatures, the components of each T_i are drawn conditioned on the character token e_i. This means that, as part of our parameterization of the font, for each character type c we have vectors of multinomial parameters θ_c^{LPAD}, θ_c^{GLYPH}, and θ_c^{RPAD} governing the distribution of the dimensions of character boxes of type c. These parameters are depicted on the right-hand side of Figure 2.3. We can now express the typesetting layout portion of the model as:

$$\begin{aligned} P(T|E) &= \prod_{i=1}^{m} P(T_i|e_i) \\ &= \prod_{i=1}^{m} \left[P(l_i; \theta_{e_i}^{\text{LPAD}}) \cdot P(g_i; \theta_{e_i}^{\text{GLYPH}}) \cdot P(r_i; \theta_{e_i}^{\text{RPAD}}) \right] \end{aligned}$$

Each character type c in our font has another set of parameters, a matrix ϕ_c. These are weights that specify the *shape* of the character type's glyph, and are depicted in Figure 2.3 as part of the font parameters. ϕ_c will come into play when we begin generating pixels in Section 2.3.

Inking Model $P(R)$

Before we start filling the character boxes with pixels, we need to specify some properties of the inking and rendering process, including the amount of ink used and vertical variation along the text baseline. Our model does this by generating, for each character token index i, a discrete value d_i that specifies the overall inking level in the character's bounding box, and a discrete value v_i that specifies the glyph's vertical offset. These variations in the inking and typesetting process are mostly independent of character type. Thus, in our model, their distributions are not character-specific. There is one global set of multinomial parameters governing inking level (θ^{INK}), and another governing offset (θ^{VERT}); both are depicted on the left-hand side of Figure 2.3. Let $R_i = (d_i, v_i)$ and let R be the concatenation of all R_i so that we can express the inking model as:

$$\begin{aligned} P(R) &= \prod_{i=1}^{m} P(R_i) \\ &= \prod_{i=1}^{m} \left[P(d_i; \theta^{\text{INK}}) \cdot P(v_i; \theta^{\text{VERT}}) \right] \end{aligned}$$

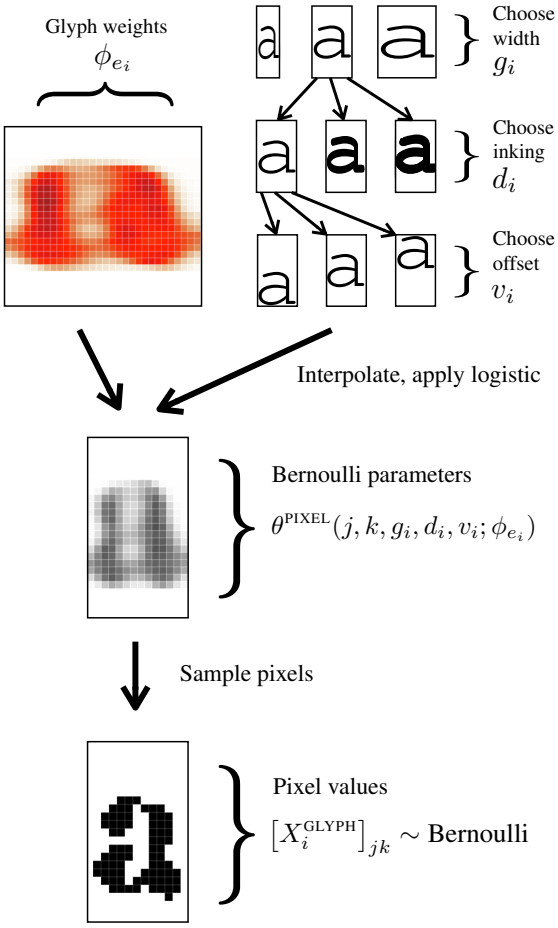


Figure 2.4: We generate the pixels for the character token e_i by first sampling a glyph width g_i, an inking level d_i, and a vertical offset v_i. Then we interpolate the glyph weights ϕ_{e_i} and apply the logistic function to produce a matrix of Bernoulli parameters of width g_i, inking d_i, and offset v_i. $\theta^{\text{PIXEL}}(j, k, g_i, d_i, v_i; \phi_{e_i})$ is the Bernoulli parameter at row j and column k. Finally, we sample from each Bernoulli distribution to generate a matrix of pixel values, X_i^{GLYPH}.

The d_i and v_i variables are suppressed in Figure 2.3 to reduce clutter but are expressed in Figure 2.4, which depicts the process of rendering a glyph box.

Noise Model $P(X|E, T, R)$

Now that we have generated a typesetting layout T and an inking context R, we have to actually generate each of the pixels in each of the character boxes, left padding boxes, and right padding boxes; the matrices that these groups of pixels comprise are denoted X_i^{GLYPH}, X_i^{LPAD}, and X_i^{RPAD}, respectively, and are depicted at the bottom of Figure 2.3.

We assume that pixels are binary valued and sample their values independently from

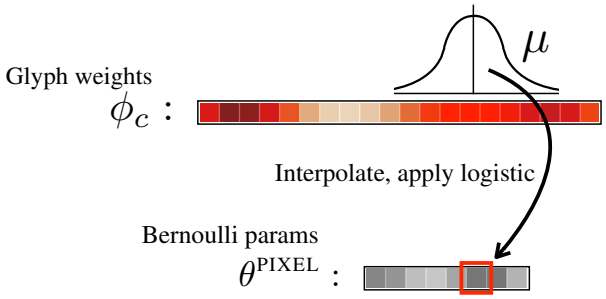


Figure 2.5: In order to produce Bernoulli parameter matrices θ^{PIXEL} of variable width, we interpolate over columns of ϕ_c with vectors μ, and apply the logistic function to each result.

Bernoulli distributions.[2] The probability of black (the Bernoulli parameter) depends on the type of pixel generated. All the pixels in a padding box have the same probability of black that depends only on the inking level of the box, d_i. Since we have already generated this value and the widths l_i and r_i of each padding box, we have enough information to generate left and right padding pixel matrices X_i^{LPAD} and X_i^{RPAD}.

The Bernoulli parameter of a pixel inside a glyph bounding box depends on the pixel's location inside the box (as well as on d_i and v_i, but for simplicity of exposition, we temporarily suppress this dependence) and on the model parameters governing glyph shape (for each character type c, the parameter matrix ϕ_c specifies the shape of the character's glyph.) The process by which glyph pixels are generated is depicted in Figure 2.4.

The dependence of glyph pixels on location complicates generation of the glyph pixel matrix X_i^{GLYPH} since the corresponding parameter matrix ϕ_{e_i} has some type-level width w which may differ from the current token-level width g_i. Introducing distinct parameters for each possible width would yield a model that can learn completely different glyph shapes for slightly different widths of the same character. We, instead, need a parameterization that ties the shapes for different widths together, and at the same time allows mobility in the parameter space during learning.

Our solution is to horizontally interpolate the weights of the shape parameter matrix ϕ_{e_i} down to a smaller set of columns matching the token-level choice of glyph width g_i. Thus, the type-level matrix ϕ_{e_i} specifies the canonical shape of the glyph for character e_i when it takes its maximum width w. After interpolating, we apply the logistic function to produce the individual Bernoulli parameters. If we let $[X_i^{\text{GLYPH}}]_{jk}$ denote the value of the pixel at the jth row and kth column of the glyph pixel matrix X_i^{GLYPH} for token i, and let $\theta^{\text{PIXEL}}(j, k, g_i; \phi_{e_i})$ denote the token-level Bernoulli parameter for this pixel, we can write:

$$[X_i^{\text{GLYPH}}]_{jk} \sim \text{Bernoulli}\big(\theta^{\text{PIXEL}}(j, k, g_i; \phi_{e_i})\big)$$

[2]We could generate real-valued pixels with a different choice of noise distribution.

The interpolation process for a single row is depicted in Figure 2.5. We define a constant interpolation vector $\mu(g_i, k)$ that is specific to the glyph box width g_i and glyph box column k. Each $\mu(g_i, k)$ is shaped according to a Gaussian centered at the relative column position in ϕ_{e_i}. The glyph pixel Bernoulli parameters are defined as follows:

$$\theta^{\text{PIXEL}}(j, k, g_i; \phi_{e_i}) = \text{logistic}\Big(\sum_{k'=1}^{w} \Big[\mu(g_i, k)_{k'} \cdot [\phi_{e_i}]_{jk'}\Big]\Big)$$

The fact that the parameterization is log-linear will ensure that, during the unsupervised learning process, updating the shape parameters ϕ_c is simple and feasible using methods described in Chapter 5.

By varying the magnitude of μ we can change the level of smoothing in the logistic model and cause it to permit areas that are over-inked. This is the effect that d_i controls. By offsetting the rows of ϕ_c that we interpolate weights from, we change the vertical offset of the glyph, which is controlled by v_i. The full pixel generation process is diagrammed in Figure 2.4, where the dependence of θ^{PIXEL} on d_i and v_i is also represented.

2.4 Learning

We use the EM algorithm (Dempster et al. 1977) to find the maximum-likelihood font parameters: ϕ_c, θ_c^{LPAD}, θ_c^{GLYPH}, and θ_c^{RPAD}. The image X is the only observed random variable in our model. The identities of the characters E the typesetting layout T and the inking R will all be unobserved. We do not learn θ^{INK} and θ^{VERT}, which are set to the uniform distribution.

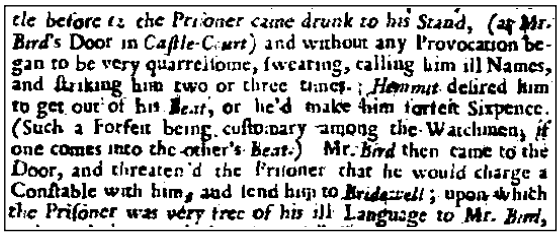
Expectation Maximization

During the E-step we compute expected counts for E and T, but maximize over R, for which we compute hard counts. Our model is an instance of a hidden semi-Markov model (HSMM), and therefore the computation of marginals is tractable with the semi-Markov forward-backward algorithm (Levinson 1986).

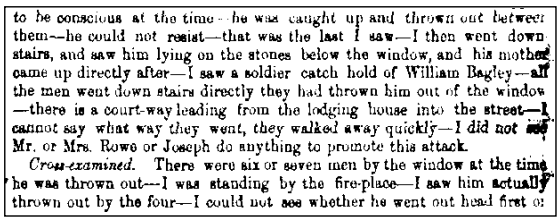
During the M-step, we update the parameters θ_c^{LPAD}, θ_c^{RPAD} using the standard closed-form multinomial updates and use a specialized closed-form update for θ_c^{GLYPH} that enforces unimodality of the glyph width distribution.[3] The glyph weights, ϕ_c, do not have a closed-form update. The noise model that ϕ_c parameterizes is a local log-linear model, so we use L-BFGS (Liu and Nocedal 1989) to optimize the expected likelihood with respect to ϕ_c. This approach is described in more detail in Chapter 5.

[3]We compute the weighted mean and weighted variance of the glyph width expected counts. We set θ_c^{GLYPH} to be proportional to a discretized Gaussian with the computed mean and variance. This update is approximate in the sense that it does not necessarily find the unimodal multinomial that maximizes expected log-likelihood, but it works well in practice.

(a) Old Bailey, 1725:

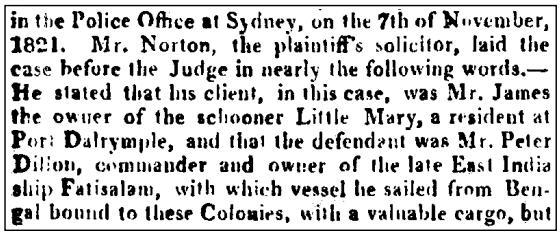

tle before the Prisoner came drunk to his Stand, (at Mr. Bird's Door in Castle-Court) and without any Provocation began to be very quarrelsome, swearing, calling him ill Names, and striking him two or three times; Hemmit desired him to get out of his Beat, or he'd make him forfeit Sixpence. (Such a Forfeit being customary among the Watchmen, if one comes into the other's Beat.) Mr. Bird then came to the Door, and threaten'd the Prisoner that he would charge a Constable with him, and send him to Bridewell; upon which the Prisoner was very free of his ill Language to Mr. Bird,

(b) Old Bailey, 1875:

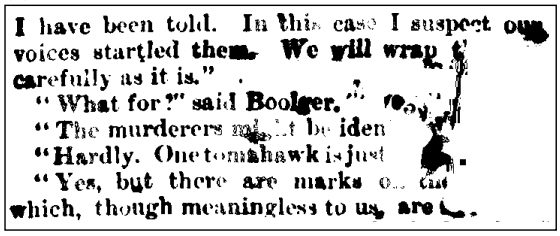

to be conscious at the time he was caught up and thrown out between them—he could not resist—that was the last I saw—I then went down stairs, and saw him lying on the stones below the window, and his mother came up directly after—I saw a soldier catch hold of William Bagley—all the men went down stairs directly they had thrown him out of the window—there is a court-way leading from the lodging house into the street—I cannot say what way they went, they walked away quickly—I did not see Mr. or Mrs. Rowe or Joseph do anything to promote this attack.

Cross-examined. There were six or seven men by the window at the time he was thrown out—I was standing by the fire-place—I saw him actually thrown out by the four—I could not see whether he went out head first o

(c) Trove, 1823:


in the Police Office at Sydney, on the 7th of November, 1821. Mr. Norton, the plaintiff's solicitor, laid the case before the Judge in nearly the following words.—He stated that his client, in this case, was Mr. James the owner of the schooner Little Mary, a resident at Port Dalrymple, and that the defendant was Mr. Peter Dillon, commander and owner of the late East India ship Fatisalam, with which vessel he sailed from Bengal bound to these Colonies, with a valuable cargo, but

(d) Trove, 1883:


I have been told. In this case I suspect our voices startled them. We will wrap it carefully as it is."

"What for?" said Boolger.

"The murderers would be iden

"Hardly. One tomahawk is just

"Yes, but there are marks on the which, though meaningless to us, are

Figure 2.6: Portions of several documents from our test set representing a range of difficulties are displayed. On document (a), which exhibits noisy typesetting, our system achieves a word error rate (WER) of 25.2. Document (b) is cleaner in comparison, and on it we achieve a WER of 15.4. On document (c), which is also relatively clean, we achieve a WER of 12.5. On document (d), which is severely degraded, we achieve a WER of 70.0.

Coarse-to-Fine Learning and Inference

The number of states in the dynamic programming lattice grows exponentially with the order of the language model (Jelinek 1998; Koehn 2004). As a result, inference can become slow when the language model order n is large. To remedy this, we take a coarse-to-fine approach to both learning and inference. On each iteration of EM, we perform two passes: a coarse pass using a low-order language model, and a fine pass using a high-order language model (Petrov et al. 2008; Zhang and Gildea 2008). We use the marginals[4] from the coarse pass to prune states from the dynamic program of the fine pass.

In the early iterations of EM, our font parameters are still inaccurate, and to prune heavily based on such parameters would rule out correct analyses. Therefore, we gradually increase the aggressiveness of pruning over the course of EM. To ensure that each iteration takes approximately the same amount of computation, we also gradually increase the order of the fine pass, only reaching the full order n on the last iteration. To produce a decoding of the image into text, on the final iteration we run a Viterbi pass using the pruned fine model.

[4]In practice, we use max-marginals for pruning to ensure that there is still a valid path in the pruned lattice.

2.5 Data

We perform experiments on two historical datasets consisting of images of documents printed between 1700 and 1900 in England and Australia. Examples from both datasets are displayed in Figure 2.6.

Old Bailey

The first dataset comes from a large set of images of the proceedings of the Old Bailey, a criminal court in London, England (Shoemaker 2005). The Old Bailey curatorial effort, after deciding that current OCR systems do not adequately handle 18th century fonts, manually transcribed the documents into text. We will use these manual transcriptions to evaluate the output of our system. From the Old Bailey proceedings, we extracted a set of 20 images, each consisting of 30 lines of text to use as our first test set. We picked 20 documents, printed in consecutive decades. The first document is from 1715 and the last is from 1905. We choose the first document in each of the corresponding years, choose a random page in the document, and extracted an image of the first 30 consecutive lines of text consisting of full sentences.[5] The ten documents in the Old Bailey dataset that were printed before 1810 use the long `s` glyph, while the remaining ten do not.

Trove

Our second dataset is taken from a collection of digitized Australian newspapers that were printed between the years of 1803 and 1954. This collection is called Trove, and is maintained by the the National Library of Australia (Holley 2010). We extracted ten images from this collection in the same way that we extracted images from Old Bailey, but starting from the year 1803. We manually produced our own gold annotations for these ten images. Only the first document of Trove uses the long `s` glyph.

Pre-processing

Many of the images in historical collections are bitonal (binary) as a result of how they were captured on microfilm for storage in the 1980s (Arlitsch and Herbert 2004). This is part of the reason our model is designed to work directly with binarized images. For consistency, we binarized the images in our test sets that were not already binary by thresholding pixel values.

Our model requires that the image be pre-segmented into lines of text. We automatically segment lines by training an HSMM over rows of pixels. After the lines are segmented, each line is resampled so that its vertical resolution is 30 pixels. The line extraction process

[5] This ruled out portions of the document with extreme structural abnormalities, like title pages and lists. These might be interesting to model, but are not within the scope of this thesis.

also identifies pixels that are not located in central text regions, and are part of large connected components of ink, spanning multiple lines. The values of such pixels are treated as unobserved in the model since, more often than not, they are part of ink blotches.

2.6 Experiments

We evaluate our system by comparing our text recognition accuracy to that of two state-of-the-art systems.

Baselines

Our first baseline is Google's open source OCR system, Tesseract (R. Smith 2007). Tesseract takes a pipelined approach to recognition. Before recognizing the text, the document is broken into lines, and each line is segmented into words. Then, Tesseract uses a classifier, aided by a word-unigram language model, to recognize whole words.

Our second baseline, ABBYY FineReader 11 Professional Edition,[6] is a state-of-the-art commercial OCR system. It is the OCR system that the National Library of Australia used to recognize the historical documents in Trove (Holley 2010).

Evaluation

We evaluate the output of our system and the baseline systems using two metrics: character error rate (CER) and word error rate (WER). Both these metrics are based on edit distance. CER is the edit distance between the predicted and gold transcriptions of the document, divided by the number of characters in the gold transcription. WER is the word-level edit distance (words, instead of characters, are treated as tokens) between predicted and gold transcriptions, divided by the number of words in the gold transcription. When computing WER, text is tokenized into words by splitting on whitespace.

Language Model

We ran experiments using two different language models. The first language model was trained on the initial one million sentences of the New York Times (NYT) portion of the Gigaword corpus (Graff et al. 2007), which contains about 36 million words. This language model is out of domain for our experimental documents. To investigate the effects of using an in domain language model, we created a corpus composed of the manual annotations of all the documents in the Old Bailey proceedings, excluding those used in our test set. This corpus consists of approximately 32 million words. In all experiments we used a character n-gram order of six for the final Viterbi decoding pass and an order of three for all coarse passes.

[6] http://www.abbyy.com

System	CER	WER
Old Bailey		
Google Tesseract	29.6	54.8
ABBYY FineReader	15.1	40.0
Ocular w/ NYT (this work)	12.6	28.1
Ocular w/ OB (this work)	**9.7**	**24.1**
Trove		
Google Tesseract	37.5	59.3
ABBYY FineReader	22.9	49.2
Ocular w/ NYT (this work)	**14.9**	**33.0**

Table 2.1: We evaluate the predicted transcriptions in terms of both character error rate (CER) and word error rate (WER), and report macro-averages across documents. We compare with two baseline systems: Google's open source OCR system, Tesseract, and a state-of-the-art commercial system, ABBYY FineReader. We refer to our system as Ocular w/ NYT and Ocular w/ OB, depending on whether NYT or Old Bailey is used to train the language model.

Initialization and Tuning

We used as a development set ten additional documents from the Old Bailey proceedings and five additional documents from Trove that were not part of our test set. On this data, we tuned the model's hyperparameters[7] and the parameters of the pruning schedule for our coarse-to-fine approach.

In experiments we initialized θ_c^{RPAD} and θ_c^{LPAD} to be uniform, and initialized θ_c^{GLYPH} and ϕ_c based on the standard modern fonts included with the Ubuntu Linux 12.04 distribution.[8] For documents that use the long `s` glyph, we introduce a special character type for the non-word-final `s`, and initialize its parameters from a mixture of the modern `f` and `|` glyphs. As described in Chapter 5, we use a regularization term in the optimization of the log-linear model parameters ϕ_c during the M-step. Instead of regularizing towards zero, we regularize towards the initializer. This slightly improves performance on our development set and can be thought of as placing a prior on the glyph shape parameters.

2.7 Results and Analysis

The results of our experiments are summarized in Table 2.1. We refer to our system as Ocular w/ NYT or Ocular w/ OB, depending on whether the language model was trained using NYT or Old Bailey, respectively. We compute macro-averages across documents from

[7]One of the hyperparameters we tune is the exponent of the language model. This balances the contributions of the language model and the typesetting model to the posterior (Och and Ney 2004).

[8]http://www.ubuntu.com/

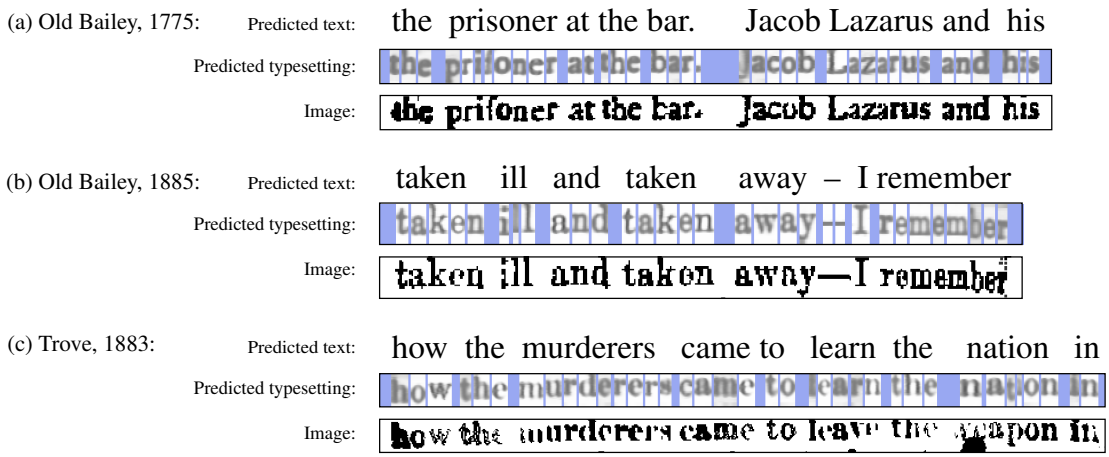


Figure 2.7: For each of these portions of test documents, the first line shows the transcription predicted by our model and the second line shows a representation of the learned typesetting layout. The grayscale glyphs show the Bernoulli pixel distributions learned by our model, while the padding regions are depicted in blue. The third line shows the input image.

all years. Our system, using the NYT language model, achieves an average WER of 28.1 on Old Bailey and an average WER of 33.0 on Trove. This represents a substantial error reduction compared to both baseline systems.

If we average over the documents in both Old Bailey and Trove, we find that Tesseract achieved an average WER of 56.3, ABBYY FineReader achieved an average WER of 43.1, and our system, using the NYT language model, achieved an average WER of 29.7. This means that while Tesseract incorrectly predicts more than half of the words in these documents, our system gets more than three-quarters of them right. Overall, we achieve a relative reduction in WER of 47% compared to Tesseract and 31% compared to ABBYY FineReader.

The baseline systems do not have special provisions for the long `s` glyph. In order to make sure the comparison is fair, we separately computed average WER on only the documents from after 1810 (which do no use the long `s` glyph). We found that using this evaluation our system actually acheives a larger relative reduction in WER: 50% compared to Tesseract and 35% compared to ABBYY FineReader.

Finally, if we train the language model using the Old Bailey corpus instead of the NYT corpus, we see an average improvement of 4 WER on the Old Bailey test set. This means that the domain of the language model is important, but, the results are not affected drastically even when using a language model based on modern corpora (NYT).

Learned Typesetting Layout

Figure 2.7 shows a representation of the typesetting layout learned by our model for portions of several test documents. For each portion of a test document, the first line shows

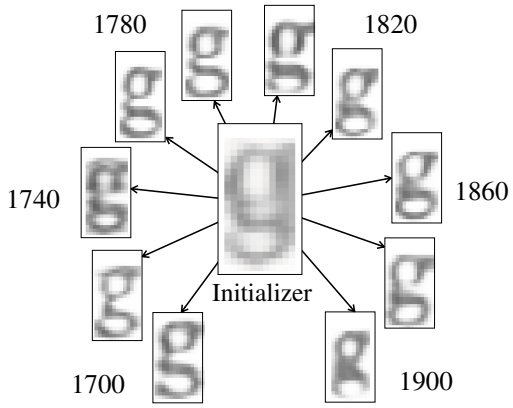


Figure 2.8: The central glyph is a representation of the initial model parameters for the glyph shape for g, and surrounding this are the learned parameters for documents from various years.

the transcription predicted by our model, and the second line shows padding and glyph regions predicted by the model, where the grayscale glyphs represent the learned Bernoulli parameters for each pixel. The third line shows the input image.

Figure 2.7a demonstrates a case where our model has effectively explained both the uneven baseline and over-inked glyphs by using the vertical offsets v_i and inking variables d_i. In Figure 2.7b the model has used glyph widths g_i and vertical offsets to explain the thinning of glyphs and falling baseline that occurred near the binding of the book. In separate experiments on the Old Bailey test set, using the NYT language model, we found that removing the vertical offset variables from the model increased WER by 22, and removing the inking variables increased WER by 16. This indicates that it is very important to model both these aspects of printing press rendering.

Figure 2.7c shows the output of our system on a difficult document. Here, missing characters and ink blotches confuse the model, which picks something that is reasonable according to the language model, but incorrect.

Learned Fonts

It is interesting to look at the fonts learned by our system, and track how historical fonts changed over time. Figure 2.8 shows several grayscale images representing the Bernoulli pixel probabilities for the most likely width of the glyph for g under various conditions. At the center is the representation of the initial parameter values, and surrounding this are the learned parameters for documents from various years. The learned shapes are visibly different from the initializer, which is essentially an average of modern fonts, and also vary across decades.

We can ask to what extent learning the font structure actually improved our performance. If we turn off learning and just use the initial parameters to decode, WER increases by 8 on the Old Bailey test set when using the NYT language model.

Figure 2.9: This Old Bailey document from 1719 has severe ink bleeding from the facing page. We annotated these blotches (in red) and treated the corresponding pixels as unobserved in the model. The layout shown is predicted by the model.

Unobserved Ink Blotches

As noted earlier, one strength of our generative model is that we can make the values of certain pixels unobserved in the model, and let inference fill them in. We conducted an additional experiment on a document from the Old Bailey proceedings that was printed in 1719. This document, a fragment of which is shown in Figure 2.9, has severe ink bleeding from the facing page. We manually annotated the ink blotches (shown in red), and made them unobserved in the model. The resulting typesetting layout learned by the model is also shown in Figure 2.9. The model correctly predicted most of the obscured words. Running the model with the manually specified unobserved pixels reduced the WER on this document from 58 to 19 when using the NYT language model.

Error Analysis

We performed error analysis on our development set by randomly choosing 100 word errors from the WER alignment and manually annotating them with relevant features. Specifically, for each word error we recorded whether or not the error contained punctuation (either in the predicted word or the gold word), whether the text in the corresponding portion of the original image was italicized, and whether the corresponding portion of the image exhibited over-inking, missing ink, or significant ink blotches. These last three feature types are subjective in nature but may still be informative. We found that 56% of errors were accompanied by over-inking, 50% of errors were accompanied by ink blotches, 42% of errors contained punctuation, 21% of errors showed missing ink, and 12% of errors contained text that was italicized in the original image.

Our own subjective assessment indicates that many of these error features are in fact causal. More often than not, italicized text is incorrectly transcribed. In cases of extreme ink blotching, or large areas of missing ink, the system usually makes an error. In Chapter 3 we address some of these problems by incorporating a richer typesetting process into our generative model of the printing press.

Chapter 3

Richer Typesetting Models

3.1 Overview

In Chapter 2 we proposed a system for historical OCR that generatively models the noisy typesetting process of printing-press era documents and learns the font for each input document in an unsupervised fashion. The system achieves state-of-the-art results on the task of historical document recognition. In this chapter, we take the original system from Chapter 2 as a starting point and consider extensions of the typesetting model that address two shortcomings: (1) the original layout model assumed that baseline offset noise is independent for each glyph and (2) the original font model assumed a single font is used in every document. Both of these assumptions are untrue in many historical datasets. In this chapter, by extending the model so that it more accurately reflect the true typesetting process, we aim to make unsupervised transcription work better.

As described in Chapter 2, the baseline of the text in printing-press era documents is not rigid as in modern documents but rather drifts up and down noisily (see Figure 3.2). In practice, the vertical offsets of character glyphs change gradually along a line. This means the vertical offsets of neighboring glyphs are correlated, a relationship that is not captured by the original model. In our first extension, we let the vertical offsets of character glyphs be generated from a Markov chain, penalizing large changes in offset. We find that this extension decreases transcription error rates. Our system achieves a relative word error reduction of 22% compared to the state-of-the-art original model on a test set of historical newspapers (see Section 3.4), and a 11% relative reduction on a test set of historical court proceedings.

Multiple font styles are also frequently used in printing-press era documents; the most common scenario is for a basic font style to co-occur with an italic variant. For example, it is common for proper nouns and quotations to be italicized in the Old Bailey corpus (Shoemaker 2005). In our second extension, we incorporate a Markov chain over font styles, extending the original model so that it is capable of simultaneously learning italic and non-italic fonts within a single document. In experiments, this model is able to detect which words are

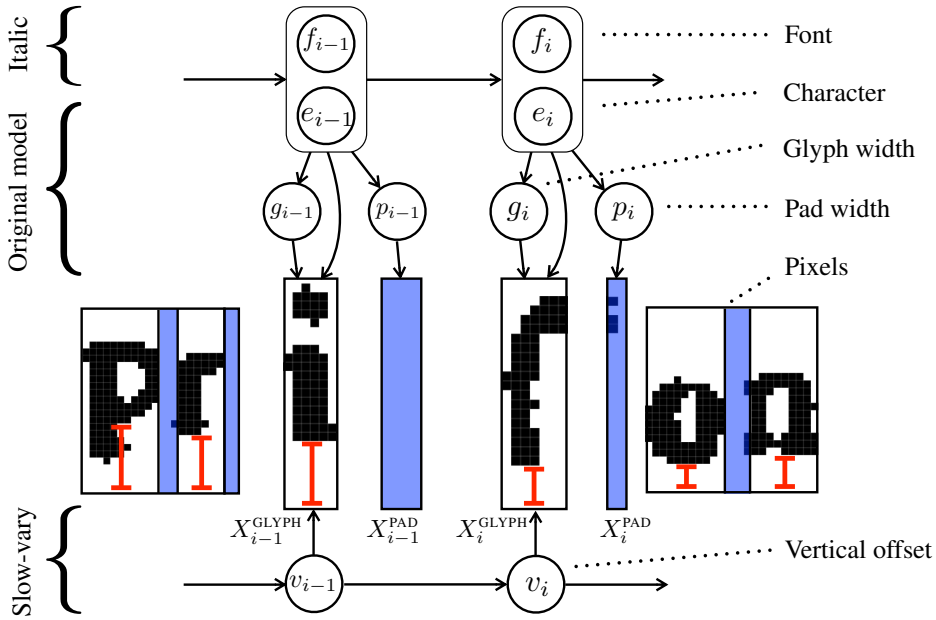


Figure 3.1: See Section 3.2 for a description of the generative process. We consider an extended model that generates v_i conditioned on the previous vertical offset v_{i-1} (labeled Slow-vary) and generates a sequence of font styles f_i (labeled Italic).

italicized with 93% precision at 74% recall in a test set of historical court proceedings (see Section 3.4).

These richer models that we propose do increase the state space and therefore make inference more costly. To remedy this, we streamline inference by replacing the coarse-to-fine inference scheme described in Chapter 2 with a forward-cost-augmented beaming scheme. Our method is over 25x faster on a typical document, yet actually yields *improved* transcriptions.

3.2 Extended Model

We first describe the generative model from Chapter 2 using new notation that will simplify our description of the extensions. While the original model generated two pad boxes for each character token, one to the left and one to the right, our baseline model in this chapter only generates one pad box, always to the right. The graphical model corresponding to the basic generative process for a single line of text is diagrammed in Figure 3.1, where we also diagram our extensions. In the basic model, a Kneser-Ney (Kneser and Ney 1995) character 6-gram language model generates a sequence of characters $E = (e_1, e_2, \ldots, e_n)$. For each character index i, a glyph box width g_i and a pad box width p_i are generated, conditioned on the character e_i. g_i specifies the width of the bounding box that will eventually house the pixels of the glyph for character e_i. p_i specifies the width of a padding box which contains

the horizontal space before the next character begins. Next, a vertical offset v_i is generated for the glyph corresponding to character e_i. v_i allows the model to capture variance in the baseline of the text in the document. We will later let v_i depend on v_{i-1}, as depicted in Figure 3.1, but in the original system they are independent. Finally, the pixels in the ith glyph bounding box X_i^{GLYPH} are generated conditioned on the character e_i, width g_i, and vertical offset v_i, and the pixels in the ith pad bounding box X_i^{PAD} are generated conditioned on the width p_i. In this chapter we have omitted the token-level inking random variables for simplicity. These can be treated as part of the pixel generation process.

Let X denote the matrix of pixels for the entire line, $V = (v_1, \ldots, v_n)$, $P = (p_1, \ldots, p_n)$, and $G = (g_1, \ldots, g_n)$. The joint distribution is written:

$$
\begin{aligned}
&P(X,V,P,G,E) = \\
&\quad P(E) && \text{[Language model]} \\
&\quad \cdot \prod_{i=1}^{n} P(g_i|e_i;\Phi) && \text{[Glyph widths]} \\
&\quad \cdot \prod_{i=1}^{n} P(p_i|e_i;\Phi) && \text{[Pad widths]} \\
&\quad \cdot \prod_{i=1}^{n} P(v_i) && \text{[Vertical offsets]} \\
&\quad \cdot \prod_{i=1}^{n} P(X_i^{\text{PAD}}|p_i) && \text{[Pad pixels]} \\
&\quad \cdot \prod_{i=1}^{n} P(X_i^{\text{GLYPH}}|v_i,g_i,e_i;\Phi) && \text{[Glyph pixels]}
\end{aligned}
$$

The font is parameterized by the vector Φ which governs the shapes of glyphs and the distributions over box widths. Φ is learned in an unsupervised fashion. Document recognition is accomplished via Viterbi decoding over the character random variables e_i.

Slow-varying Offsets

The original model generates the vertical offsets v_i independently, and therefore cannot model how neighboring offsets are correlated. This correlation is actually strong in printing-press era documents. The baseline of the text wanders in the input image for two reasons: (1) the physical groove along which character templates were set was uneven and (2) the original document was imaged in a way that produced distortion. Both these underlying causes are likely to yield baselines that wander slowly up and down across a document. We refer to this behavior of vertical offsets as slow-varying, and extend the model to capture it.

In our first extension, we augment the model by incorporating a Markov chain over the vertical offset random variables v_i, as depicted in Figure 3.1. Specifically, v_i is generated

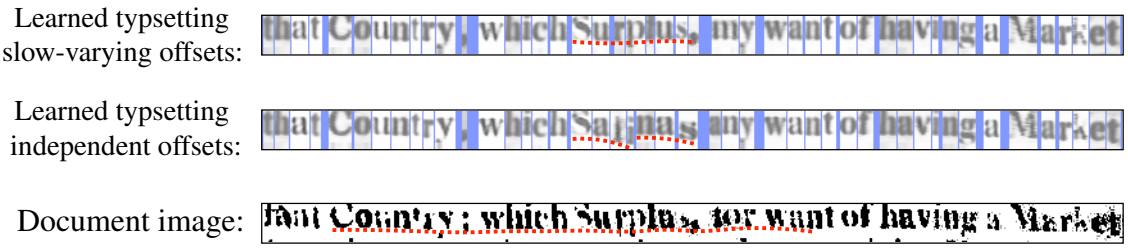


Figure 3.2: The first line depicts the Viterbi typesetting layout predicted by the OCULAR-BEAM-SV model. The second line depicts the same, but for the OCULAR-BEAM model. Pad boxes are shown in blue. Glyphs boxes are shown in white and display the Bernoulli template probabilities used to generate the observed pixels. The third line shows the corresponding portion of the input image.

from a discretized Gaussian centered at v_{i-1}:

$$P(v_i|v_{i-1}) \propto \exp\left(\frac{(v_i - v_{i-1})^2}{2\sigma^2}\right)$$

This means that the if v_i differs substantially from v_{i-1}, a large penalty is incurred. As a result, the model should prefer sequences of v_i that vary slowly. In experiments, we set $\sigma^2 = 0.05$.

Italic Font Styles

Many of the documents in the Old Bailey corpus contain both italic and non-italic font styles (Shoemaker 2005). The way that italic fonts are used depends on the year the document was printed, but generally italics are reserved for proper nouns, quotations, and sentences that have a special role (e.g. the final judgment made in a court case). The switch between font styles almost always occurs at space characters.

Our second extension of the typesetting model deals with both italic and non-italic font styles. We augment the model with a Markov chain over font styles f_i, as depicted in Figure 3.1. Each font style token f_i takes on a value in {ITALIC, NON-ITALIC} and is generated conditioned on the previous font style f_{i-1} and the current character token e_i. Specifically, after generating a character token that is not a space, the language model deterministically generates the last font used. If the language model generates a space character token, the decision of whether to switch font styles is drawn from a Bernoulli distribution. This ensures that the font style only changes at space characters.

The font parameters Φ are extended to contain entries for the italic versions of all characters. This means the shapes and widths of italic glyphs can be learned separately from non-italic ones. As in Chapter 2, we initialize the font parameters from mixtures of modern fonts, using mixtures of modern italic font styles for italic characters.

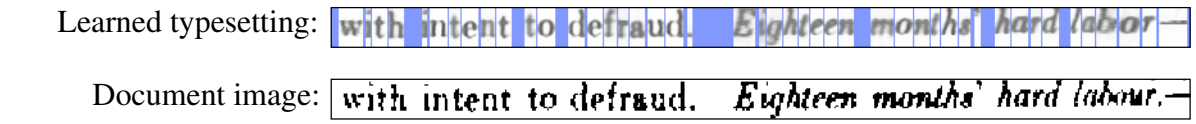


Figure 3.3: This first line depicts the Viterbi typesetting layout predicted by the OCULAR-BEAM-IT model. Pad boxes are shown in blue. Glyphs boxes are shown in white and display the Bernoulli template probabilities used to generate the observed pixels. The second line shows the corresponding portion of the input image.

3.3 Streamlined Inference

Inference in our extended typesetting models is costly because the state space is large; we propose an new inference procedure that is fast and simple.

In Chapter 2 we used EM to learn the font parameters Φ, and therefore required expected sufficient statistics (indicators on (e_i, g_i, v_i) tuples), which were computed using coarse-to-fine inference (Petrov et al. 2008; Zhang and Gildea 2008) with a semi-Markov dynamic program (Levinson 1986). This approach is effective, but slow. For example, while transcribing a typical document consisting of 30 lines of text, the original system spends 63 minutes computing expected sufficient statistics and decoding when run on a 4.5GHz 4-core CPU.

In this chapter, we instead use hard counts of the sufficient statistics for learning (i.e. perform hard-EM). As a result, we are free to use inference procedures that are specialized for Viterbi computation. Specifically, we use beam-search with estimated forward costs. Because the model is semi-Markov, our beam-search procedure is very similar the one used by Pharaoh (Koehn 2004) for phrase-based machine translation, only without a distortion model. We use a beam of size 20, and estimate forward costs using a character bigram language model. On the machine mentioned above, transcribing the same document, our simplified system that uses hard-EM and beam-search spends only 2.4 minutes computing sufficient statistics and decoding. This represents a 26x speedup.

3.4 Results

We ran experiments with four different systems. The first is our baseline, the system presented in Chapter 2, which we refer to as OCULAR. The second system uses the original model, but uses beam-search for inference. We refer to this system as OCULAR-BEAM. The final two systems use beam-search for inference, but use extended models: OCULAR-BEAM-SV uses the slow-varying vertical offset extension described in Section 3.2 and OCULAR-BEAM-IT uses the italic font extension described in Section 3.2.

We evaluate on two different test sets of historical documents. The first test set, called Trove, was also used in Chapter 2 for evaluation. Trove consists of 10 documents that were printed between 1803 and 1954, each consisting of 30 lines, all taken from a collection of historical Australian newspapers hosted by the National Library of Australia (Holley 2010).

System	CER	WER
Trove		
Google Tesseract	37.5	59.3
ABBYY FineReader	22.9	49.2
Ocular (baseline)	14.9	33.0
Ocular-Beam	12.9	30.7
Ocular-Beam-SV	**11.2**	**25.6**
Old Bailey 2		
Ocular (baseline)	14.9	30.8
Ocular-Beam	10.9	28.8
Ocular-Beam-SV	**10.3**	**27.5**

Table 3.1: We evaluate the output of each system on two test sets: Trove, a collection of historical newspapers, and Old Bailey 2, a collection of historical court proceedings. We report character error rate (CER) and word error rate (WER), macro-averaged across documents.

The second test set consists of 20 documents that were printed between 1716 and 1906, each consisting of 30 lines, all taken from the proceedings of the Old Bailey Courthouse in London (Shoemaker 2005). This test set, which we refer to here as Old Bailey 2, is comparable to the the second test set used in Chapter 2 since it is derived from the same collection and covers a similar time span, but it consists of different documents. As in Chapter 2, we train the language model using 36 millions words from the New York Times portion of the Gigaword corpus (Graff et al. 2007). This means the language model is out-of-domain on both test sets. In Chapter 2, we also considered an in-domain language model, though this setting was somewhat unrealistic.

Document Recognition Performance

We evaluate predicted transcriptions using both character error rate (CER) and word error rate (WER). CER is the edit distance between the guessed transcription and the gold transcription, divided by the number of characters in the gold transcription. WER is computed in the same way, but words are treated as tokens instead of characters.

First we compare the baseline, Ocular, to our system with simplified inference, Ocular-Beam. To our surprise, we found that Ocular-Beam produced better transcriptions than Ocular. On Trove, Ocular achieved a WER of 33.0 while Ocular-Beam achieved a WER of 30.7. On Old Bailey 2, Ocular achieved a WER of 30.8 while Ocular-Beam achieved a WER of 28.8. These results are shown in Table 3.1, where we also report the performance of Google Tesseract (R. Smith 2007) and ABBYY FineReader, a state-of-the-art commercial system, on the Trove test set.

Next, we evaluate our slow-varying vertical offset model. Ocular-Beam-SV out-

performs Ocular-Beam on both test sets. On Trove, Ocular-Beam-SV achieved a WER of 25.6, and on Old Bailey 2, Ocular-Beam-SV achieved a WER of 27.5. Overall, compared to our baseline system, Ocular-Beam-SV achieved a relative reduction in WER of 22% on Trove and 11% on Old Bailey 2.

By looking at the predicted typesetting layouts we can make a qualitative comparison between the vertical offsets predicted by Ocular-Beam and Ocular-Beam-SV. Figure 3.2 shows representations of the Viterbi estimates of the typesetting random variables predicted by the models on a portion of an example document. The first line is the typesetting layout predicted by Ocular-Beam-SV and the second line is same, but for Ocular-Beam. The locations of padding boxes are depicted in blue. The white glyph bounding boxes reveal the values of the Bernoulli template probabilities used to generate the observed pixels. The Bernoulli templates are produced from type-level font parameters, but are modulated by token-level widths g_i and vertical offsets v_i (and inking random variables, whose description we have omitted for brevity). The predicted vertical offsets are visible in the shifted baselines of the template probabilities. The third line shows the corresponding portion of the input image. In this example, the text baseline predicted by Ocular-Beam-SV is contiguous, while the one predicted by Ocular-Beam is not. Given how Ocular-Beam-SV was designed, this meets our expectations. The text baseline predicted by Ocular-Beam has a discontinuity in the middle of its prediction for the gold word `Surplus`. In contrast, the vertical offsets predicted by Ocular-Beam-SV at this location vary smoothly and more accurately match the true text baseline in the input image.

Font Detection Performance

We ran experiments with the italic font style model, Ocular-Beam-IT, on the Old Bailey 2 test set (italics are infrequent in Trove). We evaluated the learned styles by measuring how accurately Ocular-Beam-IT was able to distinguish between italic and non-italic styles. Specifically, we computed the precision and recall for the system's predictions about which words were italicized. We found that, across the entire Old Bailey 2 test set, Ocular-Beam-IT was able to detect which words were italicized with 93% precision at 74% recall, suggesting that the system did successfully learn both italic and non-italic styles. While it seems plausible that learning italics could also improve transcription accuracy, we found that Ocular-Beam-IT actually performed slightly worse than Ocular-Beam. This negative result is possibly due to the fact that the model was only trained on 30 lines of text at a time. While this appears to be enough data to a learn a standard font, the model may not encounter a sufficient number of italic characters (which appear less frequently than standard characters) to accurately learn their shapes.

We can look at the typesetting layout predicted by Ocular-Beam-IT to gain insight into what has been learned by the model. The first line of Figure 3.3 shows the typesetting layout predicted by the Ocular-Beam-IT model for a line of a document image that contains italics. The second line of Figure 3.3 displays the corresponding portion of the input document image. From this example, it appears that the model has effectively learned

separate glyph shapes for italic and non-italic versions of certain characters. For example, compare the template probabilities used to generate the `d`'s in `defraud` to the template probabilities used to generate the `d` in `hard`.

Chapter 4

Transcription of Piano Music

4.1 Overview

In this chapter we present an unsupervised method for transcription of an entirely different category of human data: music. As in the last two chapters, we construct a generative model of the underlying process that generated our data—in this case, that means modeling the piano. Before giving a detailed description of our approach, we provide some background on this task.

Automatic music transcription is the task of transcribing a musical audio signal into a symbolic representation (for example MIDI or sheet music). We focus on the task of transcribing piano music, which is potentially useful for a variety of applications ranging from information retrieval to musicology. This task is extremely difficult for multiple reasons. First, even individual piano notes are quite rich. A single note is not simply a fixed-duration sine wave at an appropriate frequency, but rather a full spectrum of harmonics that rises and falls in intensity. These profiles vary from piano to piano, and therefore must be learned in a recording-specific way. Second, piano music is generally *polyphonic*, i.e. multiple notes are played simultaneously. As a result, the harmonics of the individual notes can and do collide. In fact, combinations of notes that exhibit ambiguous harmonic collisions are particularly common in music, because consonances sound pleasing to listeners. This polyphony creates a source-separation problem at the heart of the transcription task.

In our approach, we learn the timbral properties of the piano being transcribed (i.e. the spectral and temporal shapes of each note) in an unsupervised fashion, directly from the input acoustic signal. We present a novel probabilistic model that describes the process by which discrete musical events give rise to (separate) acoustic signals for each keyboard note, and the process by which these signals are superimposed to produce the observed data. Inference over the latent variables in the model yields transcriptions that satisfy an informative prior distribution on the discrete musical structure and at the same time resolve the source-separation problem.

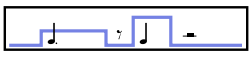
For the problem of unsupervised piano transcription where the test instrument is not

seen during training, the classic starting point is a non-negative factorization of the acoustic signal's spectrogram. Most previous work improves on this baseline in one of two ways: either by better modeling the discrete musical structure of the piece being transcribed (Nakano et al. 2012; Benetos and Weyde 2013) or by better adapting to the timbral properties of the source instrument (Vincent et al. 2010; O'Hanlon and Plumbley 2014). Combining these two kinds of approaches has proven challenging. The standard approach to modeling discrete musical structures—using hidden Markov or semi-Markov models—relies on the availability of fast dynamic programs for inference. Here, coupling these discrete models with timbral adaptation and source separation breaks the conditional independence assumptions that the dynamic programs rely on. In order to avoid this inference problem, past approaches typically defer detailed modeling of discrete structure or timbre to a postprocessing step (Poliner and Ellis 2006; Boogaart and Lienhart 2009; Weninger et al. 2013).

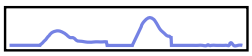
We present the first approach that tackles these discrete and timbral modeling problems jointly. We have two primary contributions: first, a new generative model that reflects the causal process underlying piano sound generation in an articulated way, starting with discrete musical structure; second, a tractable approximation to the inference problem over transcriptions and timbral parameters. Our approach surpasses state-of-the-art results on the task of polyphonic piano music transcription. On a standard dataset consisting of real piano audio data, annotated with ground-truth onsets, our approach outperforms the best published models for this task on multiple metrics, achieving a 10.6% relative gain in our primary measure of note onset F_1.

4.2 Model

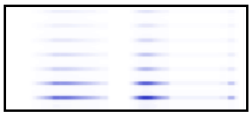
Our model is laid out in Figure 4.1. It has parallel random variables for each note on the piano keyboard. For now, we illustrate these variables for a single concrete note—say C♯ in the 4th octave—and in Section 4.2 describe how the parallel components combine to produce the observed audio signal. Consider a single song, divided into T time steps. The transcription will be I musical events long. The component model for C♯ consists of three primary random variables:



M, a sequence of I symbolic musical events, analogous to the locations and values of symbols along the C♯ staff line in sheet music,



A, a time series of T activations, analogous to the loudness of sound emitted by the C♯ piano string over time as it peaks and attenuates during each event in M,



S, a spectrogram of T frames, specifying the spectrum of frequencies over time in the acoustic signal produced by the C♯ string.

The parameters that generate each of these random variables are depicted in Figure 4.1. First, musical events, M, are generated from a distribution parameterized by $\mu^{(C\sharp)}$, which

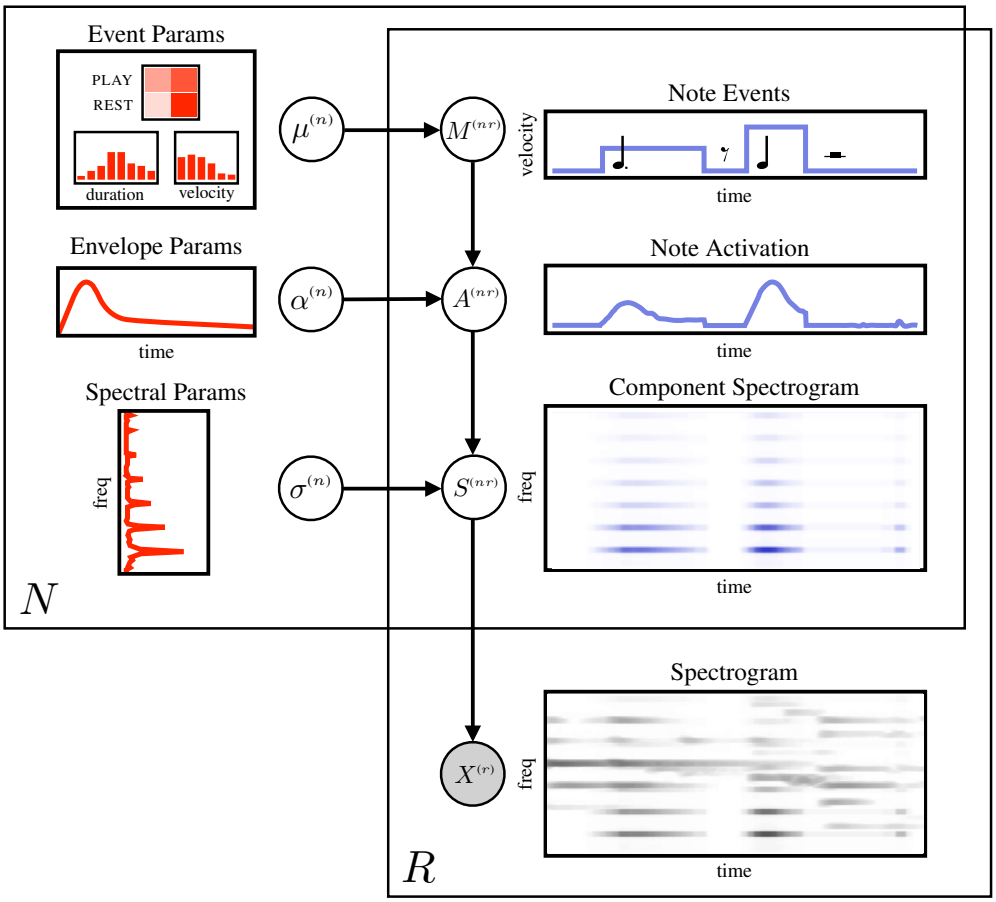


Figure 4.1: We transcribe a dataset consisting of R songs produced by a single piano with N notes. For each keyboard note, n, and each song, r, we generate a sequence of musical events, $M^{(nr)}$, parameterized by $\mu^{(n)}$. Then, conditioned on $M^{(nr)}$, we generate an activation time series, $A^{(nr)}$, parameterized by $\alpha^{(n)}$. Next, conditioned on $A^{(nr)}$, we generate a component spectrogram for note n in song r, $S^{(nr)}$, parameterized by $\sigma^{(n)}$. The observed total spectrogram for song r is produced by superimposing component spectrograms: $X^{(r)} = \sum_n S^{(nr)}$.

specifies the probability that the C♯ key is played, how long it is likely to be held for (duration), and how hard it is likely to be pressed (velocity). Next, the activation of the C♯ string over time, A, is generated conditioned on M from a distribution parameterized by a vector, $\alpha^{(\mathrm{C}\sharp)}$ (see Figure 4.1), which specifies the shape of the rise and fall of the string's activation each time the note is played. Finally, the spectrogram, S, is generated conditioned on A from a distribution parameterized by a vector, $\sigma^{(\mathrm{C}\sharp)}$ (see Figure 4.1), which specifies the frequency distribution of sounds produced by the C♯ string. As depicted in Figure 4.3, S is produced from the outer product of $\sigma^{(\mathrm{C}\sharp)}$ and A. The joint distribution for the note[1] is:

[1]For notational convenience, we suppress the C♯ superscripts on M, A, and S until Section 4.2.

$$
\begin{aligned}
P(S, A, M|\sigma^{(\mathrm{C}\sharp)}, \alpha^{(\mathrm{C}\sharp)}, \mu^{(\mathrm{C}\sharp)}) &= P(M|\mu^{(\mathrm{C}\sharp)}) && \text{[Event Model, Section 4.2]} \\
&\cdot\ P(A|M, \alpha^{(\mathrm{C}\sharp)}) && \text{[Activation Model, Section 4.2]} \\
&\cdot\ P(S|A, \sigma^{(\mathrm{C}\sharp)}) && \text{[Spectrogram Model, Section 4.2]}
\end{aligned}
$$

In the next three sections we give detailed descriptions for each of the component distributions.

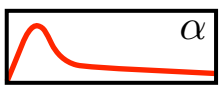
Event Model

Our symbolic representation of musical structure (see Figure 4.2) is similar to the MIDI format used by musical synthesizers. M consists of a sequence of I random variables representing musical events for the C$\sharp$ piano key: $M = (M_1, M_2, \ldots, M_I)$. Each event M_i, is a tuple consisting of a state, E_i, which is either PLAY or REST, a duration D_i, which is a length in time steps, and a velocity V_i, which specifies how hard the key was pressed (assuming E_i is PLAY).

The graphical model for the process that generates M is depicted in Figure 4.2. The sequence of states, $(E_1, E_2, \ldots, E_I)$, is generated from a Markov model. The transition probabilities, μ^{TRANS}, control how frequently the note is played (some notes are more frequent than others). An event's duration, D_i, is generated conditioned on E_i from a distribution parameterized by μ^{DUR}. The durations of PLAY events have a multinomial parameterization, while the durations of REST events are distributed geometrically. An event's velocity, V_i, is a real number on the unit interval and is generated conditioned on E_i from a distribution parameterized by μ^{VEL}. If $E_i =$ REST, deterministically $V_i = 0$. The complete event parameters for keyboard note C$\sharp$ are $\mu^{(\mathrm{C}\sharp)} = (\mu^{\text{TRANS}}, \mu^{\text{DUR}}, \mu^{\text{VEL}})$.

Activation Model

In an actual piano, when a key is pressed, a hammer strikes a string and a sound with sharply rising volume is emitted. The string continues to emit sound as long as the key remains depressed, but the volume decays since no new energy is being transferred. When the key is released, a damper falls back onto the string, truncating the decay. Examples of this trajectory are depicted in Figure 4.1 in the graph of activation values. The graph depicts the note being played softly and held, and then being played more loudly, but held for a shorter time. In our model, PLAY events represent hammer strikes on a piano string with raised damper, while REST events represent the lowered damper. In our parameterization, the shape of the rise and decay is shared by all PLAY events for a given note, regardless of their duration and velocity. We call this shape an *envelope* and describe it using a positive vector of parameters. For our running example of C$\sharp$, this parameter vector is $\alpha^{(\mathrm{C}\sharp)}$ (depicted to the right).



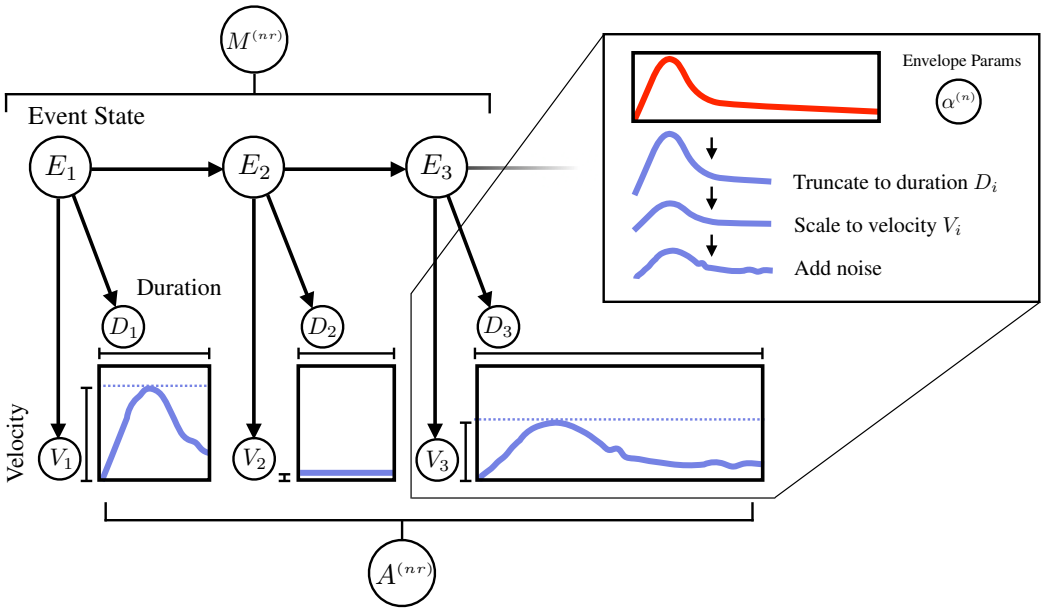


Figure 4.2: Joint distribution on musical events, $M^{(nr)}$, and activations, $A^{(nr)}$, for note n in song r, conditioned on event parameters, $\mu^{(n)}$, and envelope parameters, $\alpha^{(n)}$. The dependence of E_i, D_i, and V_i on n and r is suppressed for simplicity.

The time series of activations for the C$^\sharp$ string, A, is a positive vector of length T, where T denotes the total length of the song in time steps. Let $[A]_t$ be the activation at time step t. As mentioned in Section 4.2, A may be thought of as roughly representing the loudness of sound emitted by the piano string as a function of time. The process that generates A is depicted in Figure 4.2. We generate A conditioned on the musical events, M. Each musical event, $M_i = (E_i, D_i, V_i)$, produces a segment of activations, A_i, of length D_i. For PLAY events, A_i will exhibit an increase in activation. For REST events, the activation will remain low. The segments are appended together to make A. The activation values in each segment are generated in a way loosely inspired by piano mechanics. Given $\alpha^{(\mathrm{C}^\sharp)}$, we generate the values in segment A_i as follows: $\alpha^{(\mathrm{C}^\sharp)}$ is first truncated to length D_i, then is scaled by the velocity of the strike, V_i, and, finally, is used to parameterize an activation noise distribution which generates the segment A_i. Specifically, we add independent Gaussian noise to each dimension after $\alpha^{(\mathrm{C}^\sharp)}$ is truncated and scaled. In principle, this choice gives a formally deficient model, since activations are positive, but in practice performs well and has the benefit of making inference mathematically simple (see Section 4.3).

Component Spectrogram Model

Piano sounds have a harmonic structure; when played, each piano string primarily emits energy at a fundamental frequency determined by the string's length, but also at all integer mul-

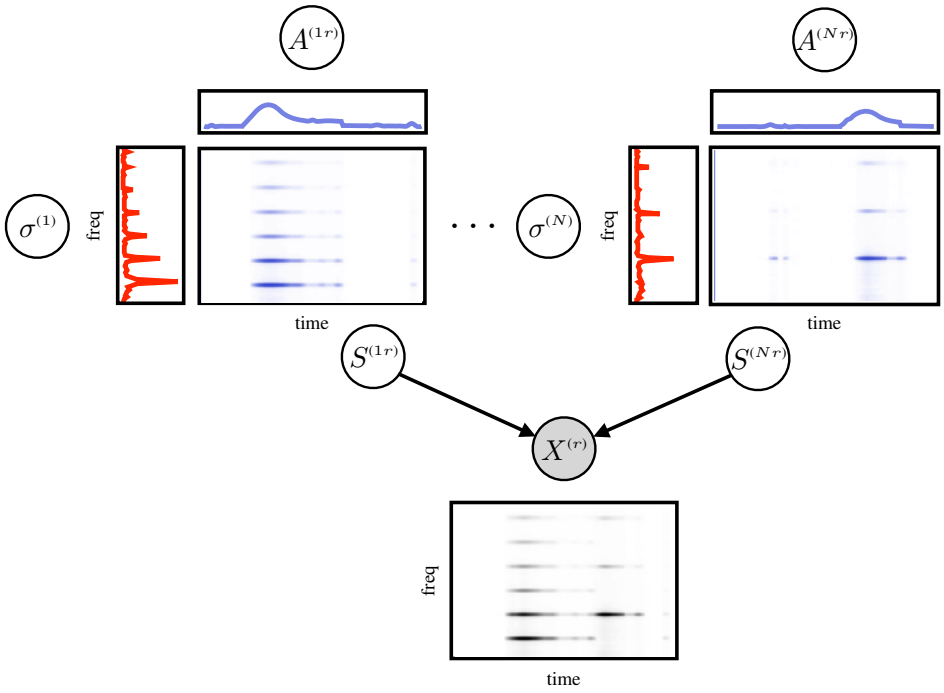
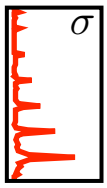


Figure 4.3: Conditional distribution for song r on the observed total spectrogram, $X^{(r)}$, and the component spectrograms for each note, $(S^{(1r)}, \ldots, S^{(Nr)})$, given the activations for each note, $(A^{(1r)}, \ldots, A^{(Nr)})$, and spectral parameters for each note, $(\sigma^{(1)}, \ldots, \sigma^{(N)})$. $X^{(r)}$ is the superposition of the component spectrograms: $X^{(r)} = \sum_n S^{(nr)}$.

tiples of that frequency (called partials) with diminishing strength (see the depiction to the right). For example, the audio signal produced by the C♯ string will vary in loudness, but its frequency distribution will remain mostly fixed. We call this frequency distribution a *spectral profile*. In our parameterization, the spectral profile of C♯ is specified by a positive spectral parameter vector, $\sigma^{(\mathrm{C}\sharp)}$ (depicted to the right). $\sigma^{(\mathrm{C}\sharp)}$ is a vector of length F, where $[\sigma^{(\mathrm{C}\sharp)}]_f$ represents the weight of frequency bin f.



In our model, the audio signal produced by the C♯ string over the course of the song is represented as a spectrogram, S, which is a positive matrix with F rows, one for each frequency bin, f, and T columns, one for each time step, t (see Figures 4.1 and 4.3 for examples). We denote the magnitude of frequency f at time step t as $[S]_{ft}$. In order to generate the spectrogram (see Figure 4.3), we first produce a matrix of intermediate values by taking the outer product of the spectral profile, $\sigma^{(\mathrm{C}\sharp)}$, and the activations, A. These intermediate values are used to parameterize a spectrogram noise distribution that generates S. Specifically, for each frequency bin f and each time step t, the corresponding value of the spectrogram, $[S]_{ft}$, is generated from a noise distribution parameterized by $[\sigma^{(\mathrm{C}\sharp)}]_f \cdot [A]_t$. In practice, the choice of noise distribution is very important. After examining residuals

resulting from fitting mean parameters to actual musical spectrograms, we experimented with various noising assumptions, including multiplicative gamma noise, additive Gaussian noise, log-normal noise, and Poisson noise. Poisson noise performed best. This is consistent with previous findings in the literature, where non-negative matrix factorization using KL divergence (which has a generative interpretation as maximum likelihood inference in a Poisson model (Peeling et al. 2010)) is commonly chosen (Weninger et al. 2013; Benetos and Weyde 2013). Under the Poisson noise assumption, the spectrogram is interpreted as a matrix of (large) integer counts.

Full Model

So far we have only looked at the component of the model corresponding to a single note's contribution to a single song. Our full model describes the generation of a collection of many songs, from a complete instrument with many notes. This full model is diagrammed in Figures 4.1 and 4.3. Let a piano keyboard consist of N notes (on a standard piano, N is 88), where n indexes the particular note. Each note, n, has its own set of musical event parameters, $\mu^{(n)}$, envelope parameters, $\alpha^{(n)}$, and spectral parameters, $\sigma^{(n)}$. Our corpus consists of R songs ("recordings"), where r indexes a particular song. Each pair of note n and song r has it's own musical events variable, $M^{(nr)}$, activations variable, $A^{(nr)}$, and component spectrogram $S^{(nr)}$. The full spectrogram for song r, which is the observed input, is denoted as $X^{(r)}$. Our model generates $X^{(r)}$ by superimposing the component spectrograms: $X^{(r)} = \sum_n S^{(nr)}$. Going forward, we will need notation to group together variables across all N notes: let $\mu = (\mu^{(1)}, \ldots, \mu^{(N)})$, $\alpha = (\alpha^{(1)}, \ldots, \alpha^{(N)})$, and $\sigma = (\sigma^{(1)}, \ldots, \sigma^{(N)})$. Also let $M^{(r)} = (M^{(1r)}, \ldots, M^{(Nr)})$, $A^{(r)} = (A^{(1r)}, \ldots, A^{(Nr)})$, and $S^{(r)} = (S^{(1r)}, \ldots, S^{(Nr)})$.

4.3 Learning and Inference

Our goal is to estimate the unobserved musical events for each song, $M^{(r)}$, as well as the unknown envelope and spectral parameters of the piano that generated the data, α and σ. Our inference will estimate both, though our output is only the musical events, which specify the final transcription. Because MIDI sample banks (piano synthesizers) are readily available, we are able to provide the system with samples from generic pianos (but not from the piano being transcribed). We also give the model information about the distribution of notes in real musical scores by providing it with an external corpus of symbolic music data. Specifically, the following information is available to the model during inference: 1) the total spectrogram for each song, $X^{(r)}$, which is the input, 2) the event parameters, μ, which we estimate by collecting statistics on note occurrences in the external corpus of symbolic music, and 3) truncated normal priors on the envelopes and spectral profiles, α and σ, which we extract from the MIDI samples.

Let $\bar{M} = (M^{(1)}, \ldots, M^{(R)})$, $\bar{A} = (A^{(1)}, \ldots, A^{(R)})$, and $\bar{S} = (S^{(1)}, \ldots, S^{(R)})$, the tuples of event, activation, and spectrogram variables across all notes n and songs r. We would

like to compute the posterior distribution on $\bar{M}$, α, and σ. However, marginalizing over the activations $\bar{A}$ couples the discrete musical structure with the superposition process of the component spectrograms in an intractable way. We instead approximate the joint MAP estimates of $\bar{M}$, $\bar{A}$, α, and σ via iterated conditional modes (Besag 1986), only marginalizing over the component spectrograms, $\bar{S}$. Specifically, we perform the following optimization via block-coordinate ascent:

$$\max_{\bar{M},\bar{A},\alpha,\sigma} \prod_r \left[\sum_{S^{(r)}} P(X^{(r)}, S^{(r)}, A^{(r)}, M^{(r)} | \mu, \alpha, \sigma) \right] \cdot P(\alpha, \sigma)$$

The updates for each group of variables are described in the following sections: $\bar{M}$ in Section 4.3, α in Section 4.3, $\bar{A}$ in Section 4.3, and σ in Section 4.3.

Updating Events

We update $\bar{M}$ to maximize the objective while the other variables are held fixed. The joint distribution on $\bar{M}$ and $\bar{A}$ is a hidden semi-Markov model (Levinson 1986). Given the optimal velocity for each segment of activations, the computation of the emission potentials for the semi-Markov dynamic program is straightforward and the update over $\bar{M}$ can be performed exactly and efficiently. We let the distribution of velocities for PLAY events be uniform. This choice, together with the choice of Gaussian activation noise, yields a simple closed-form solution for the optimal setting of the velocity variable $V_i^{(nr)}$. Let $[\alpha^{(n)}]_j$ denote the jth value of the envelope vector $\alpha^{(n)}$. Let $[A_i^{(nr)}]_j$ be the jth entry of the segment of $A^{(nr)}$ generated by event $M_i^{(nr)}$. The velocity that maximizes the activation segment's probability is given by:

$$V_i^{(nr)} = \frac{\sum_{j=1}^{D_i^{(nr)}} \left([\alpha^{(n)}]_j \cdot [A_i^{(nr)}]_j \right)}{\sum_{j=1}^{D_i^{(nr)}} [\alpha^{(n)}]_j^2}$$

Updating Envelope Parameters

Given settings of the other variables, we update the envelope parameters, α, to optimize the objective. The truncated normal prior on α admits a closed-form update. Let $I(j, n, r) = \{i : D_i^{(nr)} \leq j\}$, the set of event indices for note n in song r with durations no longer than j time steps. Let $[\alpha_0^{(n)}]_j$ be the location parameter for the prior on $[\alpha^{(n)}]_j$, and let β be the scale parameter (which is shared across all n and j). The update for $[\alpha^{(n)}]_j$ is given by:

$$[\alpha^{(n)}]_j = \frac{\sum_{(n,r)} \sum_{i \in I(j,n,r)} V_i^{(nr)} [A_i^{(nr)}]_j + \frac{1}{2\beta^2} [\alpha_0^{(n)}]_j}{\sum_{(n,r)} \sum_{i \in I(j,n,r)} [A_i^{(nr)}]_j^2 + \frac{1}{2\beta^2}}$$

Updating Activations

In order to update the activations, $\bar{A}$, we optimize the objective with respect to $\bar{A}$, with the other variables held fixed. The choice of Poisson noise for generating each of the component spectrograms, $S^{(nr)}$, means that the conditional distribution of the total spectrogram for each song, $X^{(r)} = \sum_n S^{(nr)}$, with $S^{(r)}$ marginalized out, is also Poisson. Specifically, the distribution of $[X^{(r)}]_{ft}$ is Poisson with mean $\sum_n \left([\sigma^{(n)}]_f \cdot [A^{(nr)}]_t\right)$. Optimizing the probability of $X^{(r)}$ under this conditional distribution with respect to $A^{(r)}$ corresponds to computing the supervised NMF using KL divergence (Weninger et al. 2013). However, to perform the correct update in our model, we must also incorporate the distribution of $A^{(r)}$, and so, instead of using the standard multiplicative NMF updates, we use exponentiated gradient ascent (Kivinen and Warmuth 1997) on the log objective. Let L denote the log objective, let $\tilde{\alpha}(n, r, t)$ denote the velocity-scaled envelope value used to generate the activation value $[A^{(nr)}]_t$, and let γ^2 denote the variance parameter for the Gaussian activation noise. The gradient of the log objective with respect to $[A^{(nr)}]_t$ is:

$$\frac{\partial L}{\partial [A^{(nr)}]_t} = \sum_f \left[\frac{[X^{(r)}]_{ft} \cdot [\sigma^{(n)}]_f}{\sum_{n'} \left([\sigma^{(n')}]_f \cdot [A^{(n'r)}]_t\right)} - [\sigma^{(n)}]_f \right] - \frac{1}{\gamma^2}\left([A^{(nr)}]_t - \tilde{\alpha}(n, r, t)\right)$$

Updating Spectral Parameters

The update for the spectral parameters, σ, is similar to that of the activations. Like the activations, σ is part of the parameterization of the Poisson distribution on each $X^{(r)}$. We again use exponentiated gradient ascent. Let $[\sigma_0^{(n)}]_f$ be the location parameter of the prior on $[\sigma^{(n)}]_f$, and let ξ be the scale parameter (which is shared across all n and f). The gradient of the the log objective with respect to $[\sigma^{(n)}]_f$ is given by:

$$\frac{\partial L}{\partial [\sigma^{(n)}]_f} = \sum_{(r,t)} \left[\frac{[X^{(r)}]_{ft} \cdot [A^{(nr)}]_t}{\sum_{n'} \left([\sigma^{(n')}]_f \cdot [A^{(n'r)}]_t\right)} - [A^{(nr)}]_t \right] - \frac{1}{\xi^2}\left([\sigma^{(n)}]_f - [\sigma_0^{(n)}]_f\right)$$

4.4 Experiments

Because polyphonic transcription is so challenging, much of the existing literature has either worked with synthetic (e.g. MIDI) data (Ryynanen and Klapuri 2005) or assumed access to the test instrument during training (Poliner and Ellis 2006; Boogaart and Lienhart 2009; Böck and Schedl 2012; Weninger et al. 2013). As our ultimate goal is the transcription of arbitrary recordings from real, previously-unseen pianos, we evaluate in an unsupervised setting, on recordings from an acoustic piano not observed in training.

Data We evaluate on the MIDI-Aligned Piano Sounds (MAPS) corpus (Emiya et al. 2010). This corpus includes a collection of piano recordings from a variety of time periods and styles,

performed by a human player on an acoustic "Disklavier" piano equipped with electromechanical sensors under the keys. These make it possible to transcribe directly into MIDI while the instrument is in use, providing a ground-truth transcript to accompany the audio for the purpose of evaluation. In keeping with much of the existing music transcription literature, we use the first 30 seconds of each of the 30 `ENSTDkAm` recordings as a development set, and the first 30 seconds of each of the 30 `ENSTDkCl` recordings as a test set. We also assume access to a collection of synthesized piano sounds for parameter initialization, which we take from the MIDI portion of the MAPS corpus, and a large collection of symbolic music data from the IMSLP library (*The International Music Score Library Project* 2014; Viro 2011), used to estimate the event parameters in our model.

Preprocessing Similar to the approach used by Weninger et al. (2013), we represent the input audio as a magnitude spectrum short-time Fourier transform with a 4096-frame window and a hop size of 512 frames. We temporally downsample the resulting spectrogram by a factor of 2, taking the maximum magnitude over collapsed bins. The input audio is recorded at 44.1 kHz and the resulting spectrogram has 23ms frames.

Initialization and Learning We estimate initializers and priors for the spectral parameters, σ, and envelope parameters, α, by fitting isolated, synthesized, piano sounds. We collect these isolated sounds from the MIDI portion of MAPS, and average the parameter values across several synthesized pianos. We estimate the event parameters μ by counting note occurrences in the IMSLP data. At decode time, to fit the spectral and envelope parameters and predict transcriptions, we run 5 iterations of the block-coordinate ascent procedure described in Section 4.3.

Evaluation We report two standard measures of performance: an *onset* evaluation, in which a predicted note is considered correct if it falls within 50ms of a note in the true transcription, and a *frame-level* evaluation, in which each transcription is converted to a boolean matrix specifying which notes are active at each time step, discretized to 10ms frames. Each entry is compared to the corresponding entry in the true matrix. Frame-level evaluation is sensitive to offsets as well as onsets, but does not capture the fact that note onsets have greater musical significance than do offsets. As is standard, we report precision (P), recall (R), and F_1-measure (F_1) for each of these metrics.

Comparisons

We compare our system to three state-of-the-art unsupervised systems: the hidden semi-Markov model described by Benetos and Weyde (2013) and the spectrally-constrained factorization models described by Vincent et al. (2010) and O'Hanlon and Plumbley (2014). To our knowledge, Benetos and Weyde (2013) report the best published onset results for this dataset, and O'Hanlon and Plumbley (2014) report the best frame-level results.

System	Onsets			Frames		
	P	R	F_1	P	R	F_1
Discriminative (Weninger et al. 2013)	76.8	65.1	70.4	-	-	-
Benetos (Benetos and Weyde 2013)	-	-	68.6	-	-	68.0
Vincent (Vincent et al. 2010)	62.7	76.8	69.0	79.6	63.6	70.7
O'Hanlon (O'Hanlon and Plumbley 2014)	48.6	73.0	58.3	73.4	72.8	73.2
This work	78.1	74.7	**76.4**	69.1	80.7	**74.4**

Table 4.1: Unsupervised transcription results on the MAPS corpus. "Onsets" columns show scores for identification (within ±50ms) of note start times. "Frames" columns show scores for 10ms frame-level evaluation. Our system outperforms state-of-the-art systems on both metrics.[2]

The literature also describes a number of supervised approaches to this task. In these approaches, a model is trained on annotated recordings from a known instrument. While best performance is achieved when testing on the same instrument used for training, these models can also achieve reasonable performance when applied to new instruments. Thus, we also compare to a discriminative baseline, a simplified reimplementation of a state-of-the-art supervised approach (Weninger et al. 2013) which achieves slightly better performance than the original on this task. This system only produces note onsets, and therefore is not evaluated at a frame-level. We train the discriminative baseline on synthesized audio with ground-truth MIDI annotations, and apply it directly to our test instrument, which the system has never seen before.

Results

Our model outperforms the best published approaches to this task: as shown in Table 4.1, it achieves an onset F_1 of 76.4, which corresponds to a 10.6% relative gain over the onset F_1 achieved by the system of Vincent et al. (2010), the top-performing unsupervised baseline on this metric. Surprisingly, the discriminative baseline (Weninger et al. 2013), which was not developed for the unsupervised task, outperforms all the unsupervised baselines in terms of onset evaluation, achieving an F_1 of 70.4. Evaluated on frames, our system achieves an F_1 of 74.4, corresponding to a more modest 1.6% relative gain over the system of O'Hanlon and Plumbley (2014), which is the best performing baseline on this metric.

The remarkable performance of the discriminative baseline shows that it is possible to achieve high onset accuracy on this task without adapting to the test instrument. Thus, it is reasonable to ask how much of the gain our model achieves is due to its ability to learn instrument timbre. If we run our block-coordinate ascent procedure (Section 4.3) without updating the envelope and spectral parameters, we achieve an onset F_1 of 72.6 and a frame-level F_1 of 71.0, indicating that learning instrument timbre does indeed help performance.

Figure 4.4 shows an example of our system's output passed through a commercially-available MIDI-to-sheet-music converter. This example was chosen because its onset F_1 of

75.5 and error types are broadly representative of the system's output on our data. The resulting score has musically plausible errors.

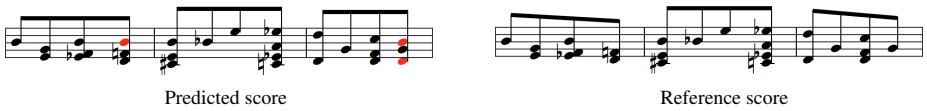


Figure 4.4: Result of passing our system's prediction and the reference transcription MIDI through the GarageBand MIDI-to-sheet-music converter. This is a transcription of the first three bars of Schumann's *Hobgoblin*, on which our system achieves an onset F_1 of 75.5.

Inspection of the system's output suggests that a large fraction of errors are either off by an octave (i.e. the frequency of the predicted note is half or double the correct frequency), or are segmentation errors (in which a single key press is transcribed as several consecutive key presses). These errors are not detrimental to musical perception, and converting the transcriptions back into an audio signal using a synthesizer yields music that is qualitatively quite similar to the original recordings.

[2] For consistency we re-ran all systems in this table with our own evaluation code (except for the system of Benetos and Weyde (2013), for which numbers are taken from the paper). For O'Hanlon and Plumbley (2014) scores are higher than the authors themselves report; this is due to a post-processing step suggested in personal correspondence.

Chapter 5

Unsupervised Learning with Features

5.1 Overview

In previous chapters we incorporated richer domain knowledge into unsupervised models by encoding that knowledge in the form of conditional independence structure. We achieved substantial empirical gains on two tasks. However, because the connection between independence and knowledge is subtle, injecting knowledge was sometimes tricky—we had to carefully consider causal relationships. Further, new conditional independence assumptions necessitated the development new inference algorithms.

In this chapter, we develop a different approach to adding inductive bias to unsupervised learning problems: instead of focusing on the model, we add structure to the *prior* by using features. We present a range of experiments wherein we improve existing unsupervised models by declaratively adding rich features sets. In particular, we parameterize the local multinomials of existing generative models using features, in a way which does not require complex new machinery but which still provides substantial flexibility. In the feature-engineering paradigm, one can worry less about the backbone structure and instead use hand-designed features to declaratively inject domain knowledge into a model. While feature engineering has historically been associated with discriminative, supervised learning settings, we argue that it can and should be applied more broadly to the unsupervised setting.

The idea of using features in unsupervised learning is neither new nor even controversial. Many top unsupervised results use feature-based models (N. A. Smith and Eisner 2005; Haghighi and Klein 2006). However, such approaches have presented their own barriers, from challenging normalization problems, to neighborhood design, to the need for complex optimization procedures. As a result, most work still focuses on the stable and intuitive approach of using the EM algorithm to optimize data likelihood in locally normalized, generative models.

The primary contribution of this chapter is to demonstrate the clear empirical success of a simple and accessible approach to unsupervised learning with features, which can be

optimized by using standard NLP building blocks. We consider the same generative, locally-normalized models that dominate past work on a range of tasks. However, we follow Chen (2003), Bisani and Ney (2008), and Bouchard-Côté et al. (2008), and allow each component multinomial of the model to be a miniature multi-class logistic regression model. In this case, the EM algorithm still applies with the E-step unchanged. The M-step involves gradient-based training familiar from standard supervised logistic regression (i.e., maximum entropy models). By integrating these two familiar learning techniques, we add features to unsupervised models without any specialized learning or inference.

A second contribution of this chapter is to show that further gains can be achieved by directly optimizing data likelihood with LBFGS (Liu and Nocedal 1989). This alternative optimization procedure requires no additional machinery beyond what EM uses. This approach is still very simple to implement, and we found that it empirically outperforms EM.

This chapter is largely empirical; the underlying optimization techniques are known, even if the overall approach will be novel to many readers. As an empirical demonstration, our results span an array of unsupervised learning tasks: part-of-speech induction, grammar induction, word alignment, and word segmentation. In each task, we show that declaring a few linguistically motivated feature templates yields state-of-the-art results.

5.2 Models

We start by explaining our feature-enhanced model for part-of-speech (POS) induction. This particular example illustrates our approach to adding features to unsupervised models in a well-known NLP task. We then explain how the technique applies more generally.

Example: Part-of-Speech Induction

POS induction consists of labeling words in text with POS tags. A hidden Markov model (HMM) is a standard model for this task, used in both a frequentist setting (Merialdo 1994; Elworthy 1994) and in a Bayesian setting (Goldwater and Griffiths 2007; Johnson 2007).

A POS HMM generates a sequence of words in order. In each generation step, an observed word emission y_i and a hidden successor POS tag z_{i+1} are generated independently, conditioned on the current POS tag z_i . This process continues until an absorbing stop state is generated by the transition model.

There are two types of conditional distributions in the model—emission and transition probabilities—that are both multinomial probability distributions. The joint likelihood factors into these distributions:

$$P_{\boldsymbol{\theta}}(\mathbf{Y} = \mathbf{y}, \mathbf{Z} = \mathbf{z}) = P_{\boldsymbol{\theta}}(Z_1 = z_1) \cdot \prod_{i=1}^{|\mathbf{z}|} P_{\boldsymbol{\theta}}(Y_i = y_i | Z_i = z_i) \cdot P_{\boldsymbol{\theta}}(Z_{i+1} = z_{i+1} | Z_i = z_i)$$

The emission distribution $P_{\boldsymbol{\theta}}(Y_i = y_i | Z_i = z_i)$ is parameterized by conditional probabilities $\theta_{y,z,\text{EMIT}}$ for each word y given tag z. Alternatively, we can express this emission distribution as the output of a logistic regression model, replacing the explicit conditional probability table by a logistic function parameterized by weights and features:

$$\theta_{y,z,\text{EMIT}}(\mathbf{w}) = \frac{\exp \langle \mathbf{w}, \mathbf{f}(y, z, \text{EMIT}) \rangle}{\sum_{y'} \exp \langle \mathbf{w}, \mathbf{f}(y', z, \text{EMIT}) \rangle}$$

This feature-based logistic expression is equivalent to the flat multinomial in the case that the feature function $\mathbf{f}(y, z, \text{EMIT})$ consists of all indicator features on tuples (y, z, EMIT), which we call BASIC features. The equivalence follows by setting weight $w_{y,z,\text{EMIT}} = \log(\theta_{y,z,\text{EMIT}})$.[1] This formulation is known as the natural parameterization of the multinomial distribution.

In order to enhance this emission distribution, we include coarse features in $\mathbf{f}(y, z, \text{EMIT})$, in addition to the BASIC features. Crucially, these features can be active across multiple (y, z) values. In this way, the model can abstract general patterns, such as a POS tag co-occurring with an inflectional morpheme. We discuss specific POS features in Section 5.4.

General Directed Models

Like the HMM, all of the models we propose are based on locally normalized generative decisions that condition on some context. In general, let $\mathbf{X} = (\mathbf{Z}, \mathbf{Y})$ denote the sequence of generation steps (random variables) where $\mathbf{Z}$ contains all hidden random variables and $\mathbf{Y}$ contains all observed random variables. The joint probability of this directed model factors as:

$$P_{\mathbf{w}}(\mathbf{X} = \mathbf{x}) = \prod_{i \in I} P_{\mathbf{w}} \left(X_i = x_i \middle| X_{\pi(i)} = x_{\pi(i)} \right),$$

where $X_{\pi(i)}$ denotes the parents of X_i and I is the index set of the variables in $\mathbf{X}$.

In the models that we use, each factor in the above expression is the output of a local logistic regression model parameterized by $\mathbf{w}$:

$$P_{\mathbf{w}}\big(X_i = d \big| X_{\pi(i)} = c\big) = \frac{\exp\langle \mathbf{w}, \mathbf{f}(d, c, t) \rangle}{\sum_{d'} \exp\langle \mathbf{w}, \mathbf{f}(d', c, t) \rangle}$$

Above, d is the generative decision value for X_i picked by the model, c is the conditioning context tuple of values for the parents of X_i, and t is the type of decision being made. For instance, the POS HMM has two types of decisions: transitions and emissions. In the emission model, the type t is EMIT, the decision d is a word and the context c is a tag. The denominator normalizes the factor to be a probability distribution over decisions.

[1] As long as no transition or emission probabilities are equal to zero. When zeros are present, for instance to model that an absorbing stop state can only transition to itself, it is often possible to absorb these zeros into a *base measure*. All the arguments in this chapter carry with a structured base measure; we drop it for simplicity.

The objective function we derive from this model is the marginal likelihood of the observations $\mathbf{y}$, along with a regularization term:

$$L(\mathbf{w}) = \log P_{\mathbf{w}}(\mathbf{Y} = \mathbf{y}) - \kappa ||\mathbf{w}||_2^2 \tag{5.1}$$

This model has two advantages over the more prevalent form of a feature-rich unsupervised model, the globally normalized Markov random field.[2] First, as we explain in Section 5.3, optimizing our objective does not require computing expectations over the joint distribution. In the case of the POS HMM, for example, we do not need to enumerate an infinite sum of products of potentials when optimizing, in contrast to Haghighi and Klein (2006). Second, we found that locally normalized models empirically outperform their globally normalized counterparts, despite their efficiency and simplicity.

5.3 Optimization

Optimizing with Expectation Maximization

In this section, we describe the EM algorithm applied to our feature-rich, locally normalized models. For models parameterized by standard multinomials, EM optimizes $L(\boldsymbol{\theta}) = \log P_{\boldsymbol{\theta}}(\mathbf{Y} = \mathbf{y})$ (Dempster et al. 1977). The E-step computes expected counts for each tuple of decision d, context c, and multinomial type t:

$$e_{d,c,t} \leftarrow \mathbb{E}_{\boldsymbol{\theta}} \left[\sum_{i \in I} \mathbb{1}(X_i = d, X_{\pi(i)} = c, t) \middle| \mathbf{Y} = \mathbf{y} \right] \tag{5.2}$$

These expected counts are then normalized in the M-step to re-estimate $\boldsymbol{\theta}$:

$$\theta_{d,c,t} \leftarrow \frac{e_{d,c,t}}{\sum_{d'} e_{d',c,t}}$$

Normalizing expected counts in this way maximizes the expected complete log likelihood with respect to the current model parameters.

EM can likewise optimize $L(\mathbf{w})$ for our locally normalized models with logistic parameterizations. The E-step first precomputes multinomial parameters from $\mathbf{w}$ for each decision, context, and type:

$$\theta_{d,c,t}(\mathbf{w}) \leftarrow \frac{\exp\langle \mathbf{w}, \mathbf{f}(d,c,t) \rangle}{\sum_{d'} \exp\langle \mathbf{w}, \mathbf{f}(d',c,t) \rangle}$$

[2]The locally normalized model class is actually equivalent to its globally normalized counterpart when the former meets the following three conditions: (1) The graphical model is a directed tree. (2) The BASIC features are included in $\mathbf{f}$. (3) We do not include regularization in the model ($\kappa = 0$). This follows from N. A. Smith and Johnson (2007).

Then, expected counts $\mathbf{e}$ are computed according to Equation 5.2. In the case of POS induction, expected counts are computed with the forward-backward algorithm in both the standard and logistic parameterizations. The only change is that the conditional probabilities $\boldsymbol{\theta}$ are now functions of $\mathbf{w}$.

The M-step changes more substantially, but still relies on canonical NLP learning methods. We wish to choose $\mathbf{w}$ to optimize the regularized expected complete log likelihood:

$$\ell(\mathbf{w}, \mathbf{e}) = \sum_{d,c,t} e_{d,c,t} \log \theta_{d,c,t}(\mathbf{w}) - \kappa ||\mathbf{w}||_2^2 \tag{5.3}$$

We optimize this objective via a gradient-based search algorithm like LBFGS. The gradient with respect to $\mathbf{w}$ takes the form

$$\nabla \ell(\mathbf{w}, \mathbf{e}) = \sum_{d,c,t} e_{d,c,t} \cdot \Delta_{d,c,t}(\mathbf{w}) - 2\kappa \cdot \mathbf{w} \tag{5.4}$$

$$\Delta_{d,c,t}(\mathbf{w}) = \mathbf{f}(d, c, t) - \sum_{d'} \theta_{d',c,t}(\mathbf{w}) \mathbf{f}(d', c, t)$$

This gradient matches that of regularized logistic regression in a supervised model: the difference Δ between the observed and expected features, summed over every decision and context. In the supervised case, we would observe the count of occurrences of (d, c, t), but in the unsupervised M-step, we instead substitute expected counts $e_{d,c,t}$.

This gradient-based M-step is an iterative procedure. For each different value of $\mathbf{w}$ considered during the search, we must recompute $\boldsymbol{\theta}(\mathbf{w})$, which requires computation in proportion to the size of the parameter space. However, $\mathbf{e}$ stays fixed throughout the M-step. Algorithm 1 outlines EM in its entirety. The subroutine climb$(\cdot, \cdot, \cdot)$ represents a generic optimization step such as an LBFGS iteration.

Algorithm 1 Feature-enhanced EM

```
repeat
  Compute expected counts e                          ▷ Eq. 5.2
  repeat
    Compute ℓ(w, e)                                  ▷ Eq. 5.3
    Compute ∇ℓ(w, e)                                 ▷ Eq. 5.4
    w ← climb(w, ℓ(w, e), ∇ℓ(w, e))
  until convergence
until convergence
```

Direct Marginal Likelihood Optimization

Another approach to optimizing Equation 5.1 is to compute the gradient of the log marginal likelihood directly (Salakhutdinov et al. 2003). The gradient turns out to have the same

form as Equation 5.4, with the key difference that $e_{d,c,t}$ is recomputed for every different value of $\mathbf{w}$. Algorithm 2 outlines the procedure. Justification for this algorithm appears in the Appendix.

Algorithm 2 Feature-enhanced direct gradient

repeat
  Compute expected counts $\mathbf{e}$ ▷ Eq. 5.2
  Compute $L(\mathbf{w})$ ▷ Eq. 5.1
  Compute $\nabla\ell(\mathbf{w}, \mathbf{e})$ ▷ Eq. 5.4
  $\mathbf{w} \leftarrow \text{climb}(\mathbf{w}, L(\mathbf{w}), \nabla\ell(\mathbf{w}, \mathbf{e}))$
until convergence

In practice, we find that this optimization approach leads to higher task accuracy for several models. However, in cases where computing $e_{d,c,t}$ is expensive, EM can be a more efficient alternative.

5.4 Part-of-Speech Induction

We now describe experiments that demonstrate the effectiveness of locally normalized logistic models. We first use the bigram HMM described in Section 5.2 for POS induction, which has two types of multinomials. For type EMIT, the decisions d are words and contexts c are tags. For type TRANS, the decisions and contexts are both tags.

POS Induction Features

We use the same set of features used by Haghighi and Klein (2006) in their baseline globally normalized Markov random field (MRF) model. These are all coarse features on emission contexts that activate for words with certain orthographic properties. We use only the BASIC features for transitions. For an emission with word y and tag z, we use the following feature templates:

BASIC:	$\mathbb{1}(y = \cdot, z = \cdot)$
CONTAINS-DIGIT:	Check if y contains digit and conjoin with z: $\mathbb{1}(\text{containsDigit}(y) = \cdot, z = \cdot)$
CONTAINS-HYPHEN:	$\mathbb{1}(\text{containsHyphen}(x) = \cdot, z = \cdot)$
INITIAL-CAP:	Check if the first letter of y is capitalized: $\mathbb{1}(\text{isCap}(y) = \cdot, z = \cdot)$
SUFFIX:	Indicator functions for character suffixes of up to length 3 present in y.

POS Induction Data and Evaluation

We train and test on the entire WSJ tag corpus (Marcus et al. 1993). We attempt the most difficult version of this task where the only information our system can make use of is the unlabeled text itself. In particular, we do not make use of a tagging dictionary. We use 45 tag clusters, the number of POS tags that appear in the WSJ corpus. There is an identifiability issue when evaluating inferred tags. In order to measure accuracy on the hand-labeled corpus, we map each cluster to the tag that gives the highest accuracy, the many-1 evaluation approach (Johnson 2007). We run all POS induction models for 1000 iterations, with 10 random initializations. The mean and standard deviation of many-1 accuracy appears in Table 5.1.

POS Induction Results

We compare our model to the basic HMM and a bigram version of the feature-enhanced MRF model of Haghighi and Klein (2006). Using EM, we achieve a many-1 accuracy of 68.1. This outperforms the basic HMM baseline by a 5.0 margin. The same model, trained using the direct gradient approach, achieves a many-1 accuracy of 75.5, outperforming the basic HMM baseline by a margin of 12.4. These results show that the direct gradient approach can offer additional boosts in performance when used with a feature-enhanced model. We also outperform the globally normalized MRF, which uses the same set of features and which we train using a direct gradient approach.

5.5 Grammar Induction

We next apply our technique to a grammar induction task: the unsupervised learning of dependency parse trees via the dependency model with valence (DMV) (Klein and Manning 2004). A dependency parse is a directed tree over tokens in a sentence. Each edge of the tree specifies a directed dependency from a head token to a dependent, or argument token. Thus, the number of dependencies in a parse is exactly the number of tokens in the sentence, not counting the artificial root token.

Dependency Model with Valence

The DMV defines a probability distribution over dependency parse trees. In this head-outward attachment model, a parse and the word tokens are derived together through a recursive generative process. For each token generated so far, starting with the root, a set of left dependents is generated, followed by a set of right dependents.

There are two types of multinomial distributions in this model. The Bernoulli STOP probabilities $\theta_{d,c,\text{STOP}}$ capture the valence of a particular head. For this type, the decision d is whether or not to stop generating arguments, and the context c contains the current head h, direction δ and adjacency adj. If a head's stop probability is high, it will be encouraged to

accept few arguments. The ATTACH multinomial probability distributions $\theta_{d,c,\text{ATTACH}}$ capture attachment preferences of heads. For this type, a decision d is an argument token a, and the context c consists of a head h and a direction δ.

We take the same approach as previous work (Klein and Manning 2004; Cohen and N. A. Smith 2009) and use gold POS tags in place of words.

Grammar Induction Features

One way to inject knowledge into a dependency model is to encode the similarity between the various morphological variants of nouns and verbs. We encode this similarity by incorporating features into both the STOP and the ATTACH probabilities. The attachment features appear below; the stop feature templates are similar and are therefore omitted.

BASIC:	$\mathbb{1}(a = \cdot, h = \cdot, \delta = \cdot)$
NOUN:	Generalize the morphological variants of nouns by using isNoun$(\cdot)$: $\mathbb{1}(a = \cdot, \text{isNoun}(h) = \cdot, \delta = \cdot)$ $\mathbb{1}(\text{isNoun}(a) = \cdot, h = \cdot, \delta = \cdot)$ $\mathbb{1}(\text{isNoun}(a) = \cdot, \text{isNoun}(h) = \cdot, \delta = \cdot)$
VERB:	Same as above, generalizing verbs instead of nouns by using isVerb$(\cdot)$
NOUN-VERB:	Same as above, generalizing with isVerbOrNoun$(\cdot) =$ isVerb$(\cdot) \vee$ isNoun$(\cdot)$
BACK-OFF:	We add versions of all other features that ignore direction or adjacency.

While the model has the expressive power to allow specific morphological variants to have their own behaviors, the existence of coarse features encourages uniform analyses, which in turn gives better accuracies.

Cohen and N. A. Smith's (2009) method has similar characteristics. They add a shared logistic-normal prior (SLN) to the DMV in order to tie multinomial parameters across related derivation events. They achieve their best results by only tying parameters between different multinomials when the corresponding contexts are headed by nouns and verbs. This observation motivates the features we choose to incorporate into the DMV.

Grammar Induction Data and Evaluation

For our English experiments we train and report directed attachment accuracy on portions of the WSJ corpus. We work with a standard, reduced version of WSJ, WSJ10, that contains only sentences of length 10 or less after punctuation has been removed. We train on sections 2-21, and use section 22 as a development set. We report accuracy on section 23. These are the same training, development, and test sets used by Cohen and N. A. Smith (2009). The regularization parameter (κ) is tuned on the development set to maximize accuracy.

For our Chinese experiments, we use the same corpus and training/test split as Cohen and N. A. Smith (2009). We train on sections 1-270 of the Penn Chinese Treebank (Xue et al. 2002), similarly reduced (CTB10). We test on sections 271-300 of CTB10, and use sections 400-454 as a development set.

The DMV is known to be sensitive to initialization. We use the deterministic harmonic initializer from Klein and Manning (2004). We ran each optimization procedure for 100 iterations. The results are reported in Table 5.1.

Grammar Induction Results

We are able to outperform Cohen and N. A. Smith's (2009) best system, which requires a more complicated variational inference method, on both English and Chinese data sets. Their system achieves an accuracy of 61.3 for English and an accuracy of 51.9 for Chinese.[3] Our feature-enhanced model, trained using the direct gradient approach, achieves an accuracy of 63.0 for English, and an accuracy of 53.6 for Chinese.

5.6 Word Alignment

Word alignment is a core machine learning component of statistical machine translation systems, and one of the few NLP tasks that is dominantly solved using unsupervised techniques. The purpose of word alignment models is to induce a correspondence between the words of a sentence and the words of its translation.

Word Alignment Models

We consider two classic generative alignment models that are both used heavily today, IBM Model 1 (Brown et al. 1994) and the HMM alignment model (Vogel et al. 1996). These models generate a hidden alignment vector $\mathbf{z}$ and an observed foreign sentence $\mathbf{y}$, all conditioned on an observed English sentence $\mathbf{e}$. The likelihood of both models takes the form:

$$P(\mathbf{y}, \mathbf{z}|\mathbf{e}) = \prod_j p(z_j = i|z_{j-1}) \cdot \theta_{y_j, e_i, \text{ALIGN}}$$

The distortion term $p(z_j = i|z_{j-1})$ is uniform in Model 1, and Markovian in the HMM. See Liang, Taskar, et al. (2006) for details on the specific variant of the distortion model of the HMM that we used. We use these standard distortion models in both the baseline and feature-enhanced word alignment systems.

The bilexical emission model $\theta_{y,e,\text{ALIGN}}$ differentiates our feature-enhanced system from the baseline system. In the former, the emission model is a standard conditional multinomial

[3] Using additional bilingual data, Cohen and N. A. Smith (2009) achieve an accuracy of 62.0 for English, and an accuracy of 52.0 for Chinese, still below our results.

that represents the probability that decision word y is generated from context word e, while in our system, the emission model is re-parameterized as a logistic regression model and feature-enhanced.

Many supervised feature-based alignment models have been developed. In fact, this logistic parameterization of the HMM has been proposed before and yielded alignment improvements, but was trained using supervised estimation techniques (Garcia-Varea et al. 2002).[4] However, most full translation systems today rely on unsupervised learning so that the models may be applied easily to many language pairs. Our approach provides efficient and consistent unsupervised estimation for feature-rich alignment models.

Word Alignment Features

The BASIC features on pairs of lexical items provide strong baseline performance. We add coarse features to the model in order to inject prior knowledge and tie together lexical items with similar characteristics.

BASIC:	$\mathbb{1}(e = \cdot, y = \cdot)$
EDIT-DISTANCE:	$\mathbb{1}(\text{dist}(y, e) = \cdot)$
DICTIONARY:	$\mathbb{1}((y, e) \in D)$ for dictionary D.
STEM:	$\mathbb{1}(\text{stem}(e) = \cdot, y = \cdot)$ for Porter stemmer.
PREFIX:	$\mathbb{1}(\text{prefix}(e) = \cdot, y = \cdot)$ for prefixes of length 4.
CHARACTER:	$\mathbb{1}(e = \cdot, \text{charAt}(y, i) = \cdot)$ for index i in the Chinese word.

These features correspond to several common augmentations of word alignment models, such as adding dictionary priors and truncating long words, but here we integrate them all coherently into a single model.

Word Alignment Data and Evaluation

We evaluate on the standard hand-aligned portion of the NIST 2002 Chinese-English development set (Ayan et al. 2005). The set is annotated with sure S and possible P alignments. We measure alignment quality using alignment error rate (AER) (Och and Ney 2000).

We train the models on 10,000 sentences of FBIS Chinese-English newswire. This is not a large-scale experiment, but large enough to be relevant for low-resource languages. LBFGS experiments are not provided because computing expectations in these models is too computationally intensive to run for many iterations. Hence, EM training is a more

[4] Garcia-Varea et al. (2002) describes unsupervised EM optimization with logistic regression models at a high level—their *dynamic training* approach—but provides no experiments.

	Model	**Inference**	**Reg**	**Eval**
POS Induction			κ	Many-1
WSJ	Basic-HMM	EM	–	63.1 (1.3)
	Feature-MRF	LBFGS	0.1	59.6 (6.9)
	Feature-HMM	EM	1.0	68.1 (1.7)
		LBFGS	1.0	**75.5** (1.1)
Grammar Induction			κ	Dir
WSJ10	Basic-DMV	EM	–	47.8
	Feature-DMV	EM	0.05	48.3
		LBFGS	10.0	**63.0**
	(Cohen and N. A. Smith 2009)			61.3
CTB10	Basic-DMV	EM	–	42.5
	Feature-DMV	EM	1.0	49.9
		LBFGS	5.0	**53.6**
	(Cohen and N. A. Smith 2009)			51.9
Word Alignment			κ	AER
NIST ChEn	Basic-Model 1	EM	–	38.0
	Feature-Model 1	EM	–	**35.6**
	Basic-HMM	EM	–	33.8
	Feature-HMM	EM	–	**30.0**
Word Segmentation			κ	F1
BR	Basic-Unigram	EM	–	76.9 (0.1)
	Feature-Unigram	EM	0.2	84.5 (0.5)
		LBFGS	0.2	**88.0** (0.1)
	(Johnson and Goldwater 2009)			87

Table 5.1: Locally normalized feature-based models outperform all proposed baselines for all four tasks. LBFGS outperformed EM in all cases where the algorithm was sufficiently fast to run. Details of each experiment appear in the main text.

appropriate optimization approach: computing the M-step gradient requires only summing over word type pairs, while the marginal likelihood gradient needed for LBFGS requires summing over training sentence alignments. The final alignments, in both the baseline and the feature-enhanced models, are computed by training the generative models in both directions, combining the result with hard union competitive thresholding (DeNero and Klein 2007), and using agreement training for the HMM (Liang, Taskar, et al. 2006). The combination of these techniques yields a state-of-the-art unsupervised baseline for Chinese-English.

Word Alignment Results

For both IBM Model 1 and the HMM alignment model, EM training with feature-enhanced models outperforms the standard multinomial models, by 2.4 and 3.8 AER respectively.[5] As expected, large positive weights are assigned to both the dictionary and edit distance features. Stem and character features also contribute to the performance gain.

5.7 Word Segmentation

Finally, we show that it is possible to improve upon the simple and effective word segmentation model presented in Liang and Klein (2009) by adding phonological features. Unsupervised word segmentation is the task of identifying word boundaries in sentences where spaces have been removed. For a sequence of characters $\mathbf{y} = (y_1, ..., y_n)$, a segmentation is a sequence of segments $\mathbf{z} = (z_1, ..., z_{|\mathbf{z}|})$ such that $\mathbf{z}$ is a partition of $\mathbf{y}$ and each z_i is a contiguous subsequence of $\mathbf{y}$. Unsupervised models for this task infer word boundaries from corpora of sentences of characters without ever seeing examples of well-formed words.

Unigram Double-Exponential Model

Liang and Klein's (2009) unigram double-exponential model corresponds to a simple derivational process where sentences of characters $\mathbf{x}$ are generated a word at a time, drawn from a multinomial over all possible strings $\theta_{z,\text{SEGMENT}}$. For this type, there is no context and the decision is the particular string generated. In order to avoid the degenerate MLE that assigns mass only to single segment sentences it is helpful to independently generate a length for each segment from a fixed distribution. Liang and Klein (2009) constrain individual segments to have maximum length 10 and generate lengths from the following distribution: $\theta_{l,\text{LENGTH}} = \exp(-l^{1.6})$ when $1 \leq l \leq 10$. Their model is deficient since it is possible to generate lengths that are inconsistent with the actual lengths of the generated segments. The likelihood equation is given by:

[5] The best published results for this dataset are supervised, and trained on 17 times more data (Haghighi, Blitzer, et al. 2009).

$$P(\mathbf{Y} = \mathbf{y}, \mathbf{Z} = \mathbf{z}) = \theta_{\text{STOP}} \prod_{i=1}^{|\mathbf{z}|} \left[(1 - \theta_{\text{STOP}}) \, \theta_{z_i, \text{SEGMENT}} \exp(-|z_i|^{1.6}) \right]$$

Segmentation Data and Evaluation

We train and test on the phonetic version of the Bernstein-Ratner corpus (Bernstein-Ratner 1987). This is the same set-up used by Liang and Klein (2009), Goldwater, Griffiths, and Johnson (2006), and Johnson and Goldwater (2009). This corpus consists of 9790 child-directed utterances transcribed using a phonetic representation. We measure segment F1 score on the entire corpus.

We run all word segmentation models for 300 iterations with 10 random initializations and report the mean and standard deviation of F1 in Table 5.1.

Segmentation Features

The SEGMENT multinomial is the important distribution in this model. We use the following features:

BASIC:	$\mathbb{1}(z = \cdot)$
LENGTH:	$\mathbb{1}(\text{length}(z) = \cdot)$
NUMBER-VOWELS:	$\mathbb{1}(\text{numVowels}(z) = \cdot)$
PHONO-CLASS-PREF:	$\mathbb{1}(\text{prefix}(\text{coarsePhonemes}(z)) = \cdot)$
PHONO-CLASS-PREF:	$\mathbb{1}(\text{suffix}(\text{coarsePhonemes}(z)) = \cdot)$

The phonological class prefix and suffix features project each phoneme of a string to a coarser class and then take prefix and suffix indicators on the string of projected characters. We include two versions of these features that use projections with different levels of coarseness. The goal of these features is to help the model learn general phonetic shapes that correspond to well-formed word boundaries.

As is the case in general for our method, the feature-enhanced unigram model still respects the conditional independence assumptions that the standard unigram model makes, and inference is still performed using a simple dynamic program to compute expected sufficient statistics, which are just segment counts.

Segmentation Results

To our knowledge our system achieves the best performance to date on the Bernstein-Ratner corpus, with an F1 of 88.0. It is substantially simpler than the non-parametric Bayesian models proposed by Johnson, Griffiths, et al. (2007), which require sampling procedures to perform inference and achieve an F1 of 87 (Johnson and Goldwater 2009). Similar to our other results, the direct gradient approach outperforms EM for feature-enhanced models, and both approaches outperform the baseline, which achieves an F1 of 76.9.

5.8 Gradient Derivation

In this section, we derive the gradient of the log marginal likelihood needed for the direct gradient approach. Let $\mathbf{w}_0$ be the current weights in Algorithm 2 and $\mathbf{e} = \mathbf{e}(\mathbf{w}_0)$ be the expectations under these weights as computed in Equation 5.2. In order to justify Algorithm 2, we need to prove that $\nabla L(\mathbf{w}_0) = \nabla \ell(\mathbf{w}_0, \mathbf{e})$.

We use the following simple lemma: if ϕ, ψ are real-valued functions such that: (1) $\phi(\mathbf{w}_0) = \psi(\mathbf{w}_0)$ for some $\mathbf{w}_0$; (2) $\phi(\mathbf{w}) \leq \psi(\mathbf{w})$ on an open set containing $\mathbf{w}_0$; and (3), ϕ and ψ are differentiable at $\mathbf{w}_0$; then $\nabla\psi(\mathbf{w}_0) = \nabla\phi(\mathbf{w}_0)$.

We set $\psi(\mathbf{w}) = L(\mathbf{w})$ and $\phi(\mathbf{w}) = \ell(\mathbf{w}, \mathbf{e}) - \sum_{\mathbf{z}} P_{\mathbf{w}_0}(\mathbf{Z} = \mathbf{z}|\mathbf{Y} = \mathbf{y}) \log P_{\mathbf{w}_0}(\mathbf{Z} = \mathbf{z}|\mathbf{Y} = \mathbf{y})$. If we can show that ψ, ϕ satisfy the conditions of the lemma we are done since the second term of ϕ depends on $\mathbf{w}_0$, but not on $\mathbf{w}$.

Property (3) can be easily checked, and property (2) follows from Jensen's inequality. Finally, property (1) follows from Lemma 2 of Neal and Hinton (1998).

Chapter 6

Phylogenetic Grammar Induction

6.1 Overview

Learning multiple languages together should be easier than learning them separately. In this chapter, we apply this notion to the problem of grammar induction by extending the techniques presented in Chapter 5 to build linguistically-motivated multilingual prior.

In the domain of syntactic parsing, a range of recent work has exploited the mutual constraint between two languages' parses of the same bitext (Kuhn 2004; Burkett and Klein 2008; Ganchev et al. 2009; D. A. Smith and Eisner 2009; Snyder, Naseem, and Barzilay 2009). Moreover, Snyder, Naseem, Eisenstein, et al. (2009) in the context of unsupervised part-of-speech induction (and Bouchard-Côté et al. (2007) in the context of phonology) show that extending beyond two languages can provide increasing benefit. However, multitexts are only available for limited languages and domains. In this chapter, we consider unsupervised grammar induction without bitexts or multitexts. Without translation examples, multilingual constraints cannot be exploited at the sentence token level. Rather, we capture multilingual constraints at a *parameter* level by using a phylogeny-structured prior to tie together the various individual languages' learning problems. Our joint, hierarchical prior couples model parameters for different languages in a way that respects knowledge about how the languages evolved.

Aspects of this chapter are closely related to Cohen and N. A. Smith (2009) and Bouchard-Côté et al. (2007). Cohen and N. A. Smith (2009) present a model for jointly learning English and Chinese dependency grammars without bitexts. In their work, structurally constrained covariance in a logistic normal prior is used to couple parameters between the two languages. Our work, though also different in technical approach, differs most centrally in the extension to multiple languages and the use of a phylogeny. Bouchard-Côté et al. (2007) considers an entirely different problem, phonological reconstruction, but shares with this chapter both the use of a phylogenetic structure as well as the use of log-linear parameterization of local model components. Our work differs from theirs primarily in the task (syntax vs. phonology) and the variables governed by the phylogeny: in our model it is the grammar parameters

that drift (in the prior) rather than individual word forms (in the likelihood model).

Specifically, we consider the same dependency model we extended in Section 5.5, the DMV model of Klein and Manning (2004). Our data is a collection of standard dependency data sets in eight languages: English, Dutch, Danish, Swedish, Spanish, Portuguese, Slovene, and Chinese. Our focus is not the DMV model itself, which is well-studied, but rather the prior which couples the various languages' parameters. While some choices of prior structure can greatly complicate inference (Cohen and N. A. Smith 2009), we choose a hierarchical Gaussian form for the drift term, which allows the gradient of the observed data likelihood to be easily computed using standard dynamic programming methods.

In our experiments, joint multilingual learning substantially outperforms independent monolingual learning. Using a limited phylogeny that only couples languages within linguistic families reduces error by 5.6% over the monolingual baseline. Using a flat, global phylogeny gives a greater reduction, almost 10%. Finally, a more articulated phylogeny that captures both inter- and intra-family effects gives an even larger average relative error reduction of 21.1%.

6.2 Model

We define our model over two kinds of random variables: dependency trees and parameters. For each language ℓ in a set L, our model will generate a collection $\mathbf{t}_\ell$ of dependency trees $\mathbf{t}_\ell^i$. We assume that these dependency trees are generated by the DMV model of Klein and Manning (2004), which we write as $\mathbf{t}_\ell^i \sim \text{DMV}(\theta_\ell)$. Here, θ_ℓ is a vector of the various model parameters for language ℓ. The prior is what couples the θ_ℓ parameter vectors across languages; it is the focus of this chapter. We first consider the likelihood model before moving on to the prior.

Dependency Model with Valence

The DMV is also described in brief in Section 5.5. We describe it here in more detail, using new notation that will facilitate our description of the phylogenetic prior. A dependency parse is a directed tree $\mathbf{t}$ over tokens in a sentence $\mathbf{s}$. Each edge of the tree specifies a directed dependency from a head token to a dependent, or argument token. The DMV is a generative model for trees $\mathbf{t}$, which has been widely used for dependency parse induction. The observed data likelihood, used for parameter estimation, is the marginal probability of generating the observed sentences $\mathbf{s}$, which are simply the leaves of the trees $\mathbf{t}$. Generation in the DMV model involves two types of local conditional probabilities: STOP distributions that capture valence and ATTACH distributions that capture argument selection.

First, the Bernoulli STOP probability distributions $P^{\text{STOP}}(c|h, \delta, adj; \theta_\ell)$ model the fertility of a particular head type h. The outcome $c \in \{stop, continue\}$ is conditioned on the head type h, direction δ, and adjacency adj. If a head type's continue probability is low, tokens of this type will tend to generate few arguments.

Second, the ATTACH multinomial probability distributions $P^{\text{ATTACH}}(a|h,\delta;\theta_\ell)$ capture attachment preferences of heads, where a and h are both token types. We take the same approach as previous work (Klein and Manning 2004; Cohen and N. A. Smith 2009) and use gold part-of-speech labels as tokens. Thus, the basic observed "word" types are actually word classes.

Log-Linear Parameterization

The DMV's local conditional distributions were originally given as simple multinomial distributions with one parameter per outcome. However, they can be re-parameterized to give the following log-linear form using the approach we developed in Chapter 5:

$$P^{\text{STOP}}(c|h,\delta,adj;\theta_\ell) = \frac{\exp\left[\langle\theta_\ell, \mathbf{f}_{\text{STOP}}(c,h,\delta,adj)\rangle\right]}{\sum_{c'}\exp\left[\langle\theta_\ell, \mathbf{f}_{\text{STOP}}(c',h,\delta,adj)\rangle\right]}$$

$$P^{\text{ATTACH}}(a|h,\delta;\theta_\ell) = \frac{\exp\left[\langle\theta_\ell, \mathbf{f}_{\text{ATTACH}}(a,h,\delta)\rangle\right]}{\sum_{a'}\exp\left[\langle\theta_\ell, \mathbf{f}_{\text{ATTACH}}(a',h,\delta)\rangle\right]}$$

The parameters are weights θ_ℓ with one weight vector per language. In the case where the vector of feature functions $\mathbf{f}$ has an indicator for each possible conjunction of outcome and conditions, the original multinomial distributions are recovered. We refer to these full indicator features as the set of BASIC features.

Phylogenetic Prior

The focus of this chapter is coupling each of the parameters θ_ℓ in a phylogeny-structured prior. Consider a phylogeny like the one shown in Figure 6.1, where each modern language ℓ in L is a leaf. We would like to say that the leaves' parameter vectors arise from a process which slowly drifts along each branch. A convenient choice is to posit additional parameter variables θ_{ℓ^+} at internal nodes $\ell^+ \in L^+$, a set of ancestral languages, and to assume that the conditional distribution $P(\theta_\ell|\theta_{\text{par}(\ell)})$ at each branch in the phylogeny is a Gaussian centered on $\theta_{\text{par}(\ell)}$, where $\text{par}(\ell)$ is the parent of ℓ in the phylogeny and ℓ ranges over $L \cup L^+$. The variance structure of the Gaussian would then determine how much drift (and in what directions) is expected. Concretely, we assume that each drift distribution is an isotropic Gaussian with mean $\theta_{\text{par}(\ell)}$ and scalar variance σ^2. The root is centered at zero. We have thus defined a joint distribution $P(\Theta|\sigma^2)$ where $\Theta = (\theta_\ell : \ell \in L \cup L^+)$. σ^2 is a hyperparameter for this prior which could itself be re-parameterized to depend on branch length or be learned; we simply set it to a plausible constant value.

Two primary challenges remain. First, inference under arbitrary priors can become complex. However, in the simple case of our diagonal covariance Gaussians, the gradient of the observed data likelihood can be computed directly using the DMV's expected counts and maximum-likelihood estimation can be accomplished by applying standard gradient optimization methods. Second, while the choice of diagonal covariance is efficient, it causes

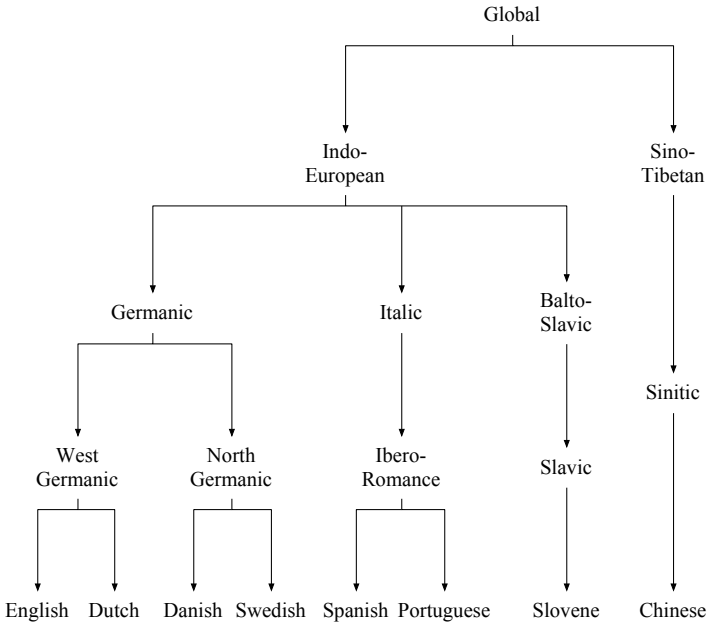


Figure 6.1: An example of a linguistically-plausible phylogenetic tree over the languages in our training data. Leaves correspond to (observed) modern languages, while internal nodes represent (unobserved) ancestral languages. It is unknown whether Indo-European and Sino-Tibetan actually share a common ancestor, but since all languages have some degree of commonality, it will be useful to posit a global ancestor in our model.

components of θ that correspond to features occurring in only one language to be marginally independent of the parameters of all other languages. In other words, only features which fire in more than one language are coupled by the prior. In the next section, we therefore increase the overlap between languages' features by using coarse projections of parts-of-speech.

Projected Features

With diagonal covariance in the Gaussian drift terms, each parameter evolves independently of the others. Therefore, our prior will be most informative when features activate in multiple languages. In phonology, it is useful to map phonemes to the International Phonetic Alphabet (IPA) in order to have a language-independent parameterization. We introduce a similarly neutral representation here by projecting language-specific parts-of-speech to a coarse, shared inventory.

Indeed, we assume that each language has a distinct tagset, and so the basic configurational features will be language specific. For example, when an English VBZ takes a left argument headed by a NNS, a feature will activate specific to VBZ-NNS-LEFT. That feature will be used in the log-linear attachment probability for English. However, because that feature does not show up in any other language, it is not usefully controlled by the prior.

BASIC:	Activate for only one conjunction of outcome and conditions: $\mathbb{1}(c = \cdot, h = \cdot, \delta = \cdot, adj = \cdot)$
SHARED:	Activate for heads from multiple languages using cross-lingual POS projection $\pi(\cdot)$: $\mathbb{1}(c = \cdot, \pi(h) = \cdot, \delta = \cdot, adj = \cdot)$

STOP distribution feature templates.

BASIC:	Activate for only one conjunction of outcome and conditions: $\mathbb{1}(a = \cdot, h = \cdot, \delta = \cdot)$
SHARED:	Activate for heads and arguments from multiple languages using cross-lingual POS projection $\pi(\cdot)$: $\mathbb{1}(\pi(a) = \cdot, \pi(h) = \cdot, \delta = \cdot)$ $\mathbb{1}(\pi(a) = \cdot, h = \cdot, \delta = \cdot)$ $\mathbb{1}(a = \cdot, \pi(h) = \cdot, \delta = \cdot)$

ATTACH distribution feature templates.

Table 6.1: Feature templates for STOP and ATTACH conditional distributions.

Therefore, we also include coarser features which activate on more abstract, cross-linguistic configurations. In the same example, a feature will fire indicating a coarse, direction-free NOUN-VERB attachment. This feature will now occur in multiple languages and will contribute to each of those languages' attachment models. Although such cross-lingual features will have different weight parameters in each language, those weights will covary, being correlated by the prior.

The coarse features are defined via a projection π from language-specific part-of-speech labels to coarser, cross-lingual word classes, and hence we refer to them as SHARED features. For each corpus used in this chapter, we use the tagging annotation guidelines to manually define a fixed mapping from the corpus tagset to the following coarse tagset: noun, verb, adjective, adverb, conjunction, preposition, determiner, interjection, numeral, and pronoun. Parts-of-speech for which this coarse mapping is ambiguous or impossible are not mapped, and do not have corresponding SHARED features.

We summarize the feature templates for the STOP and ATTACH conditional distributions in Table 6.1. Variants of all feature templates that ignore direction and/or adjacency are included. In practice, we found it beneficial for all language-independent features to ignore direction.

Again, only the coarse features occur in multiple languages, so all phylogenetic influence is through those. Nonetheless, the effect of the phylogeny turns out to be quite strong.

Learning

We now turn to learning with the phylogenetic prior. Since the prior couples parameters across languages, this learning problem requires parameters for all languages be estimated jointly. We seek to find $\Theta = (\theta_\ell : \ell \in L \cup L^+)$ which optimizes $\log P(\Theta|\mathbf{s})$, where $\mathbf{s}$ aggregates

the observed leaves of all the dependency trees in all the languages. This can be written as

$$\log P(\Theta) + \log P(\mathbf{s}|\Theta) - \log P(\mathbf{s})$$

The third term is a constant and can be ignored. The first term can be written as

$$\log P(\Theta) = \sum_{\ell \in L \cup L^+} \frac{1}{2\sigma^2} \|\theta_\ell - \theta_{\text{par}(\ell)}\|_2^2 + C$$

where C is a constant. The form of $\log P(\Theta)$ immediately shows how parameters are penalized for being different across languages, more so for languages that are near each other in the phylogeny. The second term

$$\log P(\mathbf{s}|\Theta) = \sum_{\ell \in L} \log P(\mathbf{s}_\ell|\theta_\ell)$$

is a sum of observed data likelihoods under the standard DMV models for each language, computable by dynamic programming (Klein and Manning 2004). Together, this yields the following objective function:

$$l(\Theta) = \sum_{\ell \in L \cup L^+} \frac{1}{2\sigma^2} \|\theta_\ell - \theta_{\text{par}(\ell)}\|_2^2 + \sum_{\ell \in L} \log P(\mathbf{s}_\ell|\theta_\ell)$$

which can be optimized using gradient methods or (MAP) EM. Here we used L-BFGS (Liu and Nocedal 1989). This requires computation of the gradient of the observed data likelihood $\log P(\mathbf{s}_\ell|\theta_\ell)$ which is given by:

$$\nabla \log P(\mathbf{s}_\ell|\theta_\ell) = \mathbb{E}_{\mathbf{t}_\ell|\mathbf{s}_\ell}\left[\nabla \log P(\mathbf{s}_\ell, \mathbf{t}_\ell|\theta_\ell)\right] =$$

$$\begin{bmatrix} \sum_{c,h,\delta,adj} e_{c,h,\delta,adj}(\mathbf{s}_\ell;\theta_\ell) \cdot \left[\mathbf{f}_{\text{STOP}}(c,h,\delta,adj) \; - \sum_{c'} P^{\text{STOP}}(c'|h,\delta,adj;\theta_\ell)\mathbf{f}_{\text{STOP}}(c',h,\delta,adj)\right] \\ \\ \sum_{a,h,\delta} e_{a,h,\delta}(\mathbf{s}_\ell;\theta_\ell) \cdot \left[\mathbf{f}_{\text{ATTACH}}(a,h,\delta) \; - \sum_{a'} P^{\text{ATTACH}}(a'|h,\delta;\theta_\ell)\mathbf{f}_{\text{ATTACH}}(a',h,\delta)\right] \end{bmatrix}$$

The expected gradient of the log joint likelihood of sentences and parses is equal to the gradient of the log marginal likelihood of just sentences, or the observed data likelihood (Salakhutdinov et al. 2003). $e_{a,h,\delta}(\mathbf{s}_\ell;\theta_\ell)$ is the expected count of the number of times head h is attached to a in direction δ given the observed sentences $\mathbf{s}_\ell$ and DMV parameters θ_ℓ. $e_{c,h,\delta,adj}(\mathbf{s}_\ell;\theta_\ell)$ is defined similarly. Note that these are the same expected counts required to perform EM on the DMV, and are computable by dynamic programming.

The computation time is dominated by the computation of each sentence's posterior expected counts, which are independent given the parameters, so the time required per iteration is essentially the same whether training all languages jointly or independently. In practice, the total number of iterations was also similar.

6.3 Experimental Setup

Data

We ran experiments with the following languages: English, Dutch, Danish, Swedish, Spanish, Portuguese, Slovene, and Chinese. For all languages but English and Chinese, we used corpora from the 2006 CoNLL-X Shared Task dependency parsing data set (Buchholz and Marsi 2006). We used the shared task training set to both train and test our models. These corpora provide hand-labeled part-of-speech tags (except for Dutch, which is automatically tagged) and provide dependency parses, which are either themselves hand-labeled or have been converted from hand-labeled parses of other kinds. For English and Chinese we use sections 2-21 of the Penn Treebank (PTB) (Marcus et al. 1993) and sections 1-270 of the Chinese Treebank (CTB) (Xue et al. 2002) respectively. Similarly, these sections were used for both training and testing. The English and Chinese data sets have hand-labeled constituency parses and part-of-speech tags, but no dependency parses. We used the Bikel Chinese head finder (Bikel and Chiang 2000) and the Collins English head finder (Collins 1999) to transform the gold constituency parses into gold dependency parses. None of the corpora are bitexts. For all languages, we ran experiments on all sentences of length 10 or less after punctuation has been removed.

When constructing phylogenies over the languages we made use of their linguistic classifications. English and Dutch are part of the West Germanic family of languages, whereas Danish and Swedish are part of the North Germanic family. Spanish and Portuguese are both part of the Ibero-Romance family. Slovene is part of the Slavic family. Finally, Chinese is in the Sinitic family, and is not an Indo-European language like the others. We interchangeably speak of a language family and the ancestral node corresponding to that family's root language in a phylogeny.

Models Compared

We evaluated three phylogenetic priors, each with a different phylogenetic structure. We compare with two monolingual baselines, as well as an all-pairs multilingual model that does not have a phylogenetic interpretation, but which provides very similar capacity for parameter coupling.

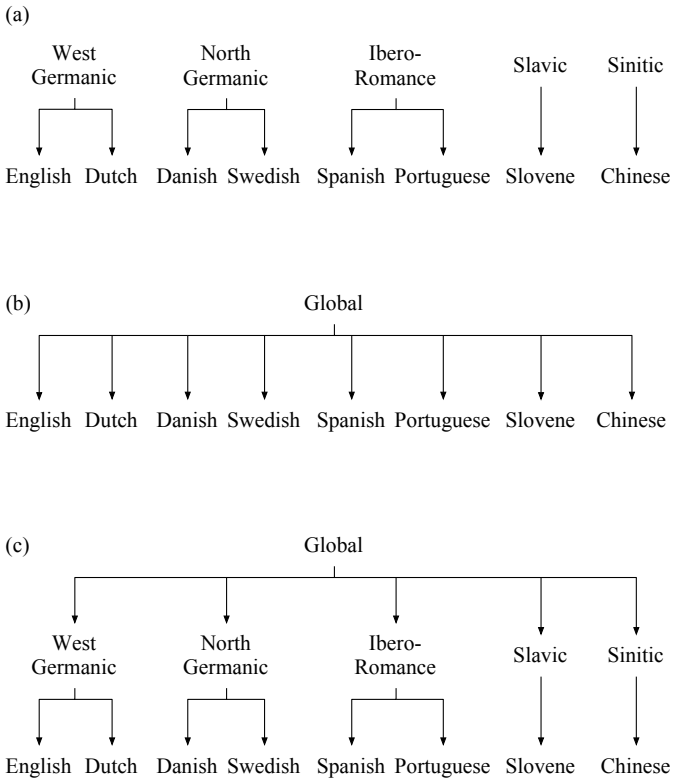


Figure 6.2: (a) Phylogeny for FAMILIES model. (b) Phylogeny for GLOBAL model. (c) Phylogeny for LINGUISTIC model.

Phylogenetic Models

The first phylogenetic model uses the shallow phylogeny shown in Figure 6.2(a), in which only languages within the same family have a shared parent node. We refer to this structure as FAMILIES. Under this prior, the learning task decouples into independent subtasks for each family, but no regularities across families can be captured.

The family-level model misses the constraints between distant languages. Figure 6.2(b) shows another simple configuration, wherein all languages share a common parent node in the prior, meaning that global regularities that are consistent across all languages can be captured. We refer to this structure as GLOBAL.

While the global model couples the parameters for all eight languages, it does so without sensitivity to the articulated structure of their descent. Figure 6.2(c) shows a more nuanced prior structure, LINGUISTIC, which groups languages first by family and then under a global node. This structure allows global regularities as well as regularities within families to be learned.

Parameterization and AllPairs Model

Daumé III (2007) and Finkel and Manning (2009) consider a formally similar Gaussian hierarchy for domain adaptation. As pointed out in Finkel and Manning (2009), there is a simple equivalence between hierarchical regularization as described here and the addition of new tied features in a "flat" model with zero-meaned Gaussian regularization on all parameters. In particular, instead of parameterizing the objective in Section 6.2 in terms of multiple sets of weights, one at each node in the phylogeny (the *hierarchical parameterization*, described in Section 6.2), it is equivalent to parameterize this same objective in terms of a single set of weights on a larger of group features (the *flat parameterization*). This larger group of features contains a duplicate set of the features discussed in Section 6.2 for each node in the phylogeny, each of which is active only on the languages that are its descendants. A linear transformation between parameterizations gives equivalence. See Finkel and Manning (2009) for details.

In the flat parameterization, it seems equally reasonable to simply tie all pairs of languages by adding duplicate sets of features for each pair. This gives the AllPairs setting, which we also compare to the tree-structured phylogenetic models above.

Baselines

To evaluate the impact of multilingual constraint, we compared against two monolingual baselines. The first baseline is the standard DMV with only Basic features, which yields the standard multinomial DMV (*weak baseline*). To facilitate comparison to past work, we used no prior for this monolingual model. The second baseline is the DMV with added Shared features. This model includes a simple isotropic Gaussian prior on parameters. This second baseline is the more direct comparison to the multilingual experiments here (*strong baseline*).

Evaluation

For each setting, we evaluated the directed dependency accuracy of the minimum Bayes risk (MBR) dependency parses produced by our models under maximum (posterior) likelihood parameter estimates. We computed accuracies separately for each language in each condition. In addition, for multilingual models, we computed the relative error reduction over the strong monolingual baseline, macro-averaged over languages.

Training

Our implementation used the flat parameterization described in Section 6.3 for both the phylogenetic and AllPairs models. We originally did this in order to facilitate comparison with the non-phylogenetic AllPairs model, which has no equivalent hierarchical param-

eterization. In practice, optimizing with the hierarchical parameterization also seemed to underperform.[1]

All models were trained by directly optimizing the observed data likelihood using L-BFGS (Liu and Nocedal 1989). In Chapter 5 we found that directly optimizing the observed data likelihood may offer improvements over the more standard expectation-maximization (EM) optimization procedure for models such as the DMV, especially when the model is parameterized using features. We stopped training after 200 iterations in all cases. This fixed stopping criterion seemed to be adequate in all experiments, but presumably there is a potential gain to be had in fine tuning. To initialize, we used the harmonic initializer presented in Klein and Manning (2004). This type of initialization is deterministic, and thus we did not perform random restarts.

We found that for all models $\sigma^2 = 0.2$ gave reasonable results, and we used this setting in all experiments. For most models, we found that varying σ^2 in a reasonable range did not substantially affect accuracy. For some models, the directed accuracy was less flat with respect to σ^2. In these less-stable cases, there seemed to be an interaction between the variance and the choice between head conventions. For example, for some settings of σ^2, but not others, the model would learn that determiners head noun phrases. In particular, we observed that even when direct accuracy did fluctuate, undirected accuracy remained more stable.

6.4 Results

Table 6.2 shows the overall results. In all cases, methods which coupled the languages in some way outperformed the independent baselines that considered each language independently.

Bilingual Models

The weakest of the coupled models was Families, which had an average relative error reduction of 5.6% over the strong baseline. In this case, most of the average improvement came from a single family: Spanish and Portuguese. The limited improvement of the family-level prior compared to other phylogenies suggests that there are important multilingual interactions that do not happen within families. Table 6.2 also reports the maximum accuracy achieved for each language when it was paired with another language (same family or otherwise) and trained together with a single common parent. These results appear in the column headed by BestPair, and show the best accuracy for the language on that row over all possible pairings with other languages. When pairs of languages were trained together in isolation, the largest benefit was seen for languages with small training corpora, not necessarily languages with common ancestry. In our setup, Spanish, Slovene, and Chinese

[1]We noticed that the weights of features shared across languages had larger magnitude early in the optimization procedure when using the flat parameterization compared to using the hierarchical parameterization, perhaps indicating that cross-lingual influences had a larger effect on learning in its initial stages.

have substantially smaller training corpora than the rest of the languages considered. Otherwise, the patterns are not particularly clear; combined with subsequent results, it seems that pairwise constraint is fairly limited.

Multilingual Models

Models that coupled multiple languages performed better in general than models that only considered pairs of languages. The GLOBAL model, which couples all languages, if crudely, yielded an average relative error reduction of 9.9%. This improvement comes as the number of languages able to exert mutual constraint increases. For example, Dutch and Danish had large improvements, over and above any improvements these two languages gained when trained with a single additional language. Beyond the simplistic GLOBAL phylogeny, the more nuanced LINGUISTIC model gave large improvements for English, Swedish, and Portuguese. Indeed, the LINGUISTIC model is the only model we evaluated that gave improvements for *all* the languages we considered.

It is reasonable to worry that the improvements from these multilingual models might be partially due to having more total training data in the multilingual setting. However, we found that halving the amount of data used to train the English, Dutch, and Swedish (the languages with the most training data) monolingual models did not substantially affect their performance, suggesting that for languages with several thousand sentences or more, the increase in statistical support due to additional monolingual data was not an important effect (the DMV is a relatively low-capacity model in any case).

Comparison of Phylogenies

Recall the structures of the three phylogenies presented in Figure 6.2. These phylogenies differ in the correlations they can represent. The GLOBAL phylogeny captures only "universals," while FAMILIES captures only correlations between languages that are known to be similar. The LINGUISTIC model captures both of these effects simultaneously by using a two layer hierarchy. Notably, the improvement due to the LINGUISTIC model is more than the sum of the improvements due to the GLOBAL and FAMILIES models.

Phylogenetic vs. AllPairs

The phylogeny is capable of allowing appropriate influence to pass between languages at multiple levels. We compare these results to the ALLPAIRS model in order to see whether limitation to a tree structure is helpful. The ALLPAIRS model achieved an average relative error reduction of 17.1%, certainly outperforming both the simple phylogenetic models. However, the rich phylogeny of the LINGUISTIC model, which incorporates linguistic constraints, outperformed the freer ALLPAIRS model. A large portion of this improvement came from English, a language for which the LINGUISTIC model greatly outperformed all other models evaluated. We found that the improved English analyses produced by the LINGUISTIC model

			Monolingual		Multilingual				
						Phylogenetic			
		Corpus Size	Baseline	Baseline w/ SHARED	ALLPAIRS	FAMILIES	BESTPAIR	GLOBAL	LINGUISTIC
West Germanic	English	6008	47.1	51.3	48.5	51.3	51.3 (Ch)	51.2	**62.3**
	Dutch	6678	36.3	36.0	44.0	36.1	36.2 (Sw)	44.0	**45.1**
North Germanic	Danish	1870	33.5	33.6	40.5	31.4	34.2 (Du)	39.6	**41.6**
	Swedish	3571	45.3	44.8	56.3	44.8	44.8 (Ch)	44.5	**58.3**
Ibero-Romance	Spanish	712	28.0	40.5	58.7	63.4	**63.8** (Da)	59.4	58.4
	Portuguese	2515	38.5	38.5	**63.1**	37.4	38.4 (Sw)	37.4	63.0
Slavic	Slovene	627	38.5	39.7	49.0	–	**49.6** (En)	49.4	48.4
Sinitic	Chinese	959	36.3	43.3	**50.7**	–	49.7 (Sw)	50.1	49.6
	Macro-Avg. Relative Error Reduction				17.1	5.6	8.5	9.9	**21.1**

Table 6.2: Directed dependency accuracy of monolingual and multilingual models, and relative error reduction over the monolingual baseline with SHARED features macro-averaged over languages. Multilingual models outperformed monolingual models in general, with larger gains from increasing numbers of languages. Additionally, more nuanced phylogenetic structures outperformed cruder ones.

were more consistent with this model's analyses of other languages. This consistency was not present for the English analyses produced by other models. We explore consistency in more detail in Section 6.5.

Comparison to Related Work

The likelihood models for both the strong monolingual baseline and the various multilingual models are the same, both expanding upon the standard DMV by adding coarse SHARED features. As we saw in Chapter 5, these coarse features, even in a monolingual setting, improved performance slightly over the weak baseline, perhaps by encouraging consistent treatment of the different finer-grained variants of parts-of-speech.[2] The only difference between the multilingual systems and the strong baseline is whether or not cross-language influence is allowed through the prior.

While this progression of model structure is similar to that explored in Cohen and N. A. Smith (2009), Cohen and N. A. Smith (2009) saw their largest improvements from tying together parameters for the varieties of coarse parts-of-speech monolinugally, and then only

[2]Coarse features that only tie nouns and verbs are explored in Chapter 5. We found that these were very effective for English and Chinese, but gave worse performance for other languages.

moderate improvements from allowing cross-linguistic influence on top of monolingual sharing. When Cohen and Smith compared their best shared logistic-normal bilingual models to monolingual counter-parts for the languages they investigate (Chinese and English), they reported a relative error reduction of 5.3%. In comparison, with the LINGUISTIC model, we saw a much larger 16.9% relative error reduction over our strong baseline for these languages. Evaluating our LINGUISTIC model on the same test sets as (Cohen and N. A. Smith 2009), sentences of length 10 or less in section 23 of PTB and sections 271-300 of CTB, we achieved an accuracy of 56.6 for Chinese and 60.3 for English. The best models of Cohen and N. A. Smith (2009) achieved accuracies of 52.0 and 62.0 respectively on these same test sets.

Our results indicate that the majority of our model's power beyond that of the standard DMV is derived from multilingual, and in particular, more-than-bilingual, interaction. These are, to the best of our knowledge, the first results of this kind for grammar induction without bitext.

6.5 Analysis

By examining the proposed parses we found that the LINGUISTIC and ALLPAIRS models produced analyses that were more consistent across languages than those of the other models. We also observed that the most common errors can be summarized succinctly by looking at attachment counts between coarse parts-of-speech. Figure 6.3 shows matrix representations of dependency counts. The area of a square is proportional to the number of order-collapsed dependencies where the column label is the head and the row label is the argument in the parses from each system. For ease of comprehension, we use the cross-lingual projections and only show counts for selected interesting classes.

Comparing Figure 6.3(c), which shows dependency counts proposed by the LINGUISTIC model, to Figure 6.3(a), which shows the same for the strong monolingual baseline, suggests that the analyses proposed by the LINGUISTIC model are more consistent across languages than are the analyses proposed by the monolingual model. For example, the monolingual learners are divided as to whether determiners or nouns head noun phrases. There is also confusion about which labels head whole sentences. Dutch has the problem that verbs modify pronouns more often than pronouns modify verbs, and pronouns are predicted to head sentences as often as verbs are. Spanish has some confusion about conjunctions, hypothesizing that verbs often attach to conjunctions, and conjunctions frequently head sentences. More subtly, the monolingual analyses are inconsistent in the way they head prepositional phrases. In the monolingual Portuguese hypotheses, prepositions modify nouns more often than nouns modify prepositions. In English, nouns modify prepositions, and prepositions modify verbs. Both the Dutch and Spanish models are ambivalent about the attachment of prepositions.

As has often been observed in other contexts (Liang, Klein, and Jordan 2008), promoting agreement can improve accuracy in unsupervised learning. Not only are the analyses proposed by the LINGUISTIC model more consistent, they are also more in accordance with the

gold analyses. Under the LINGUISTIC model, Dutch now attaches pronouns to verbs, and thus looks more like English, its sister in the phylogenetic tree. The LINGUISTIC model has also chosen consistent analyses for prepositional phrases and noun phrases, calling prepositions and nouns the heads of each, respectively. The problem of conjunctions heading Spanish sentences has also been corrected.

Figure 6.3(b) shows dependency counts for the GLOBAL multilingual model. Unsurprisingly, the analyses proposed under global constraint appear somewhat more consistent than those proposed under no multi-lingual constraint (now three languages agree that prepositional phrases are headed by prepositions), but not as consistent as those proposed by the LINGUISTIC model.

Finally, Figure 6.3(d) shows dependency counts in the hand-labeled dependency parses. It appears that even the very consistent LINGUISTIC parses do not capture the non-determinism of prepositional phrase attachment to both nouns and verbs.

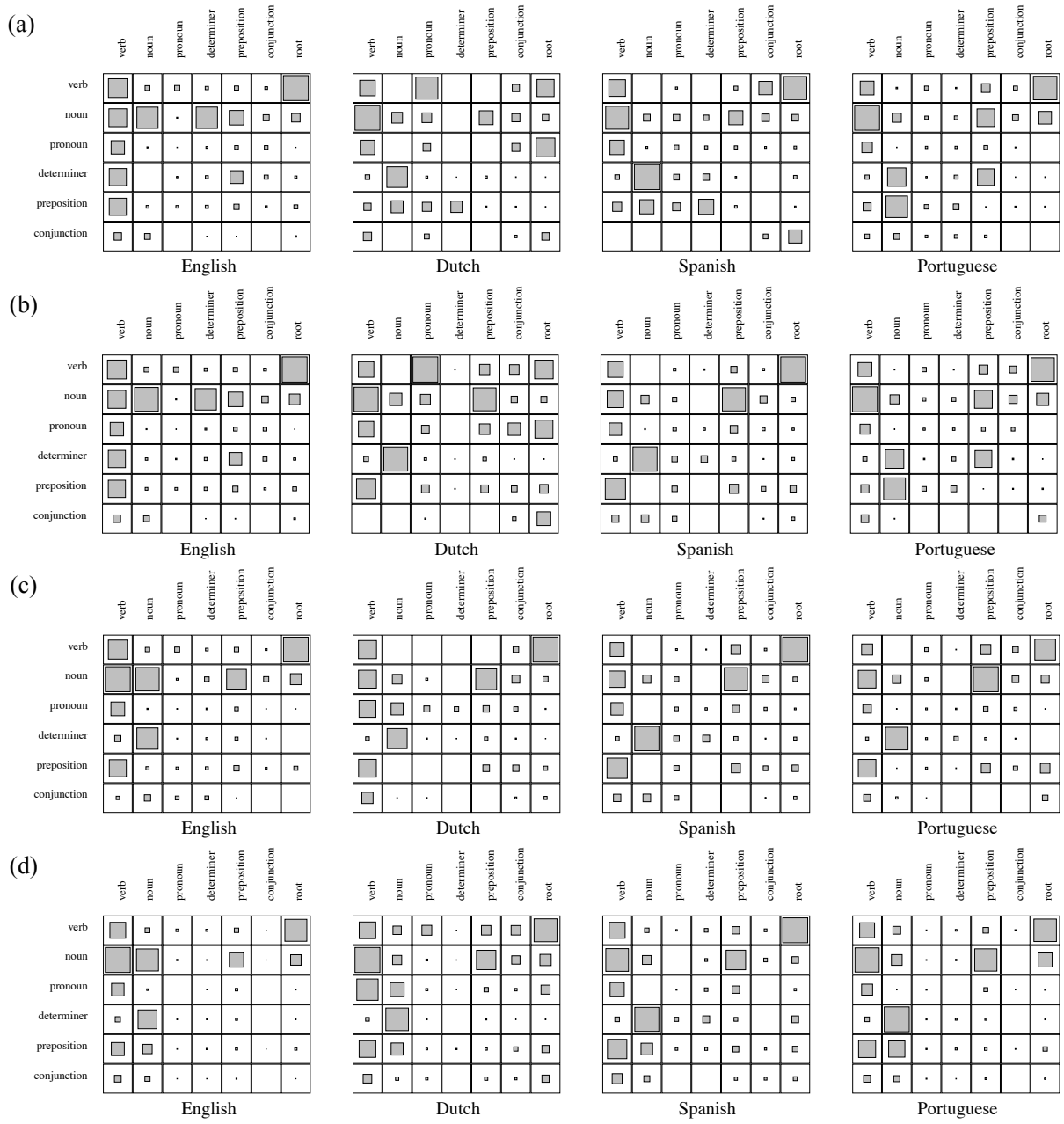


Figure 6.3: Dependency counts in proposed parses. Row label modifies column label. (a) Monolingual baseline with SHARED features. (b) GLOBAL model. (c) LINGUISTIC model. (d) Dependency counts in hand-labeled parses. Analyses proposed by monolingual baseline show significant inconsistencies across languages. Analyses proposed by LINGUISTIC model are more consistent across languages than those proposed by either the monolingual baseline or the GLOBAL model.

Chapter 7

Conclusions

This thesis has investigated several new models and priors for unsupervised analysis of different types of human data. We demonstrated a model, based on the historical typesetting process, that effectively learns font structure in an unsupervised fashion to improve transcription of historical documents into text. The parameters of the learned fonts are interpretable, as are the predicted typesetting layouts. We have also introduced a new generative model for the task of unsupervised piano transcription that surpasses state-of-the-art performance on a standard benchmark dataset by directly modeling the interaction between the discrete musical structure and the process that generates acoustic signals. We saw improvements for both tasks as a result of removing a harmful independence assumptions and building more articulated models. This suggests that it may be worthwhile to consider still further extensions of these types of model, designed to more faithfully reflect the generative process that produced the data.

Focusing on refining model priors, we have shown that simple, locally normalized models can effectively incorporate features into unsupervised models. These enriched models can be easily optimized using standard NLP building blocks. Using this approach, we found that multilingual constraints expressed in the form of a phylogenetic prior can give substantial gains in grammar induction accuracy over treating languages in isolation. Additionally, articulated phylogenies that are sensitive to evolutionary structure can outperform not only limited flatter priors but also unconstrained all-pairs interactions. Beyond the four tasks explored in this thesis—POS tagging, DMV grammar induction, word alignment, and word segmentation—the method of using features to refine unsupervised priors can be applied to many other tasks, for example grounded semantics, unsupervised PCFG induction, document clustering, and anaphora resolution.

Bibliography

[1] Kenning Arlitsch and John Herbert. "Microfilm, paper, and OCR: Issues in newspaper digitization. The Utah digital newspapers program". In: *Microform & Imaging Review* (2004).

[2] Necip F. Ayan, Bonnie J. Dorr, and Christof Monz. "Combining Word Alignments Using Neural Networks". In: *Proceedings of Empirical Methods in Natural Language Processing*. 2005.

[3] Emmanouil Benetos and Tillman Weyde. "Explicit duration hidden markov models for multiple-instrument polyphonic music transcription". In: *International Society for Information Music Retrieval*. 2013.

[4] Taylor Berg-Kirkpatrick, Jacob Andreas, and Dan Klein. "Unsupervised Transcription of Piano Music". In: *Advances in Neural Information Processing Systems*. 2014.

[5] Taylor Berg-Kirkpatrick, Alexandre Bouchard-Côté, John DeNero, and Dan Klein. "Painless unsupervised learning with features". In: *Proceedings of North American Chapter of the Association for Computational Linguistics*. 2010.

[6] Taylor Berg-Kirkpatrick, Greg Durrett, and Dan Klein. "Unsupervised Transcription of Historical Documents". In: *Proceedings of Association for Computational Linguistics*. 2013.

[7] Taylor Berg-Kirkpatrick and Dan Klein. "Improved typesetting models for historical OCR". In: *Proceedings of Association for Computational Linguistics*. 2014.

[8] Taylor Berg-Kirkpatrick and Dan Klein. "Phylogenetic grammar induction". In: *Proceedings of Association for Computational Linguistics*. 2010.

[9] Taylor Berg-Kirkpatrick and Dan Klein. "Simple effective decipherment via combinatorial optimization". In: *Proceedings of Empirical Methods in Natural Language Processing*. 2011.

[10] Nan Bernstein-Ratner. "The phonology of parent- child speech". In: *Children's Language* (1987).

[11] Julian Besag. "On the statistical analysis of dirty pictures". In: *Journal of the Royal Statistical Society* (1986).

[12] Daniel M. Bikel and David Chiang. "Two statistical parsing models applied to the Chinese Treebank". In: *Second Chinese Language Processing Workshop*. 2000.

[13] Maximilian Bisani and Hermann Ney. "Joint-sequence models for grapheme-to-phoneme conversion". In: *Speech Communication* (2008).

[14] Sebastian Böck and Markus Schedl. "Polyphonic piano note transcription with recurrent neural networks". In: *IEEE International Conference on Acoustics, Speech, and Signal Processing*. 2012.

[15] C.G. Boogaart and Rainer Lienhart. "Note onset detection for the transcription of polyphonic piano music". In: *Multimedia and Expo ICME*. 2009.

[16] Alexandre Bouchard-Côté, Percy Liang, Dan Klein, and Thomas L. Griffiths. "A probabilistic approach to diachronic phonology". In: *Proceedings of Empirical Methods in Natural Language Processing*. 2007.

[17] Alexandre Bouchard-Côté, Percy Liang, Dan Klein, and Thomas L. Griffiths. "A Probabilistic Approach to Language Change". In: *Advances in Neural Information Processing Systems*. 2008.

[18] Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. "The Mathematics of Statistical Machine Translation: Parameter Estimation". In: *Computational Linguistics* (1994).

[19] Sabine Buchholz and Erwin Marsi. "Computational Natural Language Learning-X shared task on multilingual dependency parsing". In: *Proceedings of Conference on Computational Natural Language Learning*. 2006.

[20] David Burkett and Dan Klein. "Two languages are better than one (for syntactic parsing)". In: *Proceedings of Empirical Methods in Natural Language Processing*. 2008.

[21] Stanley F. Chen. "Conditional and joint models for Grapheme-to-phoneme conversion". In: *Eurospeech*. 2003.

[22] Shay B. Cohen and Noah A. Smith. "Shared Logistic Normal Distributions for Soft Parameter Tying in Unsupervised Grammar Induction". In: *Proceedings of North American Chapter of the Association for Computational Linguistics*. 2009.

[23] Michael Collins. "Head-Driven Statistical Models for Natural Language Parsing". In: *Ph.D. thesis, University of Pennsylvania, Philadelphia*. 1999.

[24] Hal Daumé III. "Frustratingly easy domain adaptation". In: *Proceedings of Association for Computational Linguistics*. 2007.

[25] Arthur Dempster, Nan Laird, and Donald Rubin. "Maximum likelihood from incomplete data via the EM algorithm". In: *Journal of the Royal Statistical Society* (1977).

[26] John DeNero and Dan Klein. "Tailoring Word Alignments to Syntactic Machine Translation". In: *Proceedings of Association for Computational Linguistics*. 2007.

[27] David Elworthy. "Does Baum-Welch Re-estimation Help Taggers?" In: *Proceedings of Association for Computational Linguistics*. 1994.

[28] Valentin Emiya, Roland Badeau, and Bertrand David. "Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle". In: *IEEE Transactions on Audio, Speech, and Language Processing* (2010).

[29] Jenny R. Finkel and Christopher D. Manning. "Hierarchical bayesian domain adaptation". In: *Proceedings of North American Chapter of the Association for Computational Linguistics*. 2009.

[30] Kuzman Ganchev, Jennifer Gillenwater, and Ben Taskar. "Dependency Grammar Induction via Bitext Projection Constraints". In: *Proceedings of Association for Computational Linguistics*. 2009.

[31] Ismael Garcia-Varea, Franz Och, Hermann Ney, and Francisco Casacuberta. "Refined lexicon models for statistical machine translation using a maximum entropy approach". In: *Proceedings of Conference on Computational Natural Language Learning*. 2002.

[32] Sharon Goldwater and Thomas L. Griffiths. "A fully Bayesian approach to unsupervised part-of-speech tagging". In: *Proceedings of Association for Computational Linguistics*. 2007.

[33] Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. "Contextual dependencies in unsupervised word segmentation". In: *Proceedings of Association for Computational Linguistics*. 2006.

[34] David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. *English Gigaword Third Edition*. Linguistic Data Consortium, Catalog Number LDC2007T07. 2007.

[35] Aria Haghighi, John Blitzer, John DeNero, and Dan Klein. "Better Word Alignments with Supervised ITG Models". In: *Proceedings of Association for Computational Linguistics*. 2009.

[36] Aria Haghighi and Dan Klein. "Prototype-Driven Learning for Sequence Models". In: *Proceedings of Association for Computational Linguistics*. 2006.

[37] Tin Kam Ho and George Nagy. "OCR with no shape training". In: *International Conference on Pattern Recognition*. 2000.

[38] Rose Holley. "Trove: Innovation in Access to Information in Australia". In: *Ariadne* (2010).

[39] Gary Huang, Erik Learned-Miller, and Andrew McCallum. "Cryptogram decoding for optical character recognition". In: *University of Massachusetts-Amherst Technical Report* (2006).

[40] Fred Jelinek. *Statistical methods for speech recognition*. MIT press, 1998.

[41] Mark Johnson. "Why Doesnt EM Find Good HMM POS-Taggers?" In: *Proceedings of Empirical Methods in Natural Language Processing*. 2007.

[42] Mark Johnson and Sharon Goldwater. "Improving nonparametric Bayesian inference: Experiments on unsupervised word segmentation with adaptor grammars". In: *Proceedings of North American Chapter of the Association for Computational Linguistics*. 2009.

[43] Mark Johnson, Thomas L. Griffiths, and Sharon Goldwater. "Adaptor Grammars: a framework for specifying compositional nonparametric Bayesian models". In: *Advances in Neural Information Processing Systems*. 2007.

[44] Andrew Kae, Gary Huang, Carl Doersch, and Erik Learned-Miller. "Improving state-of-the-art OCR through high-precision document-specific modeling". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2010.

[45] Andrew Kae and Erik Learned-Miller. "Learning on the fly: font-free approaches to difficult OCR problems". In: *International Conference on Document Analysis and Recognition*. 2009.

[46] Jyrki Kivinen and Manfred K. Warmuth. "Exponentiated gradient versus gradient descent for linear predictors". In: *Information and Computation* (1997).

[47] Dan Klein and Christopher D. Manning. "Corpus-Based Induction of Syntactic Structure: Models of Dependency and Constituency". In: *Proceedings of Association for Computational Linguistics*. 2004.

[48] Vladimir Kluzner, Asaf Tzadok, Dan Chevion, and Eugene Walach. "Hybrid Approach to Adaptive OCR for Historical Books". In: *International Conference on Document Analysis and Recognition*. 2011.

[49] Vladimir Kluzner, Asaf Tzadok, Yuval Shimony, Eugene Walach, and Apostolos Antonacopoulos. "Word-based Adaptive OCR for historical books". In: *International Conference on on Document Analysis and Recognition*. 2009.

[50] Reinhard Kneser and Hermann Ney. "Improved backing-off for m-gram language modeling". In: *IEEE International Conference on Acoustics, Speech, and Signal Processing*. 1995.

[51] Philipp Koehn. "Pharaoh: a beam search decoder for phrase-based statistical machine translation models". In: *Machine translation: From real users to research* (2004).

[52] Okan Kolak, William Byrne, and Philip Resnik. "A generative probabilistic OCR model for NLP applications". In: *Proceedings of North American Chapter of the Association for Computational Linguistics*. 2003.

[53] Gary Kopec and Mauricio Lomelin. "Document-specific character template estimation". In: *International Society for Optics and Photonics*. 1996.

[54] Gary Kopec, Maya Said, and Kris Popat. "N-gram language models for document image decoding". In: *Society of Photographic Instrumentation Engineers*. 2001.

[55] Jonas Kuhn. "Experiments in parallel-text based grammar induction". In: *Proceedings of Association for Computational Linguistics*. 2004.

[56] Stephen Levinson. "Continuously variable duration hidden Markov models for automatic speech recognition". In: *Computer Speech & Language* (1986).

[57] Percy Liang and Dan Klein. "Online EM for Unsupervised Models". In: *Proceedings of North American Chapter of the Association for Computational Linguistics*. 2009.

[58] Percy Liang, Dan Klein, and Michael I. Jordan. "Agreement-Based Learning". In: *Advances in Neural Information Processing Systems*. 2008.

[59] Percy Liang, Ben Taskar, and Dan Klein. "Alignment by Agreement". In: *Proceedings of North American Chapter of the Association for Computational Linguistics*. 2006.

[60] Dong C. Liu and Jorge Nocedal. "On the limited memory BFGS method for large scale optimization". In: *Mathematical Programming* (1989).

[61] Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. "Building a large annotated corpus of English: the penn treebank". In: *Computational Linguistics* (1993).

[62] Bernard Merialdo. "Tagging English text with a probabilistic model". In: *Computational Linguistics* (1994).

[63] Masahiro Nakano, Yasunori Ohishi, Hirokazu Kameoka, Ryo Mukai, and Kunio Kashino. "Bayesian nonparametric music parser". In: *IEEE International Conference on Acoustics, Speech, and Signal Processing*. 2012.

[64] Radford Neal and Geoffrey Hinton. "A View Of The EM Algorithm That Justifies Incremental, Sparse, And Other Variants". In: *Learning in Graphical Models*. 1998.

[65] Franz Och and Hermann Ney. "Improved statistical alignment models". In: *Proceedings of Association for Computational Linguistics*. 2000.

[66] Franz Och and Hermann Ney. "The alignment template approach to statistical machine translation". In: *Computational Linguistics* (2004).

[67] Ken O'Hanlon and Mark D. Plumbley. "Polyphonic piano transcription using non-negative matrix factorisation with group sparsity". In: *IEEE International Conference on Acoustics, Speech, and Signal Processing*. 2014.

[68] Paul H. Peeling, Ali Taylan Cemgil, and Simon J. Godsill. "Generative spectrogram factorization models for polyphonic piano transcription". In: *IEEE Transactions on Audio, Speech, and Language Processing* (2010).

[69] Slav Petrov, Aria Haghighi, and Dan Klein. "Coarse-to-fine syntactic machine translation using language projections". In: *Proceedings of Empirical Methods in Natural Language Processing*. 2008.

[70] Graham E. Poliner and Daniel P.W. Ellis. "A discriminative model for polyphonic piano transcription". In: *EURASIP Journal on Advances in Signal Processing* (2006).

[71] Sujith Ravi and Kevin Knight. "Attacking decipherment problems optimally with low-order n-gram models". In: *Proceedings of Empirical Methods in Natural Language Processing*. 2008.

[72] Sujith Ravi and Kevin Knight. "Bayesian inference for Zodiac and other homophonic ciphers". In: *Proceedings of Association for Computational Linguistics*. 2011.

[73] Matti P. Ryynanen and Anssi Klapuri. "Polyphonic music transcription using note event modeling". In: *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*. 2005.

[74] Ruslan Salakhutdinov, Sam Roweis, and Zoubin Ghahramani. "Optimization with EM and Expectation-Conjugate-Gradient". In: *International Conference on Machine Learning*. 2003.

[75] Robert Shoemaker. "Digital London: Creating a searchable web of interlinked sources on eighteenth century London". In: *Electronic Library and Information Systems* (2005).

[76] David A. Smith and Jason Eisner. "Parser Adaptation and Projection with Quasi-Synchronous Grammar Features". In: *Proceedings of Empirical Methods in Natural Language Processing*. 2009.

[77] Noah A. Smith and Jason Eisner. "Contrastive Estimation: Training Log-Linear Models on Unlabeled Data". In: *Proceedings of Association for Computational Linguistics*. 2005.

[78] Noah A. Smith and Mark Johnson. "Weighted and probabilistic context-free grammars are equally expressive". In: *Computational Linguistics* (2007).

[79] Ray Smith. "An overview of the Tesseract OCR engine". In: *Proceedings of International Conference on Document Analysis and Recognition*. 2007.

[80] Benjamin Snyder, Regina Barzilay, and Kevin Knight. "A statistical model for lost language decipherment". In: *Proceedings of Association for Computational Linguistics*. 2010.

[81] Benjamin Snyder, Tahira Naseem, and Regina Barzilay. "Unsupervised multilingual grammar induction". In: *Proceedings of Association for Computational Linguistics*. 2009.

[82] Benjamin Snyder, Tahira Naseem, Jacob Eisenstein, and Regina Barzilay. "Adding more languages improves unsupervised multilingual part-of-speech tagging: A Bayesian non-parametric approach". In: *Proceedings of North American Chapter of the Association for Computational Linguistics*. 2009.

[83] *The International Music Score Library Project*. 2014. URL: `http://imslp.org`.

[84] Georgios Vamvakas, Basilios Gatos, Nikolaos Stamatopoulos, and Stavros Perantonis. "A complete optical character recognition methodology for historical documents". In: *IAPR International Workshop on Document Analysis Systems*. 2008.

[85] Emmanuel Vincent, Nancy Bertin, and Roland Badeau. "Adaptive harmonic spectral decomposition for multiple pitch estimation". In: *IEEE Transactions on Audio, Speech, and Language Processing* (2010).

[86] Vladimir Viro. "Peachnote: Music Score Search and Analysis Platform". In: *International Society for Music Information Retrieval*. 2011.

[87] Stephan Vogel, Hermann Ney, and Christoph Tillmann. "HMM-Based Word Alignment in Statistical Translation". In: *International Conference on Computational Linguistics*. 1996.

[88] Felix Weninger, Christian Kirst, Bjorn Schuller, and Hans-Joachim Bungartz. "A discriminative approach to polyphonic piano note transcription using supervised non-negative matrix factorization". In: *IEEE International Conference on Acoustics, Speech, and Signal Processing*. 2013.

[89] Nianwen Xue, Fu-Dong Chiou, and Martha Palmer. "Building a Large-Scale Annotated Chinese Corpus". In: *International Conference on Computational Linguistics*. 2002.

[90] Hao Zhang and Daniel Gildea. "Efficient multipass decoding for synchronous context free grammars". In: *Proceedings of Empirical Methods in Natural Language Processing*. 2008.