Generative Non-Markov Models for Information Extraction

by

Nicholas Oliver Andrews

A dissertation submitted to The Johns Hopkins University in conformity with the requirements for the degree of Doctor of Philosophy.

Baltimore, Maryland

August, 2015

Abstract

Learning from unlabeled data is a long-standing challenge in machine learning. A principled solution involves modeling the full joint distribution over inputs and the latent structure of interest, and imputing the missing data via marginalization. Unfortunately, such marginalization is expensive for most non-trivial problems, which places practical limits on the expressiveness of generative models. As a result, joint models often encode strict assumptions about the underlying process such as fixed-order Markovian assumptions and employ simple count-based features of the inputs. In contrast, conditional models, which do not directly model the observed data, are free to incorporate rich overlapping features of the input in order to predict the latent structure of interest. It would be desirable to develop expressive generative models that retain tractable inference. This is the topic of this thesis. In particular, we explore joint models which relax fixed-order Markov assumptions, and investigate the use of recurrent neural networks for automatic feature induction in the generative process.

We focus on two structured prediction problems: (1) imputing labeled segmentions of input character sequences, and (2) imputing directed spanning trees relating strings

in text corpora. These problems arise in many applications of practical interest, but we are primarily concerned with named-entity recognition and cross-document coreference resolution in this work.

For named-entity recognition, we propose a generative model in which the observed characters originate from a latent non-Markov process over words, and where the characters are themselves produced via a non-Markov process: a recurrent neural network (RNN). We propose a sampler for the proposed model in which sequential Monte Carlo is used as a transition kernel for a Gibbs sampler. The kernel is amenable to a fast parallel implementation, and results in fast mixing in practice.

For cross-document coreference resolution, we move beyond sequence modeling to consider string-to-string transduction. We stipulate a generative process for a corpus of documents in which entity names arise from copying—and optionally transforming—previous names of the same entity. Our proposed model is sensitive to both the context in which the names occur as well as their spelling. The string-to-string transformations correspond to systematic linguistic processes such as abbreviation, typos, and nicknaming, and by analogy to biology, we think of them as mutations along the edges of a phylogeny. We propose a novel block Gibbs sampler for this problem that alternates between sampling an ordering of the mentions and a spanning tree relating all mentions in the corpus.

Readers: Jason Eisner, Mark Dredze, Benjamin Van Durme

ABSTRACT

Acknowledgments

This work would not have been possible without my academic advisors, Jason Eisner and Mark Dredze, from whom I learned much along the way. I am especially indebted to them for giving me substantial freedom in pursuing research ideas, even in the early stages of the program. I will miss our long meetings, be it arguing over minor technical points at the whiteboard, or talking big blue sky ideas. Special thanks also to Benjamin Van Durme for serving on my committee and for his kind support.

I have been fortunate to meet many excellent people during my time at JHU, including (but certainly not limited to): Matt Gormley, Spence Green, Michael Paul, Frank Ferraro, Travis Wolfe, Markus Dreyer, Tim Vieira, and Hanna Wallach. Either through games of squash or long technical conversations, my time as a student was enriched by their company. I am also proud to have been part of the broader CLSP community, home to many brilliant researchers.

My work was generously supported by the Human Language Technology Centre of Excellence and the National Science Foundation, Partnerships for International Research and Education (PIRE). Through PIRE, spent a brief stint as a visiting

researcher in Australia, where I got a chance to meet many fine people at Macquarie University, the University of Sydney, and the University of Melbourne.

Finally, I am grateful to Maria-Veronica Banks for helping to keep me sane around submission deadlines, and with whom I spent most of time away from the computer.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Learning from unlabeled data is a long-standing challenge in machine learning. A principled solution involves modeling the full joint distribution over inputs and the latent structure of interest, and imputing the missing data via marginalization. Unfortunately, such marginalization is expensive for most non-trivial problems, which places practical limits on the expressiveness of generative models. As a result, joint models often encode strict assumptions about the underlying process such as fixed-order Markovian assumptions and employ simple count-based features of the inputs. In contrast, conditional models, which do not directly model the observed data, are free to incorporate rich overlapping features of the input in order to predict the latent structure of interest. It would be desirable to develop expressive generative models that retain tractable inference. This is the topic of this thesis. In particular, we explore joint models which relax fixed-order Markov assumptions, and investigate the use of

recurrent neural networks for automatic feature induction in the generative process.

We focus on two structured prediction problems: (1) imputing labeled segmentions of input character sequences, and (2) imputing directed spanning trees relating strings in text corpora. These problems arise in many applications of practical interest, but we are primarily concerned with named-entity recognition and cross-document coreference resolution in this work.

For named-entity recognition, we propose a generative model in which the observed characters originate from a latent non-Markov process over words, and where the characters are themselves produced via a non-Markov process: a recurrent neural network (RNN). We propose a sampler for the proposed model in which sequential Monte Carlo is used as a transition kernel for a Gibbs sampler. The kernel is amenable to a fast parallel implementation, and results in fast mixing in practice.

For cross-document coreference resolution, we move beyond sequence modeling to consider string-to-string transduction. We stipulate a generative process for a corpus of documents in which entity names arise from copying—and optionally transforming—previous names of the same entity. Our proposed model is sensitive to both the context in which the names occur as well as their spelling. The string-to-string transformations correspond to systematic linguistic processes such as abbreviation, typos, and nicknaming, and by analogy to biology, we think of them as mutations along the edges of a phylogeny. We propose a novel block Gibbs sampler for this problem that alternates between sampling an ordering of the mentions and a spanning tree relating

all mentions in the corpus.

1.1 Chapter Abstracts

§2 Proper names often contain rich sub-structure. For instance, person names may contain titles, first names, last names, initials, and so on. Unfortunately, it would impractical to design hand-crafted grammars for all combinations of entity types and languages. Therefore, we explore an alternate approach to name modeling using character-level language models. We develop several architectures using long-term short-term (LSTM) recurrent units. We find that the LSTM approach outperforms other state-of-the-art sequences models such as the sequence memoizer (SM), for the problem of classifying the entity type given a name.

§3 Lexical resources such as dictionaries and gazetteers are instrumental to the performance of many state-of-the-art NLP systems. This type-level supervision may be used as a source of strong features or hard constraints in discriminative models. However, incorporating strong features in discriminative models may lead to weight under-training: poor recall of out-of-gazetteer types due to overfitting the gazetteer. We observe that generative modeling provides a principled solution: we construct a model that generates both type-level and token-level data. Because it had to explain the gazetteer types, it is also good at predicting novel types. We experimentally evaluate our proposed approach on two tasks:

part-of-speech (POS) induction and named-entity recognition (NER). We find that (1) modeling dictionaries for POS induction leads to consistently improved accuracies across several languages, and (2) our generative NER model provides better generalization to novel types.

§4 Many linguistic and textual processes involve transduction of strings. We show how to learn a stochastic transducer from an unorganized collection of strings (rather than string pairs). The role of the transducer is to organize the collection. Our generative model explains similarities among the strings by supposing that some strings in the collection were not generated *ab initio*, but were instead derived by transduction from other, "similar" strings in the collection. The generative process assumes that each entity mention arises from copying and optionally mutating an earlier name from a similar context. Clustering the mentions into entities depends on recovering this copying tree jointly with estimating models of the mutation process and parent selection process. We present a block Gibbs sampler for posterior inference, and apply our approach to the problem of co-reference resolution on several datasets.

Chapter 2

Predicting Sequence Attributes using Recurrent Neural Networks

2.1 Chapter Overview

Entity names sometimes appear in a canonical form in text. For instance, Barack Obama is a canonical form of the name of the current president of the United States. Of course, there are other ways of referring to the same person (some of which may also be considered canonical). Wikipedia contains more than 100 variations of Barack Obama as redirect pages.[1] These include formal names with titles, aliases, names with middle initials or complete middle names, and many other variations. Furthermore,

[1] A Wikipedia redirect page has no content itself, but instead sends the reader to another page. By *redirect*, we are specifically referring to the title of the redirect page which often contains interesting name variations.

President Obama	Barack H. Obama, Jr.
Barak Obamba	Barry Soetoro

Figure 2.1: A selection of Wikipedia redirects for Barack Obama.

misspellings (Barrack), typos (Barck), and lowercase/uppercase variations (obama) further contribute to name variation, especially in informal genres such as blogs. Some examples are shown in Figure 2.1. In this chapter, we consider the problem of modeling the name-*ness* of different strings (sequences of characters). For person names which tend to have a predictable structure, it is tempting to define rules ahead of time, perhaps in some probabilistic framework like a probabilistic context-free grammar (PCFG). However, this approach would require hand-crafting grammars for all entity types of interest and for all languages of interest.

In this section, we pursue a more general strategy which is to model names using expressive sequence models with a capacity to learn name structure without prescribing assumptions in the model structure (for instance, context-free and context-sensitive languages [Gers and Schmidhuber, 2001]). However, we begin with a discussion of sequence models more generally.

In later chapters, we will incorporate name models as components of more involved generative processes. In this chapter, our modest goal is to design models that assign high probability to strings resembling names of the appropriate type given conditioning information, and lower probability to other strings. For instance, for two types human

(HUM) and car model (MOD):

$$\ln p(\text{Mr. Obama} \mid e.\text{type} = \text{HUM}) = p_1 \quad \ln p(\text{Mr. Obama} \mid e.\text{type} = \text{MOD}) \quad = p_3$$

$$\ln p(\text{Ford Focus} \mid e.\text{type} = \text{HUM}) = p_2 \quad \ln p(\text{Ford Focus} \mid e.\text{type} = \text{MOD}) \quad = p_4$$

we want $p_1 > p_2$ and $p_4 > p_3$.

The primary contributions in this chapter are the following:

§2.2 We introduce new datasets for training name models for many entity types, derived from Wikidata and redirect data in DBpedia [Vrandečić and Krötzsch, 2014, Auer et al., 2007].

§2.7 We propose a joint parametrization of the class-based RNN which outperforms the independently trained baseline in both low and high data-settings, while using $1/C$ of the parameters for C distinct classes.

§2.7 We show that multi-conditional training of the joint RNN model leads to consistent gains over a purely generative training criterion.

2.2 Data

The experiments in this section use data from Wikidata [Vrandečić and Krötzsch, 2014]. Wikidata is a free collaborative knowledge base with structured data from Wikipedia [Wikipedia, 2015] and other sources. Each entity in Wikidata is associated

type	**freq.**	**type**	**freq.**
person	1285998	road	19498
film	97616	city	19296
settlement	97179	ship	18184
company	55364	church	16467
band	35448	high school	16382
railway station	34485	organization	13427
mountain	20361	airport	12713

Figure 2.2: A sample of entity types in Wikidata (frequencies as of August 2015).

with different *statements*, each of which has metadata such as references to source material. For instance, entity Q937 (Albert Einstein) has two statements of the property P1412 (languages spoken or published): Q1860 (English) and Q188 (German). Each entity in Wikidata has a title (such as Albert Einstein above) and optionally a list of known aliases (*also known as*). These are of particular interest, as they will be used in the experiments in this section as a source of training data for name models. Crucially, entity titles and aliases are available in all languages for which there is a corresponding Wikipedia. Some frequent entity types are shown in Figure 2.2.

Redirects. In the previous section, we provided examples of Wikipedia redirects as examples of person name variation. This data was obtained from the DBpedia project, which as of August 2015 provides dumps containing more than 6.5 million redirect pages [Auer et al., 2007]. The titles of redirect pages provide a useful source of entity names as they contain much variation. A sample of redirects for English person names are shown in Figure 2.3.

Missing / incomplete data. A challenge when dealing with collaborative resources

Ho Chi Minh, Ho chi mihn, Ho-Chi Minh, Ho Chih-minh
Guy Fawkes, Guy fawkes, Guy faux, Guy Falks, Guy Faukes, Guy Fawks, Guy foxe
Nicholas II of Russia, Nikolai Aleksandrovich Romanov, Nicholas Alexandrovich of Russia
Bill Gates, Lord Billy, Bill Gates, BillGates, Billy Gates, William Gates III, William H. Gates
William Shakespeare, William shekspere, William shakspeare, Bill Shakespear
Bill Clinton, Billll Clinton, William Jefferson Blythe IV, Bill J. Clinton, William J Clinton

Figure 2.3: Sample alias lists scraped from Wikipedia.

such as Wikidata is that entities may have unknown properties. For instance, the absence of a *date of death* (p570) does not necessarily imply that an entity is still living. In Wikipedia, it is typically the case that a small number of popular entities have many known properties, while a long-tail of other entities have a few basic properties. A proper treatment of this problem is beyond the scope of this thesis; see [Mohan et al., 2013] for a recent discussion. For the experiments in this section, we avoid the issue of missing data by selecting entities that have no missing data, or by focusing on a subset of properties.

2.3 Types and Tokens

Example 2.1. On Tuesday, September 8th, 2015, a town in northern Wales was one of the U.K.'s warmest locations. It enjoyed a considerable surge in popularity online and in the media:[2]

> British meteorologist Liam Dutton is the toast of the web today for somehow managing to pronounce Llanfairpwllgwyngyllgogerychwyrndrobwllllantysiliogogogoch, a village in north west Wales, without missing a beat.

[2] `https://www.youtube.com/watch?v=fHxO0UdpoxM`

Example 2.1 describes a case where where an unlikely name **type**, "Llanfairpwllgwyngyllgogerychwyrndrobwllllantysiliogogogoch," occurs with high **token** frequency. When learning the shape of town names, it would be desirable not to skew a generative distribution towards the spelling of names which occur with high frequency. Another example is the name "Barack Obama," which is not representative of common English person name spellings. In the next section, we describe a Bayesian nonparametric framework that will be useful accounting for rare spellings such as the place above when estimating generative distributions over character sequences. It will be used as a building block for the Bayesian sequence model used in this chapter (§2.4.1), as well as the more complex models described in Chapter 3.

Example 2.2. Word segmentation. In word segmentation, a system is presented with a sequence of symbols, typically phonemically transcribed utterances, and must output a segmentation of the input sequence into subsequences (words). One early approach to this problem is to use a generative n-gram model over words, with a certain probability of emitting a distinguished boundary symbol between each word [Venkataraman, 2001]. Inference in this model consists of searching over all possible positions for the boundary symbols in the input symbol sequence. Unfortunately, this model suffers from a pathological defect in that the solution which maximizes likelihood is the unsegmented input [Goldwater et al., 2006]. The problem is that the n-gram model does not have a bias favoring the types of power-law word distributions exhibited in natural language [Powers, 1998].

These examples motivate a useful technical concept. The term *memoization* refers to an optimization in computer science where the result of a function call is cached, so that subsequent calls may look-up the cached value rather than repeating the calculation. The probabilistic analog of this idea *optionally* caches draws a distribution, known as the the **base distribution**. This stochastic memoization (c.f. Goodman et al. [2012]) induces a different distribution called the **adapted distribution**. A sample from the adapted distribution is produced either via a draw from the base distribution or by re-using a previously generated value. The adapted distribution has the same support as the base distribution (e.g., a distribution over words), but the memoization has the effect of concentrating probability mass on previous observations (e.g., existing Welsh place names) while reserving some mass for novel events. The trade-off between novelty and re-use depends on the choice of memoizer model (henceforth **adaptor**). Probabilistic adaptors provide a principled way of holding out probability mass for novel events, and to encourage a "rich-get-richer" effect in which some types occur with high frequency—just as some words like determiners occur frequently in natural language.

2.3.1 Pitman-Yor Processes

Pólya urn schemes provide a convenient description of a rich-get-richer process (named after the mathematician George Pólya). They are described in terms of an urn filled with balls of different colors. In a simple urn model, each draw from the

urn results in a ball of the same color being added back to the urn, *in addition* to returning the picked ball back to the urn. Thus, the likelihood of picking a ball of that color increases with successive draws. Compare this to sampling without replacement, in which a draw permanently removes a ball of the corresponding color from the urn, reducing its probability in future draws. The balls represent objects of interest: in our case, these are either characters or words, and observing a word or character increases its probability.

The Chinese restaurant process (CRP) is a particular type of Pólya urn scheme[3], which is more intuitively understood via the following analogy (from which it derives its name). The Chinese restaurant is initially empty, and when it opens customers arrive one-at-a-time and take seats around the tables (of which there are infinitely many, initially unoccupied). At any given time, the customers are in some arrangement around the tables—a partition—and each table serves a single *dish*. The dish is drawn from the *base distribution* of the CRP, which in our case will usually correspond to a distribution over words or characters, though in general may be continuous in which case draws are distinct with probability 1.

We now build on these intuitive descriptions to formalize a useful generalization of the CRP called the Pitman-Yor process (PYP) [Perman et al., 1992, Ishwaran and

[3]The CRP may be understood as the following modified urn scheme: the urn initially contains α black balls. When drawing a ball from the urn, if it is black, return the ball to the urn along with an additional non-black ball drawn from a uniform distribution over an infinite set of colors, and suppose the drawn color is the value of the actual draw [Hoppe, 1984]. This description corresponds to the case where the base distribution is over a continuous space, in which case draws are distinct with probability 1.

James, 2001].

Let PYP(α, d, P_0) denote a PYP with concentration (or strength) parameter $\alpha > -d$, discount parameter $d \in [0, 1)$, and a base distribution P_0 over Σ (e.g. colors, words, etc.). The CRP defines a distribution over partitions which may be fancifully described as an arrangement of customers around tables, where each table is associated with a dish $s \in \Sigma$ that is a draw from the base distribution of the PYP. The process by which customers are seated corresponds to the following sequential allocation scheme, in which the nth customer either

1. sits at an existing table k with probability proportional to $\frac{n_k - d}{n+\alpha}$, which corresponds to re-using a cached value equal to the dish $s \in \Sigma$ served at table k;

2. sits at a new table with probability proportional to $\frac{Kd+\alpha}{n+\alpha}$, which means drawing a new dish $s \in \Sigma$ from the base distribution P_0

where n_k is the number of customers sitting at table k, and K is the total number of tables.

Together, the discount parameter $d \in [0, 1]$ and concentration parameter $\alpha > 0$ control the manner of the clustering (partition). A high value of α will result in a large number of tables regardless of the number of customers sitting at those tables. As $d \to 1$ the arrangement is biased towards fewer customers sitting at each table.

In §2.4.1, we show how this distribution may be used as a building block in a

Bayesian language model, where restaurants are associated with particular contexts. In §3, we apply PYP priors as a component in a generative latent-variable model.

2.4 Sequence Models

We are interested in models of the form

$$p(x_1, \ldots, x_t) = \prod_{t=1}^{T} p(x_t \mid x_1, \ldots, x_{t-1}) \tag{2.1}$$

where the observed symbols x_t are drawn from some alphabet $\mathcal{A}$. In the speech recognition and machine translation communities, such models are known as **language models** (LM), and the observed symbols correspond to words drawn from a finite vocabulary. In this chapter our focus will be on character-level models.

Note that in §2.1, the probability of a symbol at time t is conditioned on all previous symbols $x_1, \ldots, x_{t-1}$, which poses increasing statistical and computational challenges as T (the length of the sequence) increases. Historically, these challenges have been avoided by approximating equation (2.1) via a fixed-order Markov assumption:

$$p(x_t \mid x_1, \ldots, x_{T-1}) \approx p(x_t \mid x_{t-N}, \ldots, x_{t-1}) \tag{2.2}$$

where the Markov order n is a hyper-parameter of the model. If the alphabet $\mathcal{A}$ is fixed, the model may then be parametrized using a categorical distribution in which

the probability of symbol is simply derived from observed counts of that symbol in different contexts $\boldsymbol{c}$:

$$\theta_{a\boldsymbol{c}} := \frac{n_{a\boldsymbol{c}}}{n_{\cdot\boldsymbol{c}}} \tag{2.3}$$

where $n_{a\boldsymbol{c}}$ and $n_{\cdot\boldsymbol{c}}$ are the number of occurences of a followed by context $\boldsymbol{c}$ and the number of occurences of context $\boldsymbol{c}$ preceeded by any symbol in $\mathcal{A}$, respectively. This distribution is "closed-form" in the sense that

$$\sum_{a' \in \mathcal{A}} \theta_{a'\boldsymbol{c}} = 1 \tag{2.4}$$

$$p_\theta(a \mid \boldsymbol{c}) = \theta_{a\boldsymbol{c}} \tag{2.5}$$

There are three main difficulties with this n-gram approach:

- As n increases, the number of unique contexts increases exponentially, leading to sparse observations and poor estimates of the predictive distribution in those contexts.
- The number of parameters also grows exponentially in n, which becomes intractable for large n.
- Out-of-vocabulary symbols may appear in held-out data that were not present during parameter estimation. This is particularly problematic when the symbols are words, since (for instance) typos are an endless source of out-of-vocabulary

words.

Intuitively, larger n correspond to more specific contexts, which will be sparser in natural language. In practice, the n-gram model above must be combined with so-called "smoothing" schemes to deal with the first two difficulties. Due to the importance of LMs in NLP, many such schemes have been proposed in the literature. In the next chapter, we describe the sequence memoizer (SM), a Bayesian nonparametric language model which addresses all of the above difficulties. This model will serve as a strong baseline to the RNN-based models described in later sections.

2.4.1 Bayesian Non-Parametric Language Models

Pitman-Yor language models use Bayesian priors for purposes of smoothing language models [Teh, 2006]. We focus on the sequence memoizer (SM), which does not make any fixed-order Markov assumptions [Wood et al., 2009]. In this model, the probability $p(s \mid \mathbf{c})$ of a symbol s in a given context $\mathbf{c}$ is given by an adapted distribution with base distribution $p(s \mid \sigma(\mathbf{c}))$, where $\sigma(c_1 c_2 c_3 \ldots) = c_2 c_3 \ldots$, and the base distribution is itself adapted. This *back-off* process repeats until the context is empty.

The resulting conditional dependencies have a simple linear structure:

$$p(s \mid \epsilon) \sim \text{PYP}(d_0, \alpha_0, H) \tag{2.6}$$

$$\dots$$

$$p(s \mid \sigma(\mathbf{c})) \sim \text{PYP}(d_{|\mathbf{c}|-1}, \alpha_{|\mathbf{c}|-1}, p(s \mid \sigma(\sigma(\mathbf{c}))) \tag{2.7}$$

$$p(s \mid \mathbf{c}) \sim \text{PYP}(d_{|\mathbf{c}|}, \alpha_{|\mathbf{c}|}, p(s \mid \sigma(\mathbf{c})) \tag{2.8}$$

where ϵ is the empty sequence. Intuitively, the model is able to obtain more robust estimates in specific contexts (large $|\mathbf{c}|$) by backing-off to less-specific contexts (small $|\mathbf{c}|$) for which there are more observations. The SM trade-off novelty and re-use at each level of the hierarchy according to the corresponding discount and concentration hyperparameters, which in practice may either be fixed, optimized or sampled (see [Wood et al., 2009, Gasthaus and Teh, 2010]). In our experiments, we follow prior work and fix discount parameters to the values recommended in Wood et al. [2009], which also fix $\alpha_{\boldsymbol{c}} = 0$.

It is instructive to consider the predictive distribution, which takes a simple form:

$$p_{\boldsymbol{c}}(s) = \frac{n_{\boldsymbol{c}s} - t_{\boldsymbol{c}s}d}{\alpha_{\boldsymbol{c}} + n_{\boldsymbol{c}\cdot}} + \frac{\alpha_{\boldsymbol{c}} + t_{\boldsymbol{c}\cdot}d}{\alpha_{\boldsymbol{c}} + n_{\boldsymbol{c}\cdot}} p_{\sigma(\boldsymbol{c})}(s) \tag{2.9}$$

where $n_{\boldsymbol{c}s}$ is the count of symbol s occurring in context $\boldsymbol{c}$, and $n_{\boldsymbol{c}\cdot}$ is the total count of *all* symbols occurring in context $\boldsymbol{c}$. Looking at the numerators, the discounting effect

of d is clear: we subtract $t_{\boldsymbol{cs}}d$ from the observed count $n_{\boldsymbol{cs}}$ and shift that probability mass to second term, where $t_{\boldsymbol{c}\cdot}d$ is added—raising the probability of sitting at a new tables. We also see that even when $\alpha_{\boldsymbol{c}} = 0$, there is nonzero probability of sitting at a new table due to the discounting effect.

The choice of top-level base distribution H depends on the space being modeled. When modeling sequences of characters, we take H to be a uniform distribution over a finite set of character symbols. In §3, we are interested in distributions over the infinite set of possible words, and therefore take H to be a distribution over character sequences of unbounded length. In general, the base distribution may either be fixed or re-estimated from data.

2.4.2 Recurrent Neural Networks

RNN language models have a number of limitations compared to nonparametric models such as the SM. First, they usually have millions of parameters and have highly non-convex likelihood functions. As a result, they are computationally expensive to estimate and prone to local optima. Second, while the PYP provides a mechanism to hold-out probability mass for novel events, this is not possible in typical RNN parametrizations: they must use an alphabet (or vocabulary) of fixed size defined before seeing any data.

What RNNs and the SM have in common is the ability to capture long-range dependencies. The RNN does this in a more powerful way than the SM by summarizing

the context in the fixed-dimensional real tensor—the state of the RNN. [Bengio et al., 2006] Until recently, RNNLMs have performed more or less on par with n-gram models that use advanced smoothing techniques; however, recent developments in parallel hardware, second-order parameter estimation methods, and RNN architectures have together rendered neural methods the *de facto* choice for many sequence learning problems.

Example 2.3. Matching parenthesis and modeling character classes. While the SM is able to capture long-term dependencies by conditioning on arbitrary amounts of context, it suffers from limited expressivity. As one example of this, consider the problem of modeling parenthetical dates. Given a novel string (i.e. unseen during parameter estimation):

```
( 1 9 8 4
```

the SM will not assign high probability to a matched parenthesis `)` if the prefix `(1984` (or coincidentally some suffix thereof) occurs in the training data. This example illustrates another weakness of the n-gram approach, which is that the symbols $1, 9, 8, 4$ are all considered entirely distinct. The SM is unable to recognize that they are all numeric symbols, nor that after an opening parenthesis it is common for four such symbols to appear.

A recurrent neural network (RNN) is a generalization of the standard feedforward neural network to sequences. Given a sequence of inputs $(\mathbf{x}_1, \ldots, \mathbf{x}_T)$, the RNN

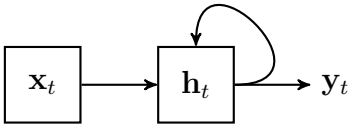

Figure 2.4: Recurrent neural network.

produces outputs via the following recurrence

$$\mathbf{h}_t = \text{sigm}(\mathbf{W}_1\mathbf{x}_t + \mathbf{W}_2\mathbf{h}_{t-1} + \mathbf{b}_h) \tag{2.10}$$

$$\mathbf{y}_t = \mathbf{W}_3\mathbf{h}_t + \mathbf{b}_y \tag{2.11}$$

The nonlinearity sigm is computed element-wise.[4] The matrices $\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3$ and the vector $\mathbf{h}_0$ are trainable parameters of the model, along with the biases $\mathbf{b}_h$ and $\mathbf{b}_y$. Note that for a given input x_t and RNN state $\mathbf{h}_{t-1}$, the output $\mathbf{y}_t$ and new state $\mathbf{h}_t$ are produced deterministically. To apply this idea to discrete symbol sequences, we require an embedding E which maps inputs x_t to corresponding vector-valued embeddings $\mathbf{x}_t$

$$E : \mathbb{Z}_+ \to \mathbb{R}^d \tag{2.12}$$

The mapping E is a deterministic slice operation (selecting a vector slice of a matrix), but the resulting embeddings $\mathbf{x}_t$ are trainable parameters of the model [Bengio et al., 2003]. We suppose without loss of generatility that the input embedding and recurrent

[4]Other nonlinearities, such as tanh, are sometimes used. The choice of nonlinearity may be thought of as a hyperparameter of the model.

layer have the same dimensionality d: $\boldsymbol{x}_t, \boldsymbol{h}_t \in \mathbb{R}^d$. (In §2.6, we relax this assumption.)

Probabilistic interpretation. The RNN defines a generative probability distribution via a softmax transformation of the sequence of output vectors $\mathbf{y}_1, \ldots, \mathbf{y}_T$:

$$p(x_t = j \mid x_1, \ldots, x_{t-1}) = \text{softmax}(\mathbf{y}_t) \tag{2.13}$$

$$= \frac{\exp(y_t^{(j)})}{\sum_k \exp(y_t^{(k)})} \tag{2.14}$$

This final layer is analogous to that of a log-linear model, with the main difference being that log-linear models are typically parametrized via fixed feature templates [Berg-Kirkpatrick et al., 2010].

Backpropagation-through-time. Neural networks are typically trained via error gradient backpropagation. Unfortunately, this approach is not immediately applicable to RNNs, as the recurrent connections form cycles in the resulting computation graph. Several strategies have been proposed in the literature to deal with this problem. We use backpropagation-through-time (BPTT) here, which involves "cloning" the network for ρ time steps and modifying the recurrent connections to flow through time from one clone to the next. There are no cycles in the modified computation graph, which is therefore amenable to standard error backpropagation techniques. Note that use of a fixed ρ limits the ability of the model to capture long distance dependencies. In principle, one could set ρ to be the length of sequence; however, in practice this may lead to numerical instability during optimization [Pascanu et al., 2013]. This difficulty

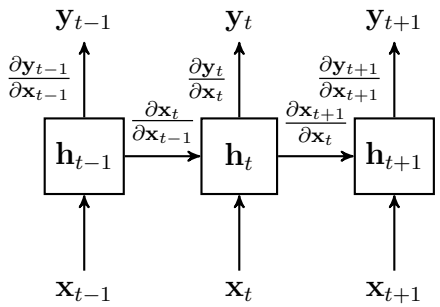

Figure 2.5: Unrolled recurrent neural network for backpropagation-through-time, shown here for the case of a single recurrent layer $\mathbf{h}_t$. Gradients flow in the opposite direction as the computation graph.

takes the form of *vanishing* and *exploding* gradients [Bengio et al., 1994].

There is a fairly straightforward solution to the *exploding gradient* problem, which involves scaling down the gradient when its norm exceeds a given threshold [Pascanu et al., 2013]; see Algorithm 1.

Algorithm 1 Clipping exploding gradients

```
function CLIP(g)
    if ‖g‖ ≥ threshold then
        return (threshold / ‖g‖) g
    end if
    return g
end function
```

The *vanishing gradient* problem is more severe. One attempt to address this problem is via optimization procedures which incorporate approximate second-order terms; we employ one such procedure in our experiments (described below). Another largely orthogonal approach is to modify the RNN architecture.

Several modifications to the vanilla RNN architecture have been proposed [Hochre-

iter and Schmidhuber, 1997a, Cho et al., 2014]. We adopt the recently popularized long short-term memory framework [Hochreiter and Schmidhuber, 1997b, Hochreiter et al., 2001]. While the LSTM-RNN has more parameters than a vanilla RNN, it enjoys the same $O(1)$ computational complexity per time step and weight.

Let x_t, h_t, and m_t denote the input, recurrent state, and memory state at time t.

$$\mathbf{i}_t = \text{sigm}(\mathbf{W}_1\mathbf{x}_t + \mathbf{W}_2\mathbf{h}_{t-1}) \tag{2.15}$$

$$\mathbf{i}'_t = \tanh(\mathbf{W}_3\mathbf{x}_t + \mathbf{W}_4\mathbf{h}_{t-1}) \tag{2.16}$$

$$\mathbf{f}_t = \text{sigm}(\mathbf{W}_5\mathbf{x}_t + \mathbf{W}_6\mathbf{h}_{t-1}) \tag{2.17}$$

$$\mathbf{o}_t = \text{sigm}(\mathbf{W}_7\mathbf{x}_t + \mathbf{W}_8\mathbf{h}_{t-1}) \tag{2.18}$$

$$\mathbf{m}_t = \mathbf{m}_{t-1} \odot \mathbf{f}_t + \mathbf{i}_t \odot \mathbf{i}'_t \tag{2.19}$$

$$\mathbf{h}_t = \mathbf{m}_t \odot \mathbf{o}_t \tag{2.20}$$

where $\odot$ denotes element-wise multiplication. The matrices $\mathbf{W}_1, \ldots, \mathbf{W}_8$ and the vector $\mathbf{h}_0$ are trainable parameters of the model, and the nonlinearities tanh and sigm are computed element-wise. Together, the gates ($\mathbf{i}$, $\mathbf{i}$', $\mathbf{f}$, $\mathbf{o}$) and memory cells ($\mathbf{m}$) allow the LSTM unit to adaptively control the flow of detected features across multiple timesteps.

Multiple layers. It is straightforward to extend the RNN-LSTM to incorporate multiple layers. This is accomplished by stacking multiple LSTM recurrent layers, feeding the output of one layer to the next. This is illustrated in Figure 2.6.

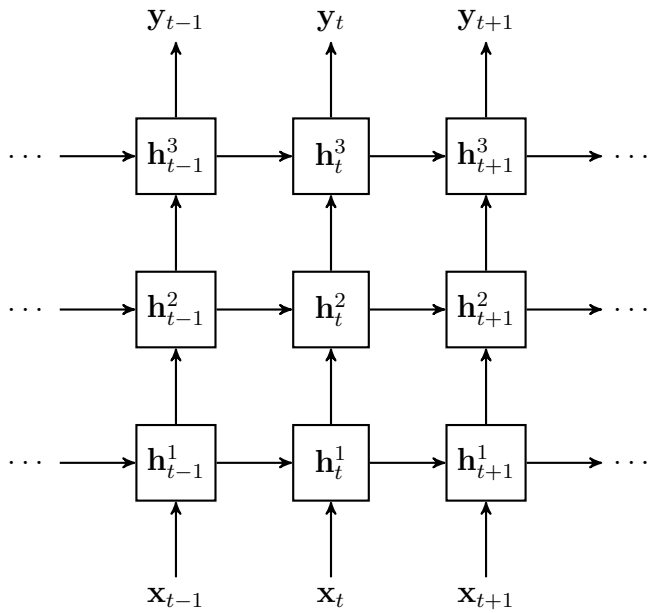

Figure 2.6: Stacked RNN architecture.

Dropout regularization. An issue with overparametrized models such as RNNs is that with limited training data, they can and often will overfit the training data [Hinton et al., 2012, Srivastava et al., 2014]. We employ dropout to both input and recurrent connections [Zaremba et al., 2014].

Stochastic gradient optimization methods. First-order stochastic gradient methods are sensitive to the learning rate schedule, which is difficult to tune in practice. For all experiments in this thesis, we instead use the Adam algorithm [Kingma and Ba, 2014]. Adam incorporates adaptive estimates of lower-order gradients, which enables it to adapt its learning rate per parameter.

Efficient training on graphics processing units. Due to the slow, incremental progress of batch gradient descent methods, many passes through the data (*epochs*)

are necessary to achieve good performance. As described in §2.2, we are working with large corpora with millions of training instances. To make optimization tractable we parallelize gradient computations on graphics processing units (GPUs). At a high-level, this involves performing both the forward and backwards passes on multiple sequences simultaneously so that at time t, the input to the RNN is a vector of inputs where input k is the t-th input symbol of instance k in the batch. As a result, it is helpful to organize batches so that they contain only sequences of the same length, so that no computation is wasted. When this is not possible (e.g., for left-over sequences), the outputs must be masked to ensure proper gradient computations.

2.5 Learning from Gazetteers

In this section we train SM and RNN name models from lists of names of different entity types. We evaluate the ability of the trained models to classify the type of novel names. As previously described in §2.2, we extract gazetteers from Wikidata.

The experimental procedure is straightforward. We select 12 entity types from among those with at least 10000 instances with English titles. We then impute the

type of novel names via Bayes' theorem:

$$\begin{aligned} q^* &= \operatorname*{argmax}_q p(\text{p31} = q \mid \text{name}) \\ &= \operatorname*{argmax}_q \frac{p(\text{q31} = q)p(\text{name} \mid \text{p31} = q)}{p(\text{name})} \\ &= \operatorname*{argmax}_q p(\text{q31} = q)p(\text{name} \mid \text{p31} = q) \end{aligned} \tag{2.21}$$

Hyperparameter settings. To pick the RNN architecture, we perform a grid search over the following dimensions (best setting in parenthesis): batch size (128), RNN state size (256), number of layers (3), gradient norm threshold (10), and initial Adam learning rate (0.001). To select the SM hyperparameters (discount and concentration parameters), we follow the recommendations in [Gasthaus and Teh, 2010].

Results. We show some predictions from the trained models in Figure 2.7. The aggregate accuracy is shown in Figure 2.8. Note that the optimal size of the RNN (128, 256, and 512) depends on the amount of training data. For low data settings, it is preferable to use smaller networks. Also, the RNN models consistently outperform the SM baseline as the amount of training data exceeds 1000. A breakdown of performance for each entity type is shown in Figure 2.9.

Entity type	**Viterbi prediction**
airport	S a n t a _ C r e s t a _ A i r p o r t
building	C a s a _ d e _ l a _ S a n t a _ d e _ l a _ B a n k
city	S a n _ J o s ? _ d e l _ R o s a
high school*	S t . _ M a r y ' s _ H i g h _ S c h o o l
person	J o h n _ M a r k _ B a r t o n
road*	M i n n e s o t a _ S t a t e _ H i g h w a y _ 1 2
band	T h e _ S t e e l _ S t r i n g _ Q u a r t e t
church	S t _ M a r y ' s _ C h u r c h , _ B r i g h t o n
film	T h e _ B o y _ M a n
mountain	M o u n t _ B a r r e y
train station	S h i m o - S h i m o t o _ S t a t i o n
ship	U S S _ L S T - 5 5 1

Figure 2.7: Predictions from trained RNN name model on 8000 sequences of the corresponding type. Each predicted sequence is terminated with a distinguished end-of-sequence symbol, omitted above. Predictions which correspond exactly to an entry in the knowledge base are marked with an asterisk (*) next to the entity type.

2.6 Entity Name Models

Wikidata provides a wealth of structured knowledge about entities. In this section we experiment with generative name models that are sensitive to attributes such as gender, nationality, and known given and family names. We condition the RNN on additional entity attributes:

$$\ln p(\mathsf{Mr.\ Obama} \mid \mathsf{male}, \mathsf{kenya}, \mathsf{usa}, \mathsf{obama}, \ldots)$$

A flexible way of incorporating entity attributes in the RNN is by framing the process as a sequence-to-sequence model [Sutskever et al., 2014, Cho et al., 2014]. We will defer a detailed description of these models to Chapter 4. The high-level idea is to read in an encoding of all entity attributes, which are then summarized in the fixed-

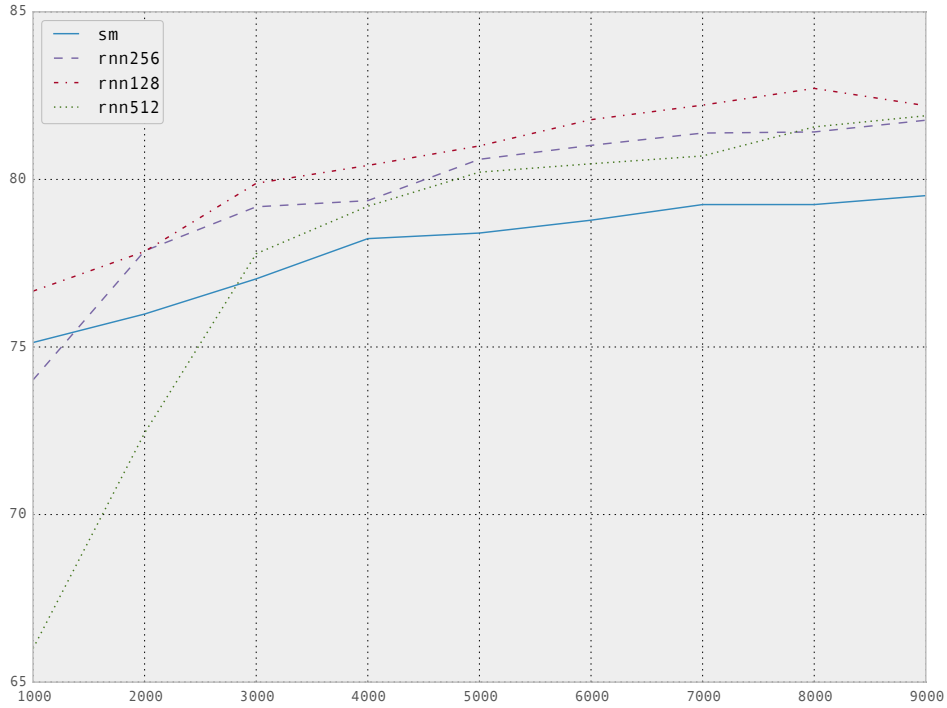

Figure 2.8: The aggregate accuracy for the SM and for RNNs of three different recurrent state sizes. The x-axis is the amount of training data used to train each name model (one for each of 12 types). The y-axis shows aggregate prediction accuracy.

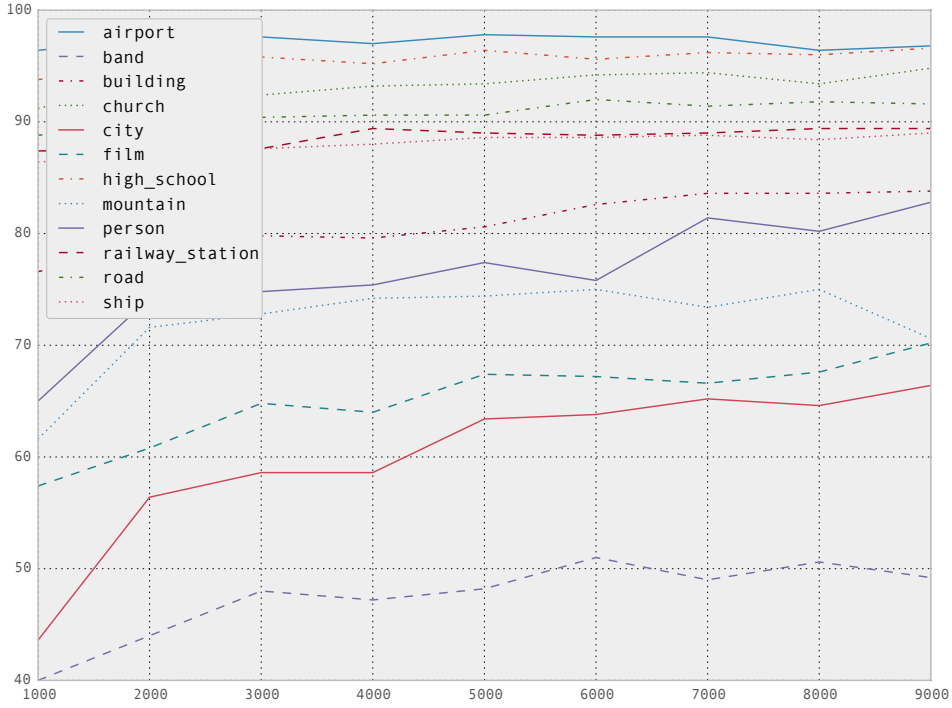

Figure 2.9: A breakdown of classification accuracy for each entity type. The x- and y-axis are as in Figure 2.8.

Varible # of entity attributes		**Viterbi prediction (true entity name)**
p31_q5 p27_q403 p172_q127885 ...	→	Andrej Pavali (Mihailo Obrenovi III)
p31_q5 p27_q30 p21_q6581097 ...	→	John Martin* (Daniel Hall)
p31_q5 p27_q142 p21_q6581072 ...	→	Marie Bourne (Sylvie Denis)
p31_q5 p27_q1045 p21_q6581097 ...	→	Abdul Al-Ali (Ismail Jim'ale Osoble)
p31_q5 p27_q38 p21_q6581097 ...	→	Antonio Martini* (Giacomo dalla Torre ...)

Figure 2.10: A selection of predictions from the sequence-to-sequence entity name model. The entity attributes refer to Wikidata entities in the format property_value. Predictions which correspond exactly to an entry in the knowledge base are marked with an asterisk (*).

dimensional RNN state (possibly consisting of multiple stacked layers). Given the input encoding, the RNN then outputs the name character by character until the distinguished end-of-sequence symbol is emitted. This approach achieves qualitatively promising results, as shown in Figure 2.10.

However, this framing of the problem suffers from a number of drawbacks. First, there is no natural ordering of entity attributes, so the order in which attributes are encoded is arbitrary. Furthermore, different numbers of attributes may adversely affect the predictive ability of the RNN, as performance tends to degrade with the length of the input sequence [Sutskever et al., 2014].

A different approach is to modify the RNN with additional inputs, one for each entity attribute. Suppose there are M attributes. We modify the LSTM units with additional embeddings $\mathbf{a}_m$ and associated weights $\mathbf{A}_m$. For instance, we modify $\mathbf{i}_t$ as follows:

$$\mathbf{i}_t = \text{sigm}(\mathbf{W}_1\mathbf{x}_t + \underbrace{\mathbf{A}_1\mathbf{a}_1 + \ldots + \mathbf{A}_m\mathbf{a}_m}_{\text{attribute weights and embeddings}} + \mathbf{W}_2\mathbf{h}_{t-1}) \tag{2.22}$$

We will use this modification in the next section.

2.7 Improved RNN Parametrizations

In this section we modify the name model in two ways: (1) we change the paremetrization so that the RNN parameters are shared between classes at every layer, including the input embeddings (2) we introduce a different training criterion for a multi-class RNNs that optimizes both the generative log-likelihood conditioned on the class label as well the classification log-likelihood conditioned on the inputs.

2.7.1 Joint conditional RNNs

We revisit the RNN architecture described in §2.5. The previous architecture was used to parametrize a model $p_{\boldsymbol{\theta}}(\boldsymbol{x} \mid c)$ where each class c is associated with a distinct set of RNN parameters θ_c. The advantage of this approach is that it allows each class-specific model to be trained independently of the others, which is amenable to parallelized training. However, there are also downsides:

Lack of parameter sharing. There are common morphological patterns accross classes. Put more broadly, certain character sequences are likely across different classes. For instance, capital letters tend to begin words and certain consonent and vowel patterns are more common. However, if each class is modeled independently, the amount of training data to discover these broad language patterns is effectively reduced

by the number of classes.

Fixed capacity. Some classes may require RNNs with more capacity, while some classes may require less. Therefore, an architecture which works well for one class may not be optimal for another. Put another way, a fixed amount of model capacity is allocated to each class, rather than adaptively allocating capacity.

Hyper-parameter tuning. If class-specific RNNs are treated as separate optimization problems, they may require separate tuning for hyper-parameters such as optimizer learning rates and schedules. If model architectures are allowed to vary between classes in addition to optimizer hyper-parameters, this compounds the amount of tuning necessary to train each class-specific model.

Our solution is to use a single RNN with additional inputs at the input as well as the recurrent layers for the class. That is, we use a single set of RNN parameters $\boldsymbol{\theta}$ that is shared across all classes. This means that the parameters of the input symbol embeddings as well as the recurrent layers are trained jointly.

The modification to the architecture is straightforward, and is shown in Figure 2.11. At each layer, including the input layer, the class label serves as an additional input. This is similar to the previous section, where different entity attributes were inputs to the RNN. Note that unlike the input symbols x_t, we do not learn an embedding of the labels. We rather apply a fixed one-hot transformation of the label, where an input scalar is converted to a vector of length C (for C classes) where dimension c is 1 and all other entries are 0.

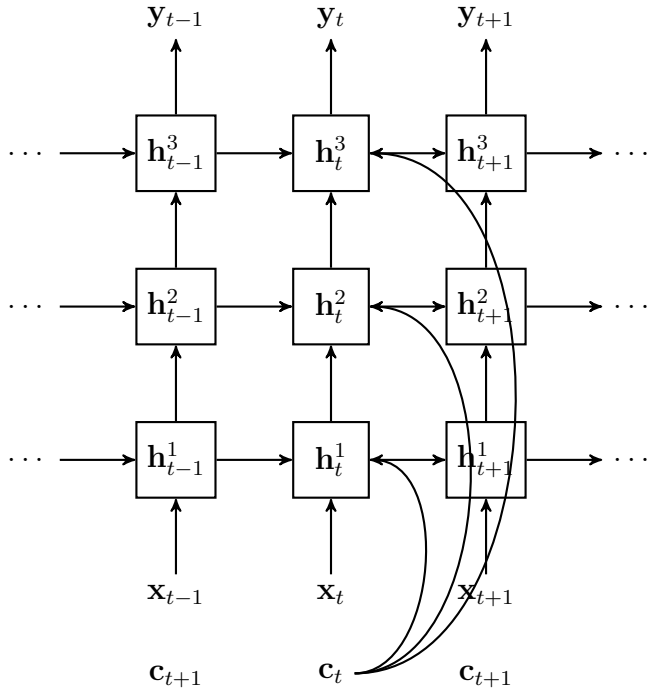

Figure 2.11: Stacked RNN architecture with one-hot encoded class inputs $\mathbf{c}_t$. Note that only the connections for $\mathbf{c}_t$ are shown for clarity.

2.7.2 Multi-Conditional Learning

In some applications such as NER, we are ultimately interested in predicting named-entity labels *conditioned* on all observed information. Most existing models for NER explicitly optimize this conditional objective using discriminative training regimes. However, in this thesis we have thus far only discussed *generative* objectives, where the training criterion is the likelihood of the observed data given the class labels. If we let $\mathbf{x}$ denote the observed inputs (e.g. a sequence of characters) and c denote a class label, we can write these two different objectives as:

$$\mathcal{L}_{\mathbf{x}|c} := p_\theta(\mathbf{x} \mid c) \text{ [generative criterion]} \tag{2.23}$$

$$\mathcal{L}_{c|\mathbf{x}} := p_\theta(c \mid \mathbf{x}) \text{ [discriminative criterion]} \tag{2.24}$$

Note that the parameters θ are common to both objectives. Multi-conditional learning is an optimization criterion which trades-off the generative and discriminative criterions [McCallum et al., 2006]. There are several interpretations of this objective. One view of the objective is that it serves as a regularizer for a purely discriminative criterion, alleviating overfitting when there is little data. Another view of the objective is to encourage parameters which not only explain the observed data, but also explain different classes "with a margin" by analogy to margin classifiers [Suykens

and Vandewalle, 1999].

$$\mathcal{L}_{\text{MC}} := p_\theta(\mathbf{x} \mid c)^\alpha p_\theta(c \mid \mathbf{x})^\beta \tag{2.25}$$

$$= \alpha \mathcal{L}_{\mathbf{x}|c} + \beta \mathcal{L}_{c|\mathbf{x}} \tag{2.26}$$

By changing the ratio α/β, the criterion gives more weight to either the generative or discriminative objectives.

2.7.3 Experiments

We evaluate the effect of the joint architecture and the MCL criterion on aggregate classification accuracy, using the same dataset as in §2.5 which has 12 classes. For each model, we fix the architecture to a three-layer RNN with input embeddings and recurrent cells of dimensionality 256. Each point in Figure 2.12 represents the best result from a grid search over learning rates and random parameter initializations. We also look at the predictions of models trained with the generative and MCL objectives in Figure 2.13. Note that the MCL criterion appears to discourage overlap between the classes, compared to the generative criterion.

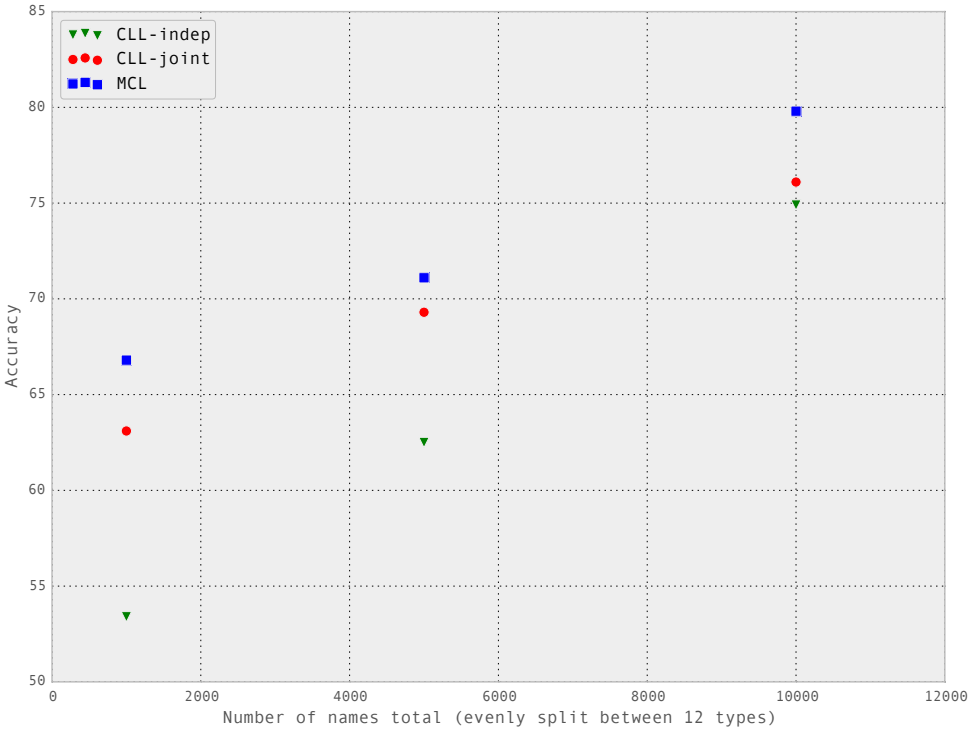

Figure 2.12: Aggregate classification accuracy for different amounts of supervised data, with three different models: conditional log-likelihood (CLL) with joint (CLL-joint) and independent (CLL-joint) parametrizations. Note that the x-axis represents the total number of training sequences, which are evenly divided between classes.

Entity type	**LL**	**MCL**
film	The Stand the **Street**	The Ten Bung Bade
band	The Shine **Street**	Back Sours
church	**St.** Mary's Church*	Church of St. Carnew
airport	Shanger Airport	Barre Airport
building	**St.** John House*	Conneyate House
high school	San High School	South Carrool High School
person	Marit Balles	Anterr Chares
road	Marylon Route 20	Maryland Route 62*
ship	USS Mark*	USS Senton
city	**Santa** Mara	Karango
railway station	**Santa** Station	Sanan Railway Station
mountain	Mount Sungan	Mount South

Figure 2.13: Predictions from RNN trained with different criterions: log-likelihood (left) and multi-conditional (right). Note that for the generative criterion, the predictions show some repeated patterns in different classes. The MCL criterion appears to encourage more differentiated classes. Predictions which correspond exactly to an entry in the knowledge base a marked with an asterisk (*).

2.8 Summary

In this chapter, we have proposed different approaches to character-level modeling of entity names. We have shown experimentally that neural networks—in particular stacked recurrent LSTMs—outperform state-of-the-art Bayesian non-parametric n-gram alternatives for purposes of classifying name strings into one of several possible types via Bayes' rule. We proposed a joint model in which the class label is an input, and all RNN parameters are shared between classes. This joint model has fewer parameters and outperforms a model with separate parameters for each class. We additionally proposed using the MCL criterion for training directly to optimize the classification objective, and show that it leads to improved performance over log-likelihood training. We looked at incorporating entity attributes into the name models,

and we presented qualitative results showing sensitivity to attributes such as gender and nationality.

2.9 Related Work

The problem of classifying sequences into one of several classes occurs in many applications. The methods we explore here—character-level sequence models—are not specific to names and could be applied to these other problems. For instance, *language identification* involves classifying the language of a document or shorter text [Torres-Carrasquillo et al., 2002, Lui and Baldwin, 2012]. Simple generative models, such as naive Bayes, are commonly used for this purpose [Lewis, 1998]. In the speech community, there has recently been some work using recurrent neural networks in order to do language identification [Gonzalez-Dominguez et al., 2014], and their approach bears some similarity to our RNN model for names. In document classification, a document must be labeled with a corresponding class label (e.g. SPORTS). Both simple generative methods like naive Bayes as well as hierarchical Bayesian topic models have been explored [Blei et al., 2003, Rennie and Rifkin, 2001, Ramage et al., 2009] Discriminative methods have also been applied to these problems, notably *string kernels*, which consider features based on all subsequences up to a given length [Lodhi et al., 2002, Manevitz and Yousef, 2002, Rennie and Rifkin, 2001]. Sentiment classification may also be viewed as an instance of sequence labeling [Tang et al., 2015]. For sentiment,

more structured approaches using syntactic information have also been applied successfully [Nakagawa et al., 2010], though these rely on additional supervision. To our knowledge, we are the first to apply generative neural sequence models to the problem modeling entity name gazetteers. In the context of named entity recognition (§3) there is prior work on modeling name structure in generative models [Charniak, 2001, Elsner et al., 2009, Bhattacharya and Getoor], though these focus on the unsupervised setting and use less expressive count-based features, instead of RNNs as we do.

Chapter 3

Hidden Non-Markov Models

3.1 Chapter Overview

Lexical resources such as dictionaries and gazetteers of places and lists of names have proven valuable resources in NLP [Kazama and Torisawa, 2007, Kozareva, 2006, Toral and Munoz, 2006]. Such resources are typically used either as constraints or as a source of features for discriminative systems, which are different ways of *conditioning* on the lexical information.

In discriminative models, novel types may be handled via spelling features (e.g. character n-grams) and features which consider the context (e.g. surrounding words). Such features generalize from the observed training data and lexical resources. However, perhaps surprisingly, incorporating large gazetteers can sometimes hurt performance because of *weight under-training*. This is a type of over-fitting in which lexical features

are given excessive weight compared to these other features [Sutton et al., 2006, Smith and Osborne, 2006]. Thus, more data may result in lower performance—a pathological problem for a machine learning method.

Given the importance of lexical resources, it would be desirable to design a machine learning method which is able to incorporate these resources in a more robust manner. We note that generative modeling provides a principled solution: we construct a model that generates both type-level and token-level data. That is, we stipulate a generative process in which both lexical resources (type information) and traditional annotated resources (tokens in context) are produced by the same model (§3.2.2).

We propose a generative model in which the observed characters originate from a latent non-Markov process over words, and where the characters are themselves produced via a non-Markov process: a recurrent neural network. We will make use of the generative sequence models discussed in §2 for this purpose. In order to perform inference in this model, we propose a sampler in which sequential Monte Carlo (SMC) is used as a transition kernel for a Gibbs sampler. Since SMC is amenable to an efficient parallel implementation, our sampler is both computationally efficient and allows for high-dimensional moves.

We apply the proposed model to two problems:

- In §3.4, we show that the proposed model and associated inference method is effective at multi-lingual POS induction with dictionaries, outperforming a baseline generative method with manually-designed features.

- In §3.7, we show that our generative model is immune to the type of overfitting exhibited by discriminative models, while performing with comparable performance despite not using any manually-designed input features.

3.2 Hidden Non-Markov Models

To begin, we make some modest simplifying assumptions regarding the underlying text. The first assumption is that sentences are independent and identically distributed (iid). This means that we do not directly model any document or cross-document effects such as topic or coreference. We also assume that both the vocabulary and the set of entity types are closed and finite. This means that we fix the number of target entity types and the size of the vocabulary at training time, and neither increases at test time (even though novel types and classes may be encountered). As in a hidden Markov model (HMM), our model is factored into two parts: a transition model $p_{\boldsymbol{\phi}}$ and an emission model $p_{\boldsymbol{\theta}}$. We explore different parameterizations of these two factors, which are all based on the sequence models we explored in the previous chapter (§2).

So, we observe a corpus of iid sentences, each of which consists of a sequence of characters $\boldsymbol{x}$. We suppose that these characters originate from a sequence of latent states y_t for $t = 1, 2, \ldots$. A state may produce multiple words, such as a named-entity phrase, or only a single word, such as a part-of-speech. The latent state sequence is converted into the observed characters as follows. The first state y_1 is chosen from an

initial distribution $p_{\boldsymbol{\phi}}(y \mid \text{BOS})$, where BOS is a distinguished beginning-of-sentence state, which is always observed (i.e. we assume a known segmentation of the input documents into sentences). Conditioned on the initial state y_1, a sequence of characters $\boldsymbol{x}_1$ is emitted from a distribution specific to y_1 until a distinguished stop character # is emitted. Then the next term y_2 is chosen conditioned on all previous terms, but independently of the symbols emitted from y_1:

$$y_2 \sim p_{\boldsymbol{\phi}}(\cdot \mid \text{BOS}, y_1) \tag{3.1}$$

Just as for y_1, a sequence of characters $\boldsymbol{x}_2$ is emitted from a y_1-specific distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x} \mid)$:

$$\boldsymbol{x}_2 \sim p_{\boldsymbol{\theta}}(\cdot \mid y_2) \tag{3.2}$$

where $\boldsymbol{x}_2$ is again #-terminated. This process continues until the distinguished end-of-sequence state is picked, $y_{i+1} = \text{EOS}$. For generality, we assume the EOS state is—like all other states—associated with a distribution over characters. This distribution may emit characters such punctuation or other delimeters such as HTML tags. When sentence boundaries are given, this distribution is not important. If sentence boundaries are unknown, they may be identified by imputing the EOS term via marginal inference.

It remains to specify the form of the emission model $p_{\boldsymbol{\phi}}$ and the transition model $p_{\boldsymbol{\theta}}$. To begin, we use the sequence memoizer (SM), a hierarchical Bayesian language

model [Wood et al., 2009] which was previously introduced in §2. It is appealing for several reasons, which include fast estimation, a closed-form predictive distribution, and minimal assumptions about the underlying sequence. Notably, the SM makes no fixed Markov-order assumptions, which allows it to capture long-range dependencies. In §3.6, we explore a hybrid model that uses a recurrent neural network parametrization for the emission model, as well as a parametrization of both the context and emission models using RNNs.

3.2.1 Context model

We parametrize the transition distribution using a sequence memoizer (SM), a type of sequence model which was previously introduced in §2.4.1. Here, we use it to model sequences of words (including distinguished named-entity types), which in general are latent.

Let $\boldsymbol{y}_{1:t}$ denote a sequence of states drawn from a finite alphabet $\mathcal{Y}$ (which includes a distinguished stop state EOS). The SM assigns a probability to a sequence $(y_1, y_2, \ldots, y_T, \text{EOS})$ via

$$p(\boldsymbol{y}_{1:T}) = \prod_{t=1}^{T} \left[p(y_t \mid \boldsymbol{y}_{1:t-1})\right] p(\text{EOS} \mid \boldsymbol{y}_{1:T}) \tag{3.3}$$

$$= \prod_{t=1}^{T} \left[G_{\boldsymbol{y}_{1:t-1}}(y_t)\right] G_{\boldsymbol{y}_{1:T}}(\text{EOS}) \tag{3.4}$$

where $G_{\boldsymbol{u}}(x)$ denotes the conditional probability of symbol y occurring in context

$\boldsymbol{u} \in \mathcal{Y}^*$. The distribution $G_{\mathbf{u}}$ is a PYP where

$$G_\epsilon \sim \text{PYP}(d_\epsilon, \alpha_\epsilon, H_0) \tag{3.5}$$

$$G_{\boldsymbol{u}} \sim \text{PYP}(d_{|\mathbf{u}|}, \alpha_{|\mathbf{u}|}, G_{\sigma(\mathbf{u})}) \tag{3.6}$$

Here, ϵ is the empty sequence, H_0 is a distribution over $\mathcal{Y}$, and $\sigma(\mathbf{u})$ drops the first symbol from $\mathbf{u}$. We take H_0 to be uniform over a finite set $\mathcal{Y}$.

3.2.2 Emission model

Each latent state $y \in \mathcal{Y}$ indexes a conditional emission distribution $p(\boldsymbol{x} \mid y)$, where $\boldsymbol{x}$ are #-terminated strings. We define a Pitman-Yor process (*cf.* §2.3.1) over the set of strings $\boldsymbol{X}^*$, i.e. where the dish served at each table is a string (a sequence of characters, e.g. a word or phrase). Our construction here is similar to Mochihashi et al. [2009]: we use a nested Bayesian model where the base distribution of the top-level PYP adaptor is a distribution over the infinite set of character sequences; that is:

$$p(\boldsymbol{x} \mid y) = \text{PYP}(\alpha_y, d_y, H(\boldsymbol{x} \mid \boldsymbol{\theta})) \tag{3.7}$$

where $H(\boldsymbol{x} \mid \boldsymbol{\theta})$ is a character sequence model (*cf.* §2.4) with parameters $\boldsymbol{\theta}$.

As previously discussed, we are interested in incorporating type-level supervision. This is an unusual form of supervision since it consists of strings out of context.

However, the advantage of our decomposition into separate emission and transition models is that we may treat the lexicon as independently observed by the emission model.

We suppose that the lexicon consists of a finite number of states, where a state y corresponds to a part-of-speech or a named-entity type. Each state is associated with a restaurant, in which the tables are labeled with string *types* drawn from the corresponding base distribution $H(\boldsymbol{x} \mid \boldsymbol{\theta}_y)$. (For completeness, suppose the parameters $\boldsymbol{\theta}_y$ are drawn from some Gaussian distribution.)

Due to the exchangeability of the CRP, we may treat the lexicon types as the first customers to the $p(\boldsymbol{x} \mid y)$ restaurants for each y. As a result, each type-level observation creates a new table whose dish is the string corresponding to the lexicon entry. Any future string *tokens* will then be able to sit at the same table without having to pay the cost of generating the string spelling anew.

The type-level information is helpful in two ways:

1. It provides initial training data for the character sequence models $H(\boldsymbol{x} \mid \boldsymbol{\theta}_y)$ serving as base distributions for the emission models.

2. Unlikely strings under $H(\boldsymbol{x} \mid \boldsymbol{\theta}_y)$, but which appear in the lexicon, will be able to "cheaply" accept new customers thanks to the PYP adaptor.

We note that our approach here is similar to the host construction due originally to Dreyer and Eisner [2011a], in which the type-level supervision consists of known

morphological paradigms, which come labeled with particular lexemes. Each lexeme is assigned a reserved table, and these reserved tables have "hosts" which constrain strings to the paradigm.

3.3 Inference via Particle Gibbs

In a completely unsupervised setting, we observe iid character sequences, and must infer a distribution over latent state sequences compatible with the observations. We are interested in the posterior distribution over assignments to the hidden states $\{\boldsymbol{y}\}_{i=1}^{N}$ for each sentence $i = 1, \ldots, N$, marginalizing over other nuisance variables. Exact inference is intractable since the probability of transitioning from one latent state to the next is a function of all previous states. As a result, there is no benefit from dynamic programming over a brute force enumeration, and so the cost of exact inference increases exponentially with sequence length.

We therefore resort to approximate inference. Markov chain Monte Carlo (MCMC) methods are attractive since, with the proper choice of Markov chain kernel, they provide unbiased estimates of the target posterior [Gilks, 2005]. The accuracy of the method is adaptive in the sense that it may always be improved by taking more samples (i.e., running the sampler for longer). However, MCMC methods suffer from two practical issues:

- Many transition kernels result in slow mixing. For example, a simple kernel

for sequential state models like HMMs is a Gibbs sampler which resamples individual states, leaving all other states fixed. This approach, unfortunately, suffers from severely slow mixing in practice, since successive states are usually correlated [Goldwater and Griffiths, 2007, Gael et al., 2008].

- It is often difficult to parallelize MCMC methods since they are sequential by design. The aforementioned single-state Gibbs sampler requires processing each state in turn (conditioned on all others), which leaves no opportunity for resampling states in parallel.

Thus, we would like an MCMC kernel which is both (1) fast mixing and (2) computationally efficient. Our proposed solution is to use sequential Monte Carlo (SMC) as the transition kernel in an MCMC chain. We begin by describing the basic SMC approach [Doucet and Johansen, 2009].

SMC makes uses of an importance distribution (henceforth *proposal*), $q(\boldsymbol{y} \mid \boldsymbol{x})$, which decomposes as follows:

$$q_1(y_1 \mid \boldsymbol{x}) \prod_{t=1} q_t(y_t \mid \boldsymbol{x}, \boldsymbol{y}_{1:t-1}) \tag{3.8}$$

To sample from q_t: first, sample an initial state y_1 from q_1, then proceed incrementally, sampling y_t from q_t for $t = 2, \ldots, T$. This decomposition is important as it allows a high-dimensional sample to be constructed from a sequence of low-dimensional samples, which are (1) typically easier to construct and (2) more computationally

efficient.

Each sample constructed in this way is called a *particle*, and the SMC procedure consists of generating a system of M weighted particles. The importance weight for each particle m is computed incrementally:

$$w_t^{(m)} := w_{t-1}^{(m)} \frac{p(\boldsymbol{x}_{1:t}^{(m)}, \boldsymbol{y}_{1:t})}{p(\boldsymbol{x}_{t-1}^{(m)}, \boldsymbol{y}_{t-1}) q(y_t^{(m)} \mid \ldots)} \tag{3.9}$$

Note that the numerator corresponds to the unnormalized target density $p(\boldsymbol{y}_{1:t} \mid \boldsymbol{x}_{1:t})$. As $M \to \infty$, SMC provides a consistent estimate of the partition function Z via:

$$\hat{Z} = \frac{1}{M} \sum_{m=1}^{M} w^{(m)} \tag{3.10}$$

and samples from the weighted particle system are distributed as samples from the target posterior distribution [Doucet and Johansen, 2009].

Particle Gibbs. Note that SMC provides a biased estimate of the posterior. However, we may employ SMC as a kernel in a Markov chain Monte Carlo sampler (MCMC) [Andrieu et al., 2010]. In particular, we use a block Gibbs sampler in which we iteratively resample the hidden states of a sentence i conditioned on the assignments of all other latent variables (*conditional* SMC). For this sampler to be ergodic, we must fix one particle to be equal to the previous assignment of latent states before resampling; thus $M \geq 2$ is required. The resulting Gibbs sampler may be understood as a "collapsed" (or marginal) Gibbs sampler (Liu [1994]), where the augmented state variables (the

particles) are integrated out conditioned on other variables. A potential issue with this sampler is that a particularly good state sequence is chosen and is then hard to move from, resulting in poor mixing. We have not observed this issue in our experiments, in part because in our discrete setting and short sequence lengths, particle degeneracy is not as severe. However, it is worth noting that some recent work has gone into avoiding this issue in more general settings [Lindsten et al., 2012].

Proposal distribution. The choice of proposal distribution is crucial to the performance of sequential Monte Carlo methods. It is common to use the transition probability $p(y' \mid \boldsymbol{y}_{1:t-1})$ as the proposal, because it is usually available in closed form. The optimal proposal distribution is the one which minimizes the variance of the importance weights, and is given by:

$$\begin{aligned} q_t(y_t^{(m)}) &:= p(y_t \mid \boldsymbol{x}_{1:t-1}^{(m)}, \boldsymbol{y}_{1:t}) \\ &= \frac{p(y_t \mid \boldsymbol{x}_{1:t-1}^{(m)}) p(y_t \mid x_t)}{p(y_t \mid \boldsymbol{x}_{1:t-1}^{(m)})} \end{aligned} \tag{3.11}$$

where $p(y' \mid \boldsymbol{y}_{1:t-1})$ is defined in §3.2.1 and $p(y \mid x')$ is given in §3.2.2. Substituting this expression in Equation 3.9 and simplifying yields the following incremental weight update:

$$w_t^{(m)} := w_{t-1}^{(m)} \sum_{x' \in \mathcal{X}} p(x' \mid \boldsymbol{x}_{1:t-1}^{(m)}) p(y \mid x')$$

Particles are propagated by sampling y_t with probability proportional to Equation 3.11.

Resampling. In filtering applications, it is common to use resampling operations to prevent weight degeneracy. We do not find resampling necessary here for three reasons. First, we treat sentences as independent, so we resample hidden state sequences that are only as long as the number of words in a given sentence. Second, we use a proposal which minimizes the variance of the weights (in particular, we use an "optimal" proposal which incorporates the current observation). Finally, we use SMC as a kernel embedded in an MCMC sampler, and therefore the sampler will revisit the same block of variables repeatedly (conditioned on the previous particle assignment). Asymptotically, this procedure yields the desired posterior regardless of degeneracy, which for particle Gibbs only affects mixing time. Practically speaking, one can diagnose the need for resampling via the effective sample size (ESS) of the particle system, which is heuristically for defined for M particles as

$$\text{ESS} = \frac{1}{\sum_{m=1}^{M} w_m^2} \tag{3.12}$$

where w_m is the normalized importance weight for particle m. An ESS of 1 corresponds to a fully degenerate particle system, in which all weight is placed on a single particle. In our experiments, we find that ESS remains at a substatial fraction of M, even for longer sentences.

3.4 Part-of-Speech Induction

We report experiments on part-of-speech induction using a dictionary as type-level supervision. Note that the same word might occur in the dictionary more than once but with different parts-of-speech. We follow the experimental procedure described in Li et al. [2012]. Their model is a second-order maximum entropy Markov model parametrized with log-linear linear features (SHMM-ME). This model uses hand-crafted features designed to distinguish between different parts-of-speech, and special handling for rare words. In contrast, our model uses no hand-crafted features.

We evaluate our model on multi-lingual data released as part of the CoNLL 2007 and CoNLL X shared tasks. In particular, we use the same set of language as Li et al. [2012], with the exception of Dutch. (Unlike other CoNLL languages, the Dutch data includes phrases and the procedure by which these were split into tokens was not fully documented.) Concretely, we ran particle Gibbs for 100 training epochs, where one epoch consists of resampling the states for a each sentence in the corpus. We note that in practice, the model appears to mix rapidly and for some languages as little as 10 iterations may be required for the state of the sampler to correspond to a good solution. Also note that no *ad hoc* mapping between the inferred parts-of-speech and the true parts-of-speech is necessary, since the dictionary provides both methods with an initialization.

Discussion. Overall, our proposed model tends to outperform the baseline approach of Li et al. [2012] on most of the tested languages. This is an encouraging result, since

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

Table 3.1: Part-of-speech induction results in multiple languages.

our proposed model lacks the inductive bias from the manually designed features of the baseline model. Our model also converges quickly to an accurate solution, which was obtained by freezing the state of the sampler after a fixed number of iterations (in this case, we ran our Gibbs sampler for 50 epochs). Wiktionary contains entries with many ambiguous tags, with no special marking for the most frequent usage. This was a source of errors for both methods, but particularly so the baseline, which treats tags in Wiktionary as hard constraints rather than simply observations under our generative model.

3.5 Latent Segmentations

The inference procedure described in §3.3 may be generalized for the case where there is an unknown segmentation of the input. This problem arises in named-entity recognition, where the named-entities may span multiple words. We make the simplifying assumption that segment boundaries only occur at word boundaries; in other words, all characters within a word are assumed to have the same named-entity type.

The sampler proceeds as follows. For each latent state y, we consider two spe-

cializations: (1) I-y corresponds to a partially complete chunk spanning *at least* one more word, and (2) E-y corresponds to the final word in a chunk. For instance, I-PER E-PER is equivalent to labeling the entire name "Barack Obama" with the entity type PER. As a special case, the OTHER category is always associated with E, which encodes the assumption that context phrases never span more than one word. This is property of the annotated data—of course, there are multi-word context phrases in natural language. They just happen to be annotated as a sequence of OTHER words in our dataset.

Within a chunk, the previous state y_{t-1} is always I-y by construction. Let s denote the position in the sentence at which the segment began, so $\boldsymbol{y}_{s:t}$ denotes the word sequence associated with the chunk. At position t there are two possible actions associated with y_t: (1) either extend the segment to include the word at time $t+1$ and *continue* (I-y), or (2) extend the segment to include the word at time $t+1$ and *stop* (E-y).

To associate probabilities with these actions, we define the following useful quantities. Let $\boldsymbol{x}_{s:t}$ denote the span of characters starting at the beginning of word s, and ending at the end of word t. Let $p_{\text{prefix}}(\boldsymbol{x}_{s:t} \mid y)$ denote the total probability under the PYP emission model for y of emitting the prefix $\boldsymbol{x}_{s:t}$ followed by the space symbol. Similarly, define $p_{\text{halt}}(\boldsymbol{x}_{s:t} \mid y)$ as the probability of emitting the prefix $\boldsymbol{x}_{s:t}$ and

stopping, i.e. emitting the # symbol.

$$q_t(y) \propto \begin{cases} p_{\text{prefix}}(\boldsymbol{x}_{s:t} \mid \text{I-}y), & y = \text{I-}y \\ p_{\text{halt}}(\boldsymbol{x}_{s:t} \mid \text{I-}y), & y = \text{E-}y \end{cases} \tag{3.13}$$

Between states, one may transition to any other state I-y or E-y. In our experiments, we take the possible transitions to be I-{PER,ORG,LOC}, E-{PER,ORG,LOC}, and E-O for context words. The proposal probabilities for each of these states are given by:

$$q_t(y) \propto \begin{cases} p(y \mid \boldsymbol{y}^{(m)}_{1:t-1}) p_{\text{prefix}}(\boldsymbol{x}_{t:t} \mid y), & y = \text{I-}y \\ p(y \mid \boldsymbol{y}^{(m)}_{t:t-1}) p_{\text{halt}}(\boldsymbol{x}_{t:t} \mid y), & y = \text{E-}y \end{cases} \tag{3.14}$$

Between segments, the incremental importance weight is updated as in Equation 3.11. This concludes the description of the sampler for labeled segmentations, which generalizes the sampler used for POS induction. Note that the model itself has not changed, only the inference procedure.

3.6 A Memoized Neural Model

In §3.2, we proposed a hidden non-Markov sequence model. The model is non-Markov since the probability of a state depends on all previous states. The SM was

used for both the transition and emission distributions. As described in §2 (and experimentally validated), neural sequence models are more expressive than count-based models such as sequence memoizers. It would therefore be desirable to incorporate recurrent neural networks in both the transition and emission components of the model.

We experimented with a version of the model in which both the transition and emission factors are parametrized using stacked LSTMs. To be precise, we replace the base distribution of the emission model—a distribution over character sequences $\mathcal{X}^*$—with a neural sequence model (replacing the SM used for POS induction). We also re-parametrized the transition model with an LSTM over latent state sequences, which includes all context words in the vocabulary as well as distinguished named-entity types (PER, ORG, MISC, LOC).

Unfortunately, these substitutions result in over-fitting on the CoNLL data. The over-fitting manifests during inference where the context model becomes over-confident in certain types occurring in certain contexts, overwhelming evidence from the emission model (the name model). For instance, the CoNLL training data contains repeated sentences such as

PSV Eindhoven 3 3 0 0 11 3 9

where PSV Eindhoven is an organization. Thus, the ORG "word" will occur disproportionately at the beginning of sentences. This over-fitting leads to poor exploration of the posterior with SMC, and therefore incorrect segmentations on test data (and lower

F1 scores). We experimented with several *post-hoc* methods to improve performance:

- **Regularization.** To alleviate overfitting, we experimented with high dropout at every layer of the RNN. This yields improvements in F1 on the order of 1–2% on development data, compared to a baseline with no regularization. We also experimented with adding Gaussian noise to the estimated gradient, but this gave no benefit over dropout regularization.

- **Annealing the proposal distribution.** A significant improvement in F1 is possible by annealing the context factor in the proposal distribution of the SMC sampler. This effectively "flattens" the context factor, alleviating the over-confidence that is symptomatic of overfitting. On English CoNLL validation data, this improved the results from 76% to 80% F1.

- **Initialization of the word embedding.** A larger unlabeled corpus may be used to obtain initial word embeddings; however, an issue with this approach in the context of our model is that it contains latent-variables: the distinguished named-entity types. Nonetheless, for context words we may use pre-trained embeddings. To investigate the effect of these initializations, we employed pre-trained word-embeddings estimated via Hellinger PCA on English Wikipedia, Reuters, and WSJ [Lebret and Collobert, 2014]. We experimented with three different embedding sizes: 50, 100, and 200 dimensions. We also experiment with two strategies: (a) fixing the context word embeddings to the pre-trained

vectors, and (b) using the pre-trained vectors as initializations for the context words. The best combination was with the largest embeddings (200) and using the embeddings as an initialization, which together yield an improvement of approximately 1.9% on English CoNLL validation. In contrast, fixing the context word embeddings to the pre-trained vectors resulted in lower performance.

The combination of these techniques results in substantial improvements over a straightforward RNN parametrization. However, overall performance remains lower than a baseline which uses a sequence memoizer (SM) for the transition model over words and entity types, and a neural name model for the base distribution of the emission distribution. Therefore, for the experiments presented in §3.7, we use this hybrid model.

3.7 NER with the Memoizer-Neural Model

We simplify the inference problem as follows. We assume that we observe context word lemmas such as THE via a deterministic transformation of the observed string. That is, each of "The" and "the" are case-normalized to THE, and we take this to be an observation of a context type. In the generative story, we then emit the observed spelling of THE in the same manner as for named-entities. However, we will tie the

parameters of the context emission distribution, so that

$$p((t,h,e,\#) \mid \text{THE}) = p((t,h,e,\#) \mid \text{SAID}) \tag{3.15}$$

where $\{\text{THE}, \text{SAID}\} \in \mathcal{Y}$, i.e. index particular latent states. So there are emission distributions for each named-entity type, plus a distinguished emission distribution for context words.

Effectively, we are fixing an assignment of latent-variables to simplify the inference problem. Note however that since we observe a finite vocabulary at train time, we must account for novel types at test time. To do so, we assume a distinguished OOV state which is responsible for all non-entity context words that were not observed at training time. We suppose that all singleton terms are deterministically mapped to OOV at training time, and so the strings associated with OOV will tend to come from a diverse set of rare terms.

We form gazetteers by selecting words and named-entity spans at random, separately from the training set and development sets. In this way, we control the expected gazetteer coverage at train and test time. Note however that no special effort was made to prevent overlap between the train and test gazetteer sets; thus, a test gazetteer type may appear in the training set. In Figure 3.3 and Figure 3.6, the axes show the proportion of types selected at random from train and test sets.

We show results for our generative model and a canonical discriminative approach:

Feature	Description
WORD FORM	word identity
CLASS	entity type
CHAR 6-GRAMS	char n-grams up to length 6
PREVIOUS WORD	previous words
WORD CLASS PAIR	word and entity type
DISJUNCTIVE	combinations
TYPE SEQUENCES	sequences of labels
WORD SHAPE	encoding of the word shape

Table 3.2: Log-linear features used in the CRF baseline.

a conditional random field (CRF) [Wang et al., 2013]. The CRF baselines uses the feature templates shown in Table 3.2, with the addition of gazetteer features. We report the F1 performance measure, which is standard in NER [Nadeau and Sekine, 2007].

Discussion. The heat maps in Figures 3.6 and 3.3 show test performance for both the proposed generative approach as well as the baseline discriminative method. The diagonal of the heat maps correspond to a typical "in-domain" setting, in which gazetteer coverage is the same at train time and test time. The performance for the two methods is comparable in this setting, which is encouraging for the proposed approach as it relies on less manual feature-engineering. However, we see that for a given level of test set gazetteer coverage, the performance of disriminative method tends to *decrease* as train coverage *increases*, a result of overfitting the gazetteer features. In contrast, our proposed method is immune from this effect, simply treating the type-level lexical information as additional observations of the generative process.

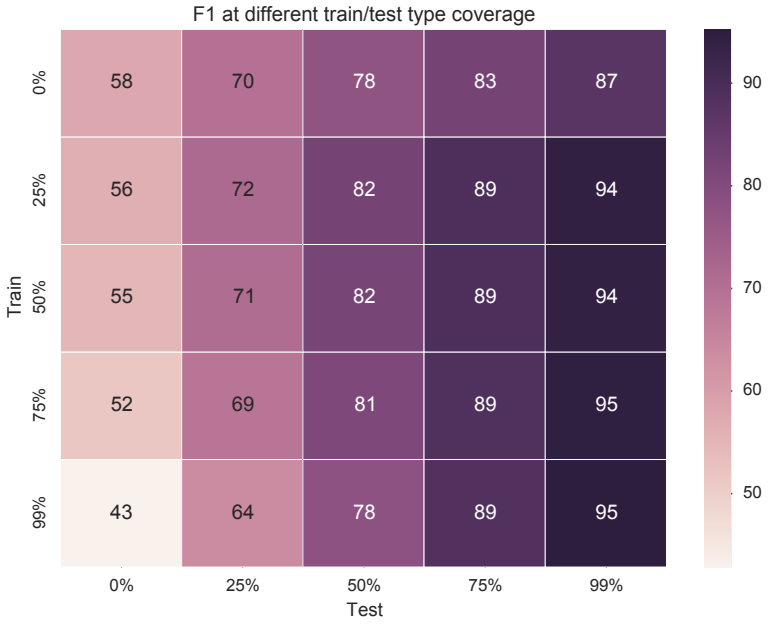

Figure 3.1: German: Discriminative model F1.

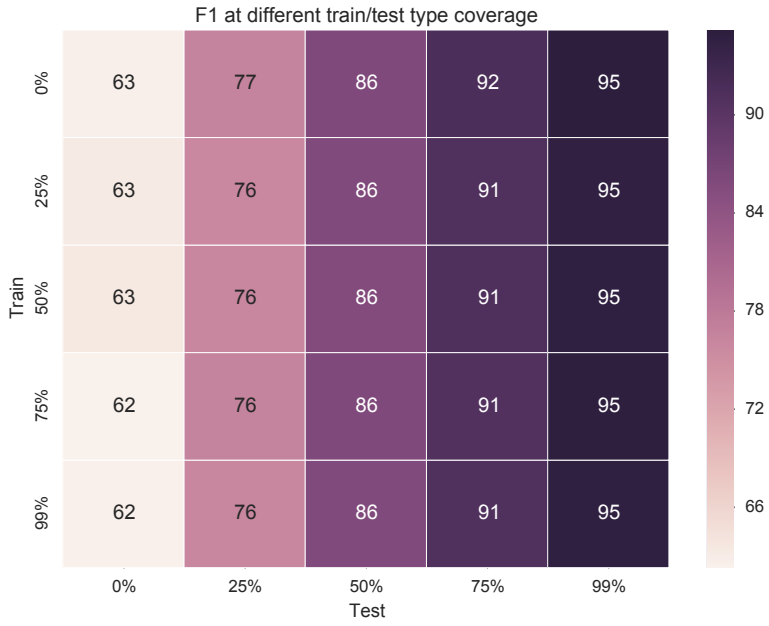

Figure 3.2: German: Generative model F1.

Figure 3.3: German CoNLL 2003.

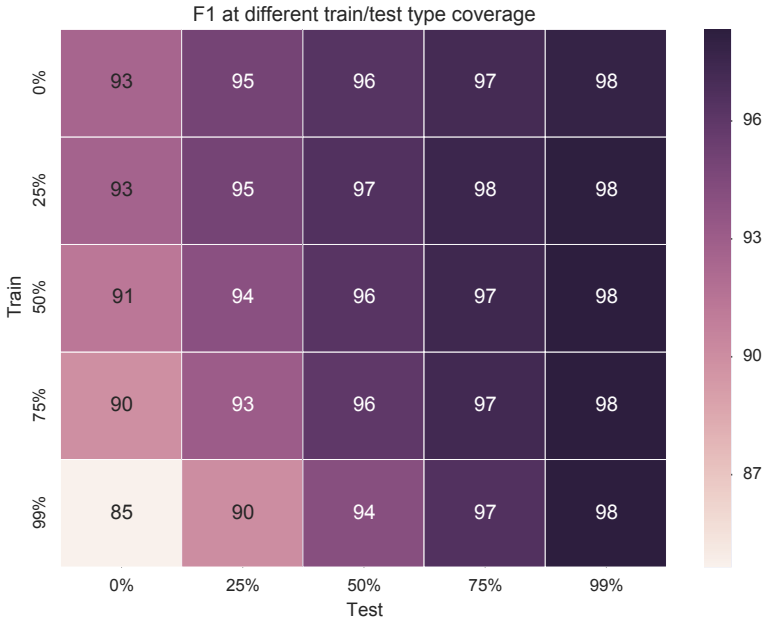

Figure 3.4: English: Discriminative model F1.

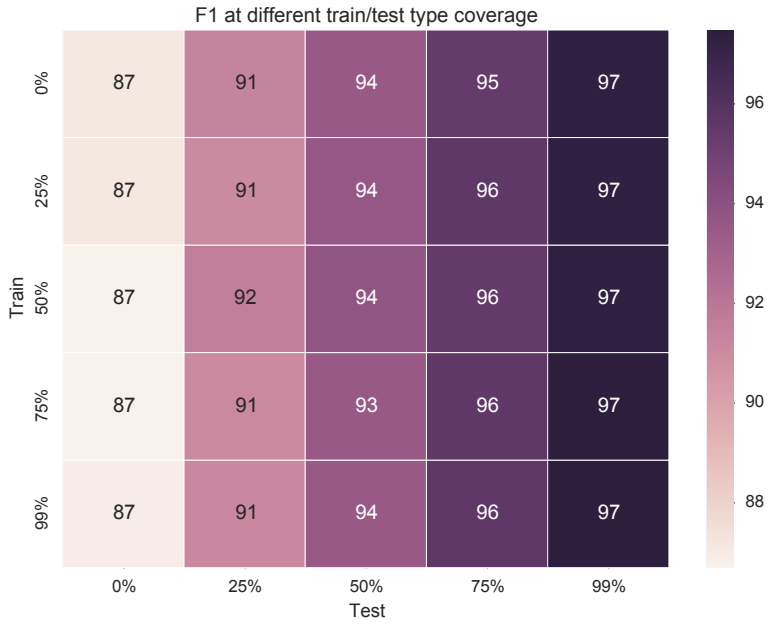

Figure 3.5: English: Generative model F1.

Figure 3.6: English CoNLL 2003.

3.8 Related Work

Unsupervised part-of-speech tagging is typically addressed using generative models [Christodoulopoulos et al., 2010]. Unfortunately, these models often make unrealistic assumptions about the data, and do not incorporate rich features. For instance, Goldwater and Griffiths [2007] use a count-based Bayesian non-parametric model, which makes fixed-order Markov assumptions. A similar Bayesian nonparametric approach but with milder modeling assumptions is Dubbin and Blunsom [2012], which also uses sequential Monte Carlo for inference.

The most successful approaches have used additional forms of supervision, and so effectively become *semi* supervised methods for POS induction. For instance, bilingual constraints have been used to guide inference [Das and Petrov, 2011]. Most related to our approach is Li et al. [2012], which uses dictionaries to constrain the possible parts-of-speech for lexical items contained in the dictionary. In constrast, we will treat the dictionary as observations under an assumed generative process. There have also been attempts at using minimal amounts of annotated data [Garrette and Baldridge, 2013].

Conditional random field (CRF) autoencoders are one approach at adapting discriminative training methods to unsupervised learning settings [Ammar et al., 2014]. Manually-designed features are used to provide an inductive bias in a procedure that alternates between predicting latent structure given observations, and observations given the latent structure.

Regarding named-entity recognition, most approaches considered state-of-the-art have been discriminative—that is, they model the target labeled segmentation *conditioned* on the input [Tjong Kim Sang and De Meulder, 2003]. An early approach based a hidden Markov model is Zhou and Su [2002]. More recently, Elsner et al. [2009] propose a structured word-level model for entity names. In constrast, we propose a character-level model which makes no assumptions concerning name structure, and is therefore applicable across different entity types and languages without modification.

Our inclination to directly model the data with minimal assumptions encoded in the model is related to the "from scratch" neural network approach of Collobert et al. [2011]. However, their approach relies on full supervision, and uses a heuristic approach for decoding, while we propose a principled joint model and associated probabilistic inference methods. This enables re-estimation of the model on unlabeled data, as well as the incorporation of lexical resources without concerns of over-fitting.

Chapter 4

Learning String-to-String Transducers via Phylogenetic Inference

4.1 Introduction

We have thus far considered strings in isolation. In this chapter, we propose a generative model in which some strings are not generated *ab initio*, but rather via transformative linguistic processes such as abbreviation, morphological derivation, historical sound or spelling change, loanword formation, translation, transliteration, editing, or transcription error. We propose learning from an *unorganized* collection of strings rather than from pairs of strings known to be in some relation.

CHAPTER 4. LEARNING STRING-TO-STRING TRANSDUCERS VIA PHYLOGENETIC INFERENCE

Systematic relationships between pairs of strings are at the core of problems such as transliteration [Knight and Graehl, 1998], morphology [Dreyer and Eisner, 2011b], cross-document coreference resolution [Bagga and Baldwin, 1998b], canonicalization [Culotta et al., 2007], and paraphrasing [Barzilay and Lee, 2003]. Stochastic transducers such as probabilistic finite-state transducers are often used to capture such relationships. They model a conditional distribution $p(y \mid x)$, and are ordinarily trained on input-output *pairs* of strings [Dreyer et al., 2008b].

The difficulty is that most or all of these parent-child relationships are unobserved. We must reconstruct this evolutionary phylogeny. At the same time, we must fit the parameters of a model of the relevant linguistic process $p(y \mid x)$, which says what sort of children y might plausibly be derived from parent x. Learning this model of $p(y \mid x)$ helps us organize the training collection by reconstructing its phylogeny, and also permits us to generalize to new forms.

Conceptually, the model described in this chapter may be viewed as an extension to the generative story for name spellings (§2) and for named-entity recognition (§3). In particular, it stipulates that the spelling of a given named-entity mention may optionally depend on the spelling of a *previously observed name of an entity of the same type*. However, in order to compare with prior work, we evaluate the contributions in this chapter in isolation. Specifically, we assume that named-entities and their types are observed, e.g. by using a sample from our model in §3.

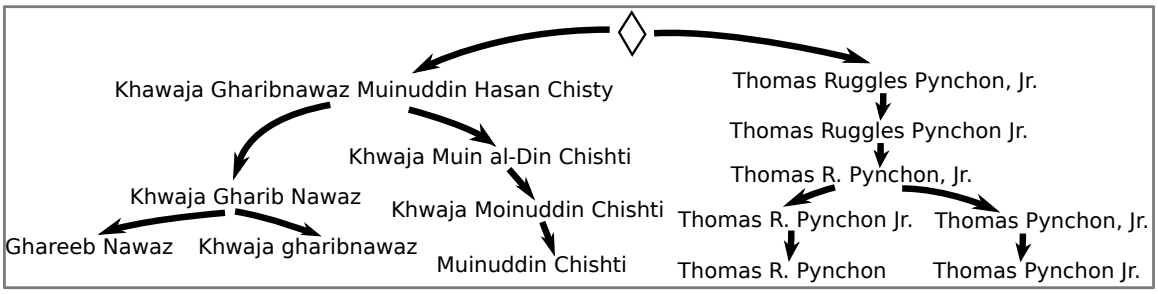

Figure 4.1: A portion of a spanning tree found by our model.

4.2 Name Phylogeny

We will focus on the problem of name variation. We observe a collection of person names—full names, nicknames, abbreviated or misspelled names, etc. Some of these names can refer to the same person; we hope to detect this. It would be an unlikely coincidence if two mentions of John Jacob Jingleheimer Schmidt referred to different people, since this is a long and unusual name. Similarly, John Jacob Jingelhimer Smith and Dr. J. J. Jingleheimer may also be related names for this person. That is, these names may be derived from one another, via unseen relationships, although we cannot be sure.

We propose a generative process that makes explicit assumptions about how strings are copied with possible mutation (edits). It is assumed to have generated all the names in the collection, in an unknown order. Given learned parameters, we can ask the model whether a name Dr. J. J. Jingelheimer in the collection is more likely to have been generated from scratch, or derived from some previous name in the collection.

A fragment of a phylogeny for person names is shown in Figure 4.1. Our procedure

learned this automatically from a collection of name tokens, without observing any input/output pairs. The nodes of the phylogeny are the observed name types, each one associated with a count of observed tokens.

Each arrow corresponds to a hypothesized mutation. These mutations reflect linguistic processes such as misspelling, initialism, nicknaming, transliteration, etc. As an exception, however, each arrow from the distinguished root node $\diamondsuit$ generates an initial name for a new entity. The descendants of this initial name are other names that subsequently evolved for that entity. Thus, the child subtrees of $\diamondsuit$ give a partition of the name types into entities.

Thanks to the phylogeny, the seemingly disparate names Ghareeb Nawaz and Muinuddin Chishti are seen to refer to the same entity. They may be traced back to their common ancestor Khawaja Gharibnawaz Muinuddin Hasan Chisty, from which both were derived via successive mutations.

Not shown in Figure 4.1 is our learned family p of conditional probability distributions, which models the likely mutations in this corpus. Via the EM algorithm, we estimate p jointly with the phylogeny [A. P. Dempster, 1977]. This procedure alternates between improving p and estimating the distribution over phylogenies. At the end, we extract the single best phylogeny (the "viterbi" spanning tree).

Together, the learned p and the phylogeny in Figure 4.1 form an *explanation* of the observed collection of names. What makes it more probable than other explanations? Informally, two properties:

- Each node in the tree is plausibly derived from its parent. More precisely, the product of the edge probabilities under p is comparatively high. A different p would have reduced the probability of the events in this phylogeny. A different phylogeny would have involved a more improbable collection of events, such as replacing Chishti with Pynchon, or generating many unrelated copies of Pynchon directly from $\diamondsuit$.

- In the phylogeny, the parent names tend to be used often enough that it is plausible for variants of these names to have emerged. Our model says that new tokens are derived from previously generated tokens. Thus—other things equal—Barack Obama is more plausibly a variant of Barack Obama, Jr. than of Barack Obama, Sr. (which has fewer tokens).

The above is merely an informal description of how statistical inference will behave under our model of the data. We now state the model more precisely.

4.2.1 Generative Story: Simple Version

Our model should reflect the *reasons* that name variation exists. A named entity has the form $y = (e, w)$ where w is a string being used to refer to entity e. A single entity e may be referred to on different occasions by different name strings w. We suppose that this is the result of *copying* the entity with occasional *mutation* of its name (by analogy to asexual reproduction).

Thus, we assume the following simple generative process that produces an *ordered sequence* of tokens $y_1, y_2, \ldots$, where $y_i = (e_i, w_i)$. We proposed an expanded generative process in §4.6.1, which also incorporates various contextual features of the entity names.

- After the first k tokens $y_1, \ldots y_k$ have been generated, the author responsible for generating y_{k+1} must choose whom to talk about next. She is likely to think of someone she has heard about often in the past. So to make this choice, she selects one of the previous tokens y_i uniformly at random, each having probability $1/(k+\alpha)$; or else she selects $\diamondsuit$, with probability $\alpha/(k+\alpha)$.

- If the author selected a previous token y_i, then with probability $1-\mu$ she copies it faithfully, so $y_{k+1} = y_i$. But with probability μ, she instead draws a mutated token $y_{k+1} = (e_{k+1}, w_{k+1})$ from the mutation model $p(\cdot \mid y_i)$, described in §4.3. This preserves the entity ($e_{k+1} = e_i$ with probability 1), but the new name w_{k+1} is a stochastic transduction of w_i drawn from $p(\cdot \mid w_i)$. Straightforward extensions are to allow a variable mutation rate $\mu(y_i)$ that depends on properties of y_i, and to allow w_{k+1} to depend on known properties of e_i. For example, in referring to e_i, the author may shorten and respell $w_i =$ Khwaja Gharib Nawaz into $w_{k+1} =$ Ghareeb Nawaz (Figure 4.1).

- If the author selected $\diamondsuit$, she must choose a fresh entity $y_{k+1} = (e_{k+1}, w_{k+1})$ to talk about. So she sets e_{k+1} to a newly created entity, sampling its name w_{k+1}

from the distribution $p(\cdot \mid \diamondsuit)$. For example, w_{k+1} = Thomas Ruggles Pynchon, Jr. (Figure 4.1). Nothing prevents w_{k+1} from being a name that is already in use for another entity (i.e., w_{k+1} may equal w_j for some $j \leq k$).

4.2.2 Relationship to other models

If we ignore the name strings, we can see that the sequence of entities $e_1, e_2, \ldots e_N$ is being generated from a Chinese restaurant process (CRP) with concentration parameter α [Aldous, 1985]. To the extent that α is low (so that $\diamond$ is rarely used), a few randomly chosen entities will dominate the corpus. The CRP is equivalent to sampling $e_1, e_2, \ldots$ IID from an unknown distribution that was itself drawn from a Dirichlet process with concentration α. This is indeed a standard model of a distribution over entities. For example, Hall et al. [2008] use it to model venues in bibliographic entries. From this characterization of the CRP, one can see that any permutation of this entity sequence would have the same probability. That is, our distribution over sequences of *entities* e is *exchangeable*.

However, our distribution over sequences of *named entities* $y = (e, w)$ is *non-exchangeable*. It assigns different probabilities to different orderings of the same tokens. This is because our model posits that later authors are influenced by earlier authors, copying entity names from them with mutation. So ordering is important. The mutation process is not symmetric—for example, Figure 4.1 reflects a tendency to shorten rather than lengthen names.

Non-exchangeability is one way that our present model differs from (parametric) transformation models [Eisner, 2002] and (non-parametric) transformation processes [Andrews and Eisner, 2011]. These too are defined using mutation of strings or other types. From a transformation process, one can draw a distribution over types, from which the tokens are then sampled IID. This results in an *exchangeable* sequence of tokens, just as in the Dirichlet process.

We avoid transformation models here for three reasons.

1. Inference is more expensive.

2. A transformation process seems less realistic as a model of authorship. It constructs a distribution over derivational *paths*, similar to the paths in Figure 4.1. It effectively says that each token is generated by recapitulating some previously used path from $\diamondsuit$, but with some chance of deviating *at each step*. For an author to generate a name token this way, she would have to know the whole derivational history of the previous name she was adapting. Our present model instead allows an author simply to select a name she previously saw and copy or mutate its surface form.

3. One should presumably prefer to explain a novel name y as a mutation of a *frequent* name x, other things equal (§4.2). But surprisingly, inference under the transformation process does not prefer this. The very fact that x has been frequently observed demonstrates that it has often chosen to stop mutating. This

implies that it is likely to choose stop again rather than mutate into y.

4.3 A Mutation Model for Name Strings

Let x denote a mention with parent $p = x.p$. We assume that its name $x.n$ is a stochastic transduction of its parent's name $p.n$ [Ristad and Yianilos, 1998]. That is, $p_{\boldsymbol{\theta}}(x.n \mid p.n)$ is given by the probability that applying a random sequence of edits to the characters of $p.n$ that yields $x.n$. The probabilities of different edits depend on parameters $\boldsymbol{\theta}$, which will be re-estimated from unlabeled data (i.e. we do not assume access to supervised pairs of input and output strings).

This process has four character-level edit operations: copy, substitute, insert, delete. It also has a distinguished no-edit operation that behaves exactly like copy. At each step, the process first randomly chooses whether to edit or no-edit, conditioned only on whether the previous operation was an edit. If it chooses to edit, it chooses a random edit type with some probability conditioned on the next input character. In the case of insert or substitute, it then randomly chooses an output character, conditioned on the type of edit *and* the next input character.

It is common to mutate a name by editing contiguous substrings (e.g., words). Contiguous regions of copying versus editing can be modeled by a low probability of transitioning between no-edit and edit regions. This somewhat resembles the traditional affine gap penalty in computational biology [Gusfield, 1997], which makes

deletions or insertions cheaper if they are consecutive. We instead make consecutive edits cheaper regardless of the edit type. Note that an edit region may include some copy edits (or substitute edits that replace a character with itself) without leaving the edit region. This is why we distinguish copy from no-edit.

The probability of a *single* edit sequence, which corresponds to a monotonic alignment of $x.n$ to $p.n$, is a product of individual edit probabilities of the form $p_{\boldsymbol{\theta}}(\binom{a}{b} \mid \hat{a})$, which is conditioned on the next input character $\hat{a}$. The edit $\binom{a}{b}$ replaces input $a \in \{\epsilon, \hat{a}\}$ with output $b \in \{\epsilon\} \cup \Sigma$ (where ϵ is the empty string and Σ is the alphabet of language $x.\ell$). Insertions and deletions are the cases where respectively $a = \epsilon$ or $b = \epsilon$—we do not allow both at once. All other edits are substitutions. When $\hat{a}$ is the special end-of-string symbol #, the only allowed edits are the insertion $\binom{\epsilon}{b}$ and the substitution $\binom{\#}{\#}$. We define the edit probability using a locally normalized log-linear model:

$$p_{\boldsymbol{\theta}}(\binom{a}{b} \mid \hat{a}) = \frac{\exp(\boldsymbol{\theta} \cdot \mathbf{f}(\hat{a}, a, b))}{\sum_{a', b'} \exp(\boldsymbol{\theta} \cdot \mathbf{f}(\hat{a}, a', b'))} \tag{4.1}$$

We use a small set of simple feature functions $\mathbf{f}$, which consider conjunctions of the attributes of the characters $\hat{a}$ and b: character, character class (letter, digit, etc.), and case (upper vs. lower). In §5.1, we discuss extensions to this approach that obviate the need for manual feature engineering using recurrent neural networks.

Notice that we use a locally normalized probability for each edit. This enables

faster and simpler training than the similar model of Dreyer et al. [2008a], which uses a globally normalized probability for the whole edit sequence. This is an important point since, in the most general inference setting, we must evaluate edit probabilities between all string pairs.

Our model of these distributions could incorporate detailed linguistic knowledge of the mutation process. Here we describe the specific model that we use in our experiments. Like many such models, it can be regarded as a stochastic finite-state string-to-string transducer parameterized by θ. That is, the probability (4.3) may also be conditioned on other variables such as on the languages $p.\ell$ and $x.\ell$—this leaves room for a transliteration model when $x.\ell \neq p.\ell$—and on the entity type $x.t$. The features in (4.1) may then depend on these variables as well.

When $p = \diamondsuit$, we are generating a new name $x.n$. We use the same model, taking $\diamondsuit.n$ to be the empty string (but with $\#_\diamondsuit$ rather than $\#$ as the end-of-string symbol). This yields a feature-based unigram language model (whose character probabilities may differ from usual insertion probabilities because they see $\#_\diamondsuit$ as the lookahead character).

Note that the conditional distribution for $p = \diamondsuit$ is similar to the neural name models that are used in §2 and §3. There are two reasons we avoid using those more expressive models here:

1. If the name model at $\diamondsuit$ is more expressive than the edit models, the model may learn to generate names preferentially from $\diamondsuit$ as new entities rather than as

mutations of existing names. One may perhaps offset this by tuning hyperparameters, but in general in unsupervised learning one must be careful to imbue the model with the right inductive biases. In §**??**, we describe an extension in which both the edit model and the mutation model may be parametrized using equally expressive neural models.

2. Since our training procedure requires (approximate) inference via sampling, there would be a substantial performance cost to using more complex transducer models as we must repeatedly score configurations of hidden variables. Therefore, for our experiments, we prefer a simpler model in which dynamic programming is tractable and which uses simple features. However, it is worth noting that there are several ways in which these computational concerns could be addressed, such as using efficient hardware parallelization.

4.3.1 Pragmatics

We can optionally make the model more sophisticated. Authors tend to *avoid* names $x.n$ that readers would misinterpret (given the previously generated names). The edit model thinks that $p_{\boldsymbol{\theta}}(\mathsf{CIA} \mid \diamondsuit)$ is relatively high (because CIA is a short string) and so is $p_{\boldsymbol{\theta}}(\mathsf{CIA} \mid \mathsf{Chuck's\ Ice\ Art})$. But in fact, if CIA has already been frequently used to refer to the Central Intelligence Agency, then an author is unlikely to use it for a different entity.

To model this pragmatic effect, we multiply our definition of $p_{\boldsymbol{\theta}}(x.n \mid p.n)$ by an extra factor $p(x.e \mid x)^{\gamma}$, where $\gamma \geq 0$ is the effect strength. Currently, we omit the step of renormalizing this deficient model. Our training procedure also ignores the extra factor. Here $p(x.e \mid x)$ is the probability that a reader correctly identifies the entity $x.e$. We take this to be the probability that a reader who knows our sub-models would guess some parent having the correct entity (or $\diamondsuit$ if x is a first mention): $\sum_{p':p'.e=x.e} w(p', x) / \sum_{p'} w(p', x)$. Here p' ranges over mentions (including $\diamondsuit$) that precede x in the ordering $\boldsymbol{i}$, and $w(p', x)$—defined later in §4.7.3—is proportional to the posterior probability that $x.p = p'$, given name $x.n$ and topic $x.z$.

4.4 Inference

The input to inference is a collection of named entity tokens y. Most are *untagged tokens* of the form $y = (?, w)$. In a semi-supervised setting, however, some of the tokens may be *tagged tokens* of the form $y = (e, w)$, whose true entity is known. The entity tags place a constraint on the phylogeny, since each child subtree of $\diamondsuit$ must correspond to exactly one entity.

4.4.1 An unrealistically supervised setting

Suppose we were lucky enough to fully observe the *sequence* of named entity tokens $y_i = (e_i, w_i)$ produced by our generative model. That is, suppose all tokens were tagged

and we knew their ordering.

Yet there would still be something to infer: which tokens were derived from which previous tokens. This phylogeny is described by a spanning tree over the tokens. Let us see how to infer it.

For each potential edge $x \to y$ between named entity tokens, define $\delta(y \mid x)$ to be the probability of choosing x and copying it (possibly with mutation) to obtain y. So

$$\delta(y_j \mid \diamondsuit) = \alpha \, p(y_j \mid \diamondsuit) \tag{4.2}$$

$$\delta(y_j \mid y_i) = \mu \, p(y_j \mid y_i) + (1 - \mu)\mathbb{1}(y_j = y_i) \tag{4.3}$$

except that if $i \geq j$ or if $e_i \neq e_j$, then $\delta(y_j \mid y_i) = 0$ (since y_j can only be derived from an *earlier* token y_i with the *same entity*).

Now the prior probability of generating $y_1, \ldots y_N$ with a given phylogenetic tree is easily seen to be a product over all tree edges, $\prod_j \delta(y_j \mid \text{pa}(y_j))$ where $\text{pa}(y_j)$ is the parent of y_j. As a result, it is known that the following are *efficient to compute* from the $(N+1) \times (N+1)$ matrix of δ values (see §4.4.4):

(a) the max-probability spanning tree

(b) the total probability of all spanning trees

(c) the marginal probability of each edge, under the posterior distribution on spanning trees

(a) is our single best guess of the phylogeny. We use this during evaluation. (b) gives the model likelihood, i.e., the total probability of the observed data $y_1, \ldots y_N$. To locally maximize the model likelihood, (c) can serve as the E step of our EM algorithm (§4.5) for tuning our mutation model. The M step then retrains the mutation model's parameters θ on input-output pairs $w_i \rightarrow w_j$, weighting each pair by its edge's posterior marginal probability (c), since that is the expected count of a $w_i \rightarrow w_j$ mutation. This computation is iterated.

Note that a phylogeny partitions the name types into some number of "clusters," where each cluster corresponds to a child subtree of $\diamondsuit$ and represents an entity. We can increase the number of "clusters" inferred by our method by increasing the ratio α/μ, which controls the preference for an entity to descend from $\diamondsuit$ versus an existing entity.

4.4.2 The unsupervised setting

Now we turn to a real setting—fully unsupervised data. Two issues will force us to use an approximate inference algorithm. First, we have an *untagged* corpus: a token's entity tag e is never observed. Second, the *order* of the tokens is not observed, so we do not know which other tokens are candidate parents.

Our first approximation is to consider only phylogenies over *types* rather than tokens. Working over types improves the quality of our second approximation, and also speeds up the spanning tree algorithms. §4.5 explains how to regard this approximation

as variational EM. The type phylogeny in Figure 4.1 represents a set of possible token phylogenies. Each node of Figure 4.1 represents an untagged name type $y = (?, w)$. By grouping all n_y tokens of this type into a single node, we mean that the *first* token of y was derived by mutation from the parent node, while each *later* token of y was derived by copying an (unspecified) earlier token of y.

A token phylogeny *cannot* be represented in this way if two or more tokens of y were created by mutations. In that case, their name strings are equal only *by coincidence*. They may have different parents (perhaps of different entities), whereas the y node in a type phylogeny can have only one parent.

We argue, however, that these unrepresentable token phylogenies are comparatively unlikely *a posteriori* and can be reasonably ignored during inference.The first token of y is necessarily a mutation, but later tokens are much more likely to be copies.

The probability of generating a later token y by copying some previous token is *at least*

$$(1 - \mu)/(N + \alpha),$$

while the probability of generating it in some other way is *at most*

$$\max(\alpha\, p(y \mid \diamondsuit),\ \ \mu \max_{x \in \mathcal{Y}} p(y \mid x))$$

where $\mathcal{Y}$ is the set of observed types.

The second probability is typically much smaller: an author is unlikely to invent

exactly the observed string y, certainly from $\diamondsuit$ but even by mutating a similar string x (especially when the mutation rate μ is small).

How do we evaluate a type phylogeny? Consider the probability of generating untagged tokens $y_1, \ldots y_N$ in that order and respecting the phylogeny:

$$\left(\prod_{k=1}^{N} \frac{1}{k+\alpha}\right) \prod_{y \in \mathcal{Y}} g(y \mid \mathrm{pa}(y)) \left(\prod_{i=1}^{n_y - 1} i\,(1-\mu)\right) \tag{4.4}$$

where $g(y \mid \mathrm{pa}(y))$ is a factor for generating the *first* token of y from its parent $\mathrm{pa}(y)$, defined by

$$g(y \mid \diamondsuit) = \alpha \cdot p(y \mid \diamondsuit) \tag{4.5}$$

$$g(y \mid x) = \mu \cdot (\#\text{ tokens of } x \text{ preceding first token of } y) \cdot p(y \mid x) \tag{4.6}$$

But we do not actually know the token order: by assumption, our input corpus is only an *unordered* bag of tokens. So we must treat the hidden ordering like any other hidden variable and maximize the *marginal* likelihood, which sums (4.4) over all *possible* orderings (permutations). This sum can be regarded as the number of permutations $N!$ (which is fixed given the corpus) times the expectation of (4.4) for a permutation chosen uniformly at random.

This leads to our second approximation. We approximate this expectation of the product (4.4) with a product of expectations of its individual factors. In general this

is an overestimate for each phylogeny. To find the expectation of (4.6), observe that the *expected* number of tokens of x that precede the first token of y is $n_x/(n_y+1)$, since each of the n_x tokens of x has a $1/(n_y+1)$ chance of falling before all n_y tokens of y. It follows that the approximated probability of generating all tokens in *some* order, with our given type parentage, is *proportional to*

$$\prod_{y \in \mathcal{Y}} \delta(y \mid \mathrm{pa}(y)) \tag{4.7}$$

where

$$\delta(y \mid \diamondsuit) = \alpha \cdot p(y \mid \diamondsuit) \tag{4.8}$$

$$\delta(y \mid x) = \mu \cdot p(y \mid x) \cdot n_x/(n_y+1) \tag{4.9}$$

and the constant of proportionality depends on the corpus.

The above equations are analogous to those in §4.4.1. Again, the approximate posterior probability of a given type parentage tree is *edge-factored*—it is the product of individual edge weights defined by δ. Thus, we are again eligible to use the spanning tree algorithms in described in §4.4.4.

Notice that n_x in the numerator of (4.9) means that y is more likely to select a frequent x as its parent. Also, $n_y + 1$ in the denominator means that a frequent y is not as likely to have any parent $x \neq \diamondsuit$, because its first token probably falls early in the sequence where there are fewer available parents $x \neq \diamondsuit$.

4.4.3 The semi-supervised setting

The semi-supervised setting is like the unsupervised setting—except that some of our tokens are now tagged with the true entity. Inference should only consider phylogenies that respect these tags.

The supervised information may break §4.4.2's assumption of "one entity per string," by revealing that there are in fact multiple John Jacob Jingleheimer Schmidts. In this case, we can no longer force all tokens of a string w into the same node of the phylogeny on the grounds that w only evolved once. But our inference will continue to assume, on the same grounds as §4.4.2, that it evolved the *minimum* possible number of times. (In computational biology jargon, we are considering only *parsiminious* phylogenies.)

What are the nodes, then? If w is tagged with k different entities, then we know that it evolved independently k times (at least). In our current experiments, the case $k > 1$ never actually arises, because of the way our training corpus was constructed. We create a separate node in the phylogeny for each of the tagged types $(e_1, w), \ldots, (e_k, w)$. We then partition untagged tokens $(?, w)$ among these nodes in proportion to the number of tagged tokens at each node. Only if $k = 0$ must we create a node for the untagged type $(?, w)$.

Given these nodes, the supervision constrains the possible phylogenies:

C1 If types x, y are tagged with different entities, they must not be in the same child subtree of $\diamond$.

C2 If type x, y are tagged with the same entity, they must be in the same child subtree of $\diamond$.

Unfortunately, we cannot enforce constraint C1 or C2 within our spanning tree algorithm. So our approach is to enforce only the weaker constraint C1′, which can be done by setting $\delta(y \mid x) = 0$:

C1′ If types x, y are tagged with different entities, the tree must not have a direct $x \to y$ edge.

Another form of supervision appears if we are provided with some supervised input-output pairs. These may be used to train initial parameters for the transducer. In subsequent steps of EM, they should be incorporated into the M step, so that they are used (along with the pairs derived from the E step) to retrain the transducer.

4.4.4 Spanning tree algorithms

Define a complete directed graph G over the vertices $\mathcal{Y} \cup \{\diamondsuit\}$. The weight of an edge $x \to y$ is defined by $\delta(y \mid x)$.

The (approximate) posterior probability of a given phylogeny given our evidence, is proportional to the product of the δ values of its edges.

Formally, let $\mathcal{T}_\diamondsuit(G)$ denote the set of spanning trees of G rooted at $\diamondsuit$, and define the weight of a particular spanning tree $T \in \mathcal{T}_\diamondsuit(G)$ to be the product of the weights

of its edges:

$$w(T) = \prod_{(x \to y) \in T} \delta(y \mid x) \tag{4.10}$$

Then the posterior probability of spanning tree T is

$$p_\theta(T) = \frac{w(T)}{Z(G)} \tag{4.11}$$

where $Z(G) = \sum_{T \in \mathcal{T}_\diamond(G)} w(T)$ is the partition function, i.e. the total probability of generating the data G via any spanning tree of the form we consider. This distribution is determined by the parameters θ of the transducer p_θ, along with the ratio α/μ.

There exist several algorithms to find the single maximum-probability spanning tree, notably Tarjan's implementation of the Chu-Liu-Edmonds algorithm, which runs in $O(m \log n)$ for a sparse graph or $O(n^2)$ for a dense graph [Cheriton and Tarjan, 1976]. Figure 4.1 shows a spanning tree found by our model using Tarjan's algorithm. Here n is the number of vertices (in our case, types and $\diamond$), while m is the number of edges.

4.5 Training the Transducer with EM

Our inference algorithm assumes that we know the transducer parameters θ. We now explain how to optimize θ to maximize the marginal likelihood of the training data.

This marginal likelihood sums over all the other latent variables in the model—the spanning tree, the alignments between strings, and the hidden token ordering.

The EM procedure repeats the following until convergence:

E-step: Given θ, compute the posterior marginal probabilities c_{xy} of all possible phylogeny edges.

M-step Given all c_{xy}, retrain θ to assign a high conditional probability to the mutations on the probable edges.

This describes a variational EM algorithm: our E step approximates the true distribution q over all phylogenies with the closest distribution p that assigns positive probability only to type-based phylogenies. This distribution is given by (4.11) and minimizes $\mathrm{KL}(p \mid\mid q)$. We argued in section §4.4.2 that it should be a good approximation. The posterior marginal probability of a directed edge from vertex x to vertex y, according to (4.11), is

$$c_{xy} = \sum_{T \in \mathcal{T}_\diamond(G): (x \to y) \in T} p_\theta(T) \tag{4.12}$$

The probability c_{xy} is a "pseudocount" for the expected number of mutations from x to y. This is at most 1 under our assumptions.

Calculating c_{xy} requires summing over all spanning trees of G, of which there are n^{n-2} for a fully connected graph with n vertices. Fortunately, Tutte [1984] shows how to compute this sum by the following method, which extends Kirchhoff's classical

matrix-tree theorem to weighted directed graphs. This result has previously been employed in non-projective dependency parsing [Koo et al., 2007, Smith and Smith, 2007].

Let $\mathbf{L} \in \mathbb{R}^{n \times n}$ denote the Laplacian of G, namely

$$\mathbf{L} = \begin{cases} \sum_{x'} \delta(y \mid x') & \text{if } x = y \\ -\delta(y \mid x) & \text{if } x \neq y \end{cases} \tag{4.13}$$

Tutte's theorem relates the determinant of the Laplacian to the spanning trees in graph G. In particular, the cofactor $\mathbf{L}^{0,0}$ equals the total weight of all directed spanning trees rooted at node 0. This yields the partition function $Z(G)$ (assuming node 0 is $\diamondsuit$).

Let $\hat{\mathbf{L}}$ be the matrix $\mathbf{L}$ with the 0th row and 0th column removed. Then the edge marginals of interest are related to the log partition function by

$$c_{xy} = \frac{\partial \log Z(G)}{\partial \, \delta(y \mid x)} = \frac{\partial \log |\hat{\mathbf{L}}|}{\partial \, \delta(y \mid x)} \tag{4.14}$$

which has the closed-form solution

$$c_{xy} = \begin{cases} \delta(y \mid \diamondsuit) \, \hat{\mathbf{L}}^{-1}_{yy} & \text{if } x = y \\ \delta(y \mid x) \, (\hat{\mathbf{L}}^{-1}_{xx} - \hat{\mathbf{L}}^{-1}_{xy}) & \text{if } x \neq y \end{cases} \tag{4.15}$$

See [Koo et al., 2007] for a derivation. Thus, computing *all* edge marginals reduces

to computing a matrix inverse, which may be done in $O(n^3)$ time.

At the M step, we retrain the mutation model parameters θ to maximize

$$\sum_{xy} c_{xy} \log p(w_y \mid w_x) \tag{4.16}$$

This is tantamount to maximum conditional likelihood training on a supervised collection of (w_x, w_y) pairs that are respectively weighted by c_{xy}.

The M step is nontrivial because the term $p(w_y \mid w_x)$ sums over a hidden alignment between two strings. It may be performed by an inner loop of EM, where the E step uses dynamic programming to efficiently consider all possible alignments, as in [Ristad and Yianilos, 1996]. In practice, we have found it effective to take only a single step of this inner loop. Such a Generalized EM procedure enjoys the same convergence properties as EM, but may reach a local optimum faster [Dempster et al., 1977].

4.6 Modeling Names in Context

Even the best model of name similarity is not enough by itself, since two names that are similar—even identical—do not *necessarily* corefer. Document context is needed to determine whether they may be talking about two different people. We now propose an extension to the generative process described in §4.2.1 for jointly (1) learning similarity between names and (2) clustering name mentions into entities, the two major components of cross-document coreference resolution systems [Baron and

Freedman, 2008, Finin et al., 2009, Rao et al., 2010, Singh et al., 2011, Lee et al., 2012, Green et al., 2012]. The main extension is that we will also consider entity context. This extension motivates a different inference procedure based on block Gibbs sampling.

Let $\boldsymbol{x} = (x_1, \ldots, x_N)$ denote an ordered sequence of distinct named-entity mentions in documents $\boldsymbol{d} = (d_1, \ldots, d_D)$. We assume that each document has a (single) known language, and that its mentions and their types have been identified by a named-entity recognizer. We use the object-oriented notation $x.v$ for attribute v of mention x.

Our model generates an ordered sequence $\boldsymbol{x}$ although we do not observe its order. Thus each mention x has *latent* position $x.i$ (e.g., $x_{729}.i = 729$). The entire corpus, including these entities, is generated according to standard topic model assumptions; we first generate a topic distribution for a document, then sample topics and words for the document [Blei et al., 2003]. However, any topic may generate an entity type, e.g. Person, which is then replaced by a specific name: when Person is generated, the model chooses a *previous* mention of any person and copies it, perhaps mutating its name.

Note that this differs from our word-level model in §3.2.1, which generated words sequentially. The reason to prefer a topic model here, which treats documents as bags-of-words, is that for cross-document coreference resolution the *d*ocument-level topic interactions would be more useful in distinguishing between entities with similar names. However, we leave evaluating this hypothesis experimentally for future work.

We make the closed-world assumption that the author is only aware of previous

mentions *from our corpus*. This means that two mentions cannot be derived from a common ancestor outside our corpus. To mitigate this unrealistic assumption, we allow any ordering $\boldsymbol{x}$ of the observed mentions, not respecting document timestamps or forcing the mentions from a given document to be generated as a contiguous subsequence of $\boldsymbol{x}$. Alternatively, the model may manufacture a name for a new person, though the name itself may not be new.

If all previous *mentions* were equally likely, this would be a Chinese Restaurant Process (CRP) in which frequently mentioned *entities* are more likely to be mentioned again ("the rich get richer"). We refine that idea by saying that the current topic, language, and document influence the choice of which previous mention to copy, similar to the distance-dependent CRP [Blei and Frazier, 2011]. This will help distinguish multiple John Smith entities if they tend to appear in different contexts.

Unlike the ddCRP, our generative story is careful to prohibit derivational cycles: each mention is copied from a *previous* mention in the latent ordering. This is why our phylogeny is a *tree*, and why our sampler is more complex. Also unlike the ddCRP, we permit asymmetric "distances": if a certain topic or language likes to copy mentions from another, the compliment is not necessarily returned.

Formally, each mention x is derived from a parent mention $x.p$ where $x.p.i < x.i$ (the parent came first), $x.e = x.p.e$ (same entity) and $x.n$ is a copy or mutation of $x.p.n$. In the special case where x is a first mention of $x.e$, $x.p$ is the special symbol $\diamondsuit$, $x.e$ is a newly allocated entity of some appropriate type, and the name $x.n$ is generated

from scratch.

Our goal is to reconstruct mappings $\boldsymbol{p}, \boldsymbol{i}, \boldsymbol{z}$ that specify the latent properties of the mentions x. The mapping $\boldsymbol{p} : x \mapsto x.p$ forms a phylogenetic tree on the mentions, with root $\diamondsuit$. Each entity corresponds to a subtree that is rooted at some child of $\diamondsuit$. The mapping $\boldsymbol{i} : x \mapsto x.i$ gives an ordering consistent with that tree in the sense that $(\forall x) x.p.i < x.i$. Finally, the mapping $\boldsymbol{z} : x \mapsto x.z$ specifies, for each mention, the topic that generated it. While $\boldsymbol{i}$ and $\boldsymbol{z}$ are not necessary for creating coref clusters, they are needed to produce $\boldsymbol{p}$.

4.6.1 Generative Story: Full Version

We assume that the corpus $\boldsymbol{d}$ was generated as follows.

First, for each topic $z = 1, \ldots K$ and each language ℓ, choose a multinomial $\boldsymbol{\beta}_{z\ell}$ over the word vocabulary, from a symmetric Dirichlet with concentration parameter η. Then set $m = 0$ (entity count), $i = 0$ (mention count), and for each document index $d = 1, \ldots, D$:

1. Choose the document's length L and language ℓ. (The distributions used to choose these are unimportant because these variables are always observed.)

2. Choose its topic distribution $\boldsymbol{\psi}_d$ from an asymmetric Dirichlet prior with parameters $\boldsymbol{m}$ [Wallach et al., 2009].

3. For each token position $k = 1, \ldots, L$:

(a) Choose a topic $z_{dk} \sim \boldsymbol{\psi}_d$.

(b) Choose a word conditioned on the topic and language, $w_{dk} \sim \boldsymbol{\beta}_{z_{dk}\ell}$.

(c) If w_{dk} is a named entity type (Person, Place, Org, ...) rather than an ordinary word, then increment i and:

i. create a new mention x with

$$x.e.t = w_{dk} \quad x.d = d \qquad x.\ell = \ell$$

$$x.i = i \qquad x.z = z_{dk} \quad x.k = k$$

ii. Choose the parent $x.p$ from a distribution conditioned on the attributes just set (see §4.6.2).

iii. If $x.p = \diamondsuit$, increment m and set $x.e =$ a new entity e_m. Else set $x.e = x.p.e$.

iv. Choose $x.n$ from a distribution conditioned on $x.p.n$ and $x.\ell$ (see §4.3).

Notice that the tokens w_{dk} in document d are exchangeable: by collapsing out $\boldsymbol{\psi}_d$, we can regard them as having been generated from a CRP. Thus, for fixed values of the non-mention tokens and their topics, the probability of generating the mention sequence $\boldsymbol{x}$ is proportional to the product of the probabilities of the choices in step 3 at the positions dk where mentions were generated. These choices generate a topic $x.z$ (from the CRP for document d), a type $x.e.t$ (from $\beta_{x.z}$), a parent mention (from the distribution over previous mentions), and a name string (conditioned on the parent's name if any). §4.7 uses this fact to construct an MCMC sampler for the latent parts of $\boldsymbol{x}$.

4.6.2 Sub-model for parent selection

To select a parent for a mention x of type $t = x.e.t$, a simple model (as mentioned above) would be a CRP: each previous mention of the same type is selected with probability proportional to 1, and $\diamondsuit$ is selected with probability proportional to $\alpha_t > 0$. A larger choice of α_t results in smaller entity clusters, because it prefers to create new entities of type t rather than copying old ones.

We modify this story by re-weighting $\diamondsuit$ and previous mentions according to their relative suitability as the parent of x:

$$p_{\boldsymbol{\phi}}(x.p \mid x) = \frac{\exp\left(\boldsymbol{\phi} \cdot \mathbf{f}(x.p, x)\right)}{Z(x)} \tag{4.17}$$

where $x.p$ ranges over $\diamondsuit$ and all previous mentions of the same type as x, that is, mentions p such that $p.i < x.i$ and $p.e.t = x.e.t$. The normalizing constant $Z(x) \stackrel{\text{def}}{=} \sum_p \exp\left(\boldsymbol{\phi} \cdot \mathbf{f}(x.p, x)\right)$ is chosen so that the probabilities sum to 1.

This is a conditional log-linear model parameterized by $\boldsymbol{\phi}$, where $\phi_k \sim \mathcal{N}(0, \sigma_k^2)$. The features $\mathbf{f}$ are extracted from the attributes of x and $x.p$. Our most important feature tests whether $x.p.z = x.z$. This binary feature has a high weight if authors mainly choose mentions from the same topic. To model which (other) topics tend to be selected, we also have a binary feature for each parent topic $x.p.z$ and each topic pair $(x.p.z, x.z)$.

Many other features could be added. In a multilingual setting, one would similarly

want to model whether English authors select Arabic mentions. One could also imagine features that reward proximity in the generative order ($x.p.i \approx x.i$), local linguistic relationships (when $x.p.d = x.d$ and $x.p.k \approx x.k$), or social information flow (e.g., from mainstream media to Twitter). One could also make more specific versions of any feature by conjoining it with the entity type t.

4.7 Inference by Block Gibbs Sampling

We use a block Gibbs sampler, which from an initial state $(\boldsymbol{p}_0, \boldsymbol{i}_0, \boldsymbol{z}_0)$ repeats these steps:

1. Sample the ordering $\boldsymbol{i}$ from its conditional distribution given all other variables.
2. Sample the topic vector $\boldsymbol{z}$ likewise.
3. Sample the phylogeny $\boldsymbol{p}$ likewise.
4. Output the current sample $s_t = (\boldsymbol{p}, \boldsymbol{i}, \boldsymbol{z})$.

It is difficult to draw exact samples at steps 1 and 2. Thus, we sample $\boldsymbol{i}$ or $\boldsymbol{z}$ from a simpler proposal distribution, but correct the discrepancy using the Independent Metropolis-Hastings (IMH) strategy: with an appropriate probability, reject the proposed new value and instead use another copy of the current value [Tierney, 1994].

4.7.1 Resampling the ordering

We resample the ordering $\boldsymbol{i}$ of the mentions $\boldsymbol{x}$, conditioned on the other variables. The current phylogeny $\boldsymbol{p}$ already defines a partial order on $\boldsymbol{x}$, since each parent must precede its children. For instance, phylogeny (a) below requires $\diamondsuit \prec x$ and $\diamondsuit \prec y$. This partial order is compatible with 2 total orderings, $\diamondsuit \prec x \prec y$ and $\diamondsuit \prec y \prec x$. By contrast, phylogeny (b) requires the total ordering $\diamondsuit \prec x \prec y$.

We first sample an ordering $\boldsymbol{i}_\diamondsuit$ (the ordering of mentions with parent $\diamondsuit$, i.e. all mentions) *uniformly* at random from the set of orderings compatible with the current $\boldsymbol{p}$. However, such orderings are not in fact equiprobable given the other variables—some orderings better explain why that phylogeny was chosen in the first place, according to our competitive parent selection model (§4.6.2). To correct for this bias using IMH, we accept the proposed ordering $\boldsymbol{i}_\diamondsuit$ with probability

$$a = \min\left(1, \frac{p(\boldsymbol{p}, \boldsymbol{i}_\diamondsuit, \boldsymbol{z}, \boldsymbol{x} \mid \boldsymbol{\theta}, \boldsymbol{\phi})}{p(\boldsymbol{p}, \boldsymbol{i}, \ \boldsymbol{z}, \boldsymbol{x} \mid \boldsymbol{\theta}, \boldsymbol{\phi})}\right) \tag{4.18}$$

where $\boldsymbol{i}$ is the current ordering. Otherwise we reject $\boldsymbol{i}_\diamondsuit$ and reuse $\boldsymbol{i}$ for the new sample.

4.7.2 Resampling the topics

Each context word and each named entity is associated with a latent topic. The topics of *context words* are assumed exchangeable, and so we resample them using Gibbs sampling [Griffiths and Steyvers, 2004].

Unfortunately, this is prohibitively expensive for the (non-exchangeable) topics of the *named mentions* x. A Gibbs sampler would have to choose a new value for $x.z$ with probability proportional to the resulting joint probability of the full sample. This probability is expensive to evaluate because changing $x.z$ will change the probability of *many* edges in the current phylogeny $\boldsymbol{p}$. (Equation (4.17) puts x is in competition with other parents, so *every* mention y that follows x must recompute how happy it is with its current parent $y.p$.)

Rather than resampling one topic at a time, we resample $\boldsymbol{z}$ as a block. We use a proposal distribution for which block sampling is efficient, and use IMH to correct the error in this proposal distribution.

Our proposal distribution is an undirected graphical model whose random variables are the topics $\boldsymbol{z}$ and whose graph structure is given by the current phylogeny $\boldsymbol{p}$:

$$Q(\boldsymbol{z}) \propto \prod_{x \neq \diamondsuit} \Psi_x(x.z)\Psi_{x.p,x}(x.p.z, x.z) \tag{4.19}$$

$Q(\boldsymbol{z})$ is an approximation to the posterior distribution over $\boldsymbol{z}$. As detailed below, a proposal can be sampled from $Q(\boldsymbol{z})$ in time $O(|\boldsymbol{z}|K^2)$ where K is the number of topics, because the only interactions among topics are along the edges of the tree $\boldsymbol{p}$. The unary factor Ψ_x gives a weight for each possible value of $x.z$, and the binary factor $\Psi_{x.p,x}$ gives a weight for each possible value of the pair $(x.p.z, x.z)$.

The $\Psi_x(x.z)$ factors in (4.19) approximate the topic model's prior distribution

over $\boldsymbol{z}$. $\Psi_x(x.z)$ is proportional to the probability that a Gibbs sampling step for an ordinary topic model would choose this value of $x.z$. This depends on whether—in the current sample—$x.z$ is currently common in x's document and $x.t$ is commonly generated by $x.z$. It ignores the fact that we will also be resampling the topics of the other mentions.

The $\Psi_{x.p,x}$ factors in (4.19) approximate $p(\boldsymbol{p} \mid \boldsymbol{z}, \boldsymbol{i})$ (up to a constant factor), where $\boldsymbol{p}$ is the current phylogeny. Specifically, $\Psi_{x.p,x}$ approximates the probability of a single edge. It ought to be given by (4.17), but we use only the numerator of (4.17), which avoids modeling the competition among parents.

We sample from Q using standard methods, similar to sampling from a linear-chain CRF by running the backward algorithm followed by forward sampling. Specifically, we run the sum-product algorithm from the leaves up to the root $\diamondsuit$, at each node x computing the following for each topic z:

$$\beta_x(z) \stackrel{\text{def}}{=} \Psi_x(z) \cdot \prod_{y \in \text{children}(x)} \sum_{z'} \Psi_{x,y}(z, z') \cdot \beta_y(z')$$

Then we sample from the root down to the leaves, first sampling $\diamondsuit.z$ from $\beta_\diamondsuit$, then at each $x \neq \diamondsuit$ sampling the topic $x.z$ to be z with probability proportional to $\Psi_{x.p,x}(x.p.z, z) \cdot \beta_x(z)$.

Again we use IMH to correct for the bias in Q: we accept the resulting proposal $\hat{\boldsymbol{z}}$

with probability

$$\min\left(1,\ \frac{p(\boldsymbol{p},\boldsymbol{i},\hat{\boldsymbol{z}},\boldsymbol{x}\mid\boldsymbol{\theta},\boldsymbol{\phi})}{p(\boldsymbol{p},\boldsymbol{i},\boldsymbol{z},\boldsymbol{x}\mid\boldsymbol{\theta},\boldsymbol{\phi})}\cdot\frac{Q(\boldsymbol{z})}{Q(\hat{\boldsymbol{z}})}\right) \tag{4.20}$$

While $p(\boldsymbol{p},\boldsymbol{i},\hat{\boldsymbol{z}},\boldsymbol{x}\mid\boldsymbol{\theta},\boldsymbol{\phi})$ might seem slow to compute because it contains many factors (4.17) with different denominators $Z(x)$, one can share work by visiting the mentions x in their order $\boldsymbol{i}$. Most summands in $Z(x)$ were already included in $Z(x')$, where x' is the latest previous mention having the same attributes as x (e.g., same topic).

4.7.3 Resampling the phylogeny

It is easy to resample the phylogeny. For each x, we must choose a parent $x.p$ from among the possible parents p (having $p.i < x.i$ and $p.e.t = x.e.t$). Since the ordering $\boldsymbol{i}$ prevents cycles, the resulting phylogeny $\boldsymbol{p}$ is indeed a tree.

Given the topics $\boldsymbol{z}$, the ordering $\boldsymbol{i}$, and the observed names, we choose an $x.p$ value according to its posterior probability. This is proportional to $w(x.p,x) \stackrel{\text{def}}{=} p_{\boldsymbol{\phi}}(x.p \mid x)\cdot p_{\boldsymbol{\theta}}(x.n \mid x.p.n)$, independent of any other mention's choice of parent. The two factors here are given by (4.17) and (4.3) respectively. As in the previous section, the denominators $Z(x)$ in the $p(x.p \mid x)$ factors can be computed efficiently with shared work.

With the pragmatic model (§4.3.1), the parent choices are no longer independent; then the samples of $\boldsymbol{p}$ should be corrected by IMH as usual.

4.7.4 Initializing the sampler

The initial sampler state $(\boldsymbol{z}_0, \boldsymbol{p}_0, \boldsymbol{i}_0)$ is obtained as follows.

1. We fix topics $\boldsymbol{z}_0$ via collapsed Gibbs sampling [Griffiths and Steyvers, 2004]. The sampler is run for 1000 iterations, and the final sampler state is taken to be $\boldsymbol{z}_0$. This process treats all topics as exchangeable, including those associated with named entities.

2. Given the topic assignment $\boldsymbol{z}_0$, initialize $\boldsymbol{p}_0$ to the phylogeny rooted at $\diamondsuit$ that maximizes $\sum_x \log w(x.p, x)$. This is a maximum rooted directed spanning tree problem that can be solved in time $O(n^2)$ [Cheriton and Tarjan, 1976]. The weight $w(x.p, x)$ is defined as in §4.7.3—except that since we do not yet have an ordering $\boldsymbol{i}$, we do not restrict the possible values of $x.p$ to mentions p with $p.i < x.p.i$.

3. Given $\boldsymbol{p}_0$, sample an ordering $\boldsymbol{i}_0$

4.8 Parameter Estimation: Revisited

Evaluating the likelihood and its partial derivatives with respect to the parameters of the model requires marginalizing over our latent variables. As this marginalization is intractable, we resort to Monte Carlo EM procedure [Levine and Casella, 2001] which iterates the following two steps:

E-step: Collect samples by MCMC simulation as in §4.7, given current model parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$.

M-step: Improve $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ to increase

$$\mathcal{L} \stackrel{\text{def}}{=} \frac{1}{S} \sum_{s=1}^{S} \log p_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x}, \boldsymbol{p}_s, \boldsymbol{i}_s, \boldsymbol{z}_s) \tag{4.21}$$

We actually do MAP-EM, which augments (4.21) by adding the log-likelihoods of θ and ϕ under a Gaussian prior.

It is not necessary to locally maximize $\mathcal{L}$ at each M-step, merely to improve it if it is not already at a local maximum [Dempster et al., 1977]. We improve it by a single update: at the tth M-step, we update our parameters to $\Phi_t = (\boldsymbol{\theta}_t, \boldsymbol{\phi}_t)$

$$\Phi_t = \Phi_{t-1} + \varepsilon \Sigma_t \nabla_\Phi \mathcal{L}(\boldsymbol{x}, \Phi_{t-1}) \tag{4.22}$$

where ε is a fixed scaling term and Σ_t is an adaptive learning rate given by AdaGrad [Duchi et al., 2011].

We now describe how to compute the gradient $\nabla_\Phi \mathcal{L}$. The gradient with respect to the parent selection parameters $\boldsymbol{\phi}$ is

$$\sum \frac{1}{S} \left(\mathbf{f}(p, x) - \sum_{p'} p_{\boldsymbol{\phi}}(p' \mid x) \mathbf{f}(p', x) \right) \tag{4.23}$$

The outer summation ranges over all edges in the S samples. The other variables

in (4.23) are associated with the edge being summed over. That edge explains a mention x as a mutation of some parent p in the context of a particular sample $(\boldsymbol{p}_s, \boldsymbol{i}_s, \boldsymbol{z}_s)$. The possible parents p' range over $\diamondsuit$ and the mentions that precede x according to the ordering $\boldsymbol{i}_s$, while the features $\boldsymbol{f}$ and distribution $p_{\boldsymbol{\phi}}$ depend on the topics $\boldsymbol{z}_s$.

As for the mutation parameters, let $c_{p,x}$ be the fraction of samples in which p is the parent of x. This is the expected number of times that the string $p.n$ mutated into $x.n$. Given this weighted set of string pairs, let $c_{\hat{a},a,b}$ be the expected number of times that edit $\binom{a}{b}$ was chosen in context $\hat{a}$: this can be computed using dynamic programming to marginalize over the latent edit sequence that maps $p.n$ to $x.n$, for each (p, x).

The gradient of $\mathcal{L}$ with respect to $\boldsymbol{\theta}$ is

$$\sum_{\hat{a},a,b} c_{\hat{a},a,b}(\mathbf{f}(\hat{a},a,b) - \sum_{a',b'} p_{\boldsymbol{\theta}}(a',b' \mid \hat{a})\mathbf{f}(\hat{a},a',b')) \tag{4.24}$$

4.9 Consensus Clustering

From a single phylogeny $\boldsymbol{p}$, we deterministically obtain a clustering $\boldsymbol{e}$ by removing the root $\diamondsuit$. Each of the resulting connected components corresponds to a cluster of mentions. Our model gives a distribution over phylogenies $\boldsymbol{p}$ (given observations $\boldsymbol{x}$ and learned parameters Φ)—and thus gives a posterior distribution over clusterings

$\boldsymbol{e}$, which can be used to answer various queries.

A traditional query is to request a *single* clustering $\boldsymbol{e}$. We prefer the clustering $\boldsymbol{e}^*$ that minimizes Bayes risk (MBR) [Bickel and Doksum, 1977]:

$$\boldsymbol{e}^* = \operatorname*{argmin}_{\boldsymbol{e}'} \sum_{\boldsymbol{e}} L(\boldsymbol{e}', \boldsymbol{e}) p(\boldsymbol{e} \mid \boldsymbol{x}, \boldsymbol{\theta}, \boldsymbol{\phi}) \tag{4.25}$$

This minimizes our expected loss, where $L(\boldsymbol{e}', \boldsymbol{e})$ denotes the loss associated with picking $\boldsymbol{e}'$ when the true clustering is $\boldsymbol{e}$.

In practice, we again estimate the expectation by sampling $\boldsymbol{e}$ values.

The Rand index [Rand, 1971]—unlike our actual evaluation measure—is an *efficient* choice of loss function L for use with (4.25):

$$R(\boldsymbol{e}', \boldsymbol{e}) \stackrel{\text{def}}{=} \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}} = \frac{\text{TP} + \text{TN}}{\binom{N}{2}}$$

where the true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) use the clustering $\boldsymbol{e}$ to evaluate how well $\boldsymbol{e}'$ classifies the $\binom{N}{2}$ mention pairs as coreferent or not. More similar clusterings achieve larger R, with $R(\boldsymbol{e}', \boldsymbol{e}) = 1$ iff $\boldsymbol{e}' = \boldsymbol{e}$. In all cases, $0 \leq R(\boldsymbol{e}', \boldsymbol{e}) = R(\boldsymbol{e}, \boldsymbol{e}') \leq 1$.

The MBR decision rule for the (negated) Rand index is easily seen to be equivalent

to

$$\begin{aligned} \boldsymbol{e}^* &= \operatorname*{argmax}_{\boldsymbol{e}'} \mathbb{E}[\text{TP}] + \mathbb{E}[\text{TN}] \\ &= \operatorname*{argmax}_{\boldsymbol{e}'} \sum_{i,j:\, x_i \sim x_j} s_{ij} + \sum_{i,j:\, x_i \not\sim x_j} (1 - s_{ij}) \end{aligned} \tag{4.26}$$

where $\sim$ denotes coreference according to $\boldsymbol{e}'$. As explained above, the s_{ij} are coreference probabilities s_{ij} that can be estimated from a sample of clusterings $\boldsymbol{e}$.

This objective corresponds to min-max graph cut [Ding et al., 2001], an NP-hard problem with an approximate solution [Nie et al., 2010]. In our experiments, we run the clustering algorithm five times, initialized from samples chosen at random from the last 10% of the sampler run, and keep the clustering that achieved highest expected Rand score.

4.10 Experiments

In this section, we describe experiments on four different datasets. The first experiment on Wikipedia redirects evaluates the model without considering the token context §4.2.1. The remaining experiments evaluate the full model §4.6. For Wikipedia, Twitter, and ACE 2008, (§4.10.1,§4.10.2,§4.10.3) we report the standard B^3 metric [Bagga and Baldwin, 1998b]. For the political blog dataset (§4.10.4), the reference does not consist of entity annotations, and so we follow the evaluation procedure

of Yogatama et al. [2012].

4.10.1 Wikipedia Redirects

We begin by evaluating the simple generative model described in §4.2.1 and the associated *type-level* EM re-estimation procedure described in §4.5. The purpose of this initial evaluation is to model string variation independently from the context of individual mentions. We do so at various degrees of supervision, ranging from completely unsupervised to completely supervised §4.4. As our dataset, we use the English Wikipedia redirect corpus, described in §2.2. We reserve a fixed portion of the corpus for testing. For each test token, our system finds a set of coreferents. Our baseline is a "flat" model which does not stipulate any intermediary name forms: each name derives directly from the canonical name for the entity. We are directly evaluating the benefit of relating strings via successive mutations, compared to a baseline which assumes that all names are related directly to a common parent. The results are reported in 4.2, where we find a consistent benefit to our phylogenetic approach, at all levels of supervision.

4.10.2 Twitter

Data. We use a novel corpus of Twitter posts discussing the 2013 Grammy Award ceremony. This is a challenging corpus, featuring many instances of name variation.

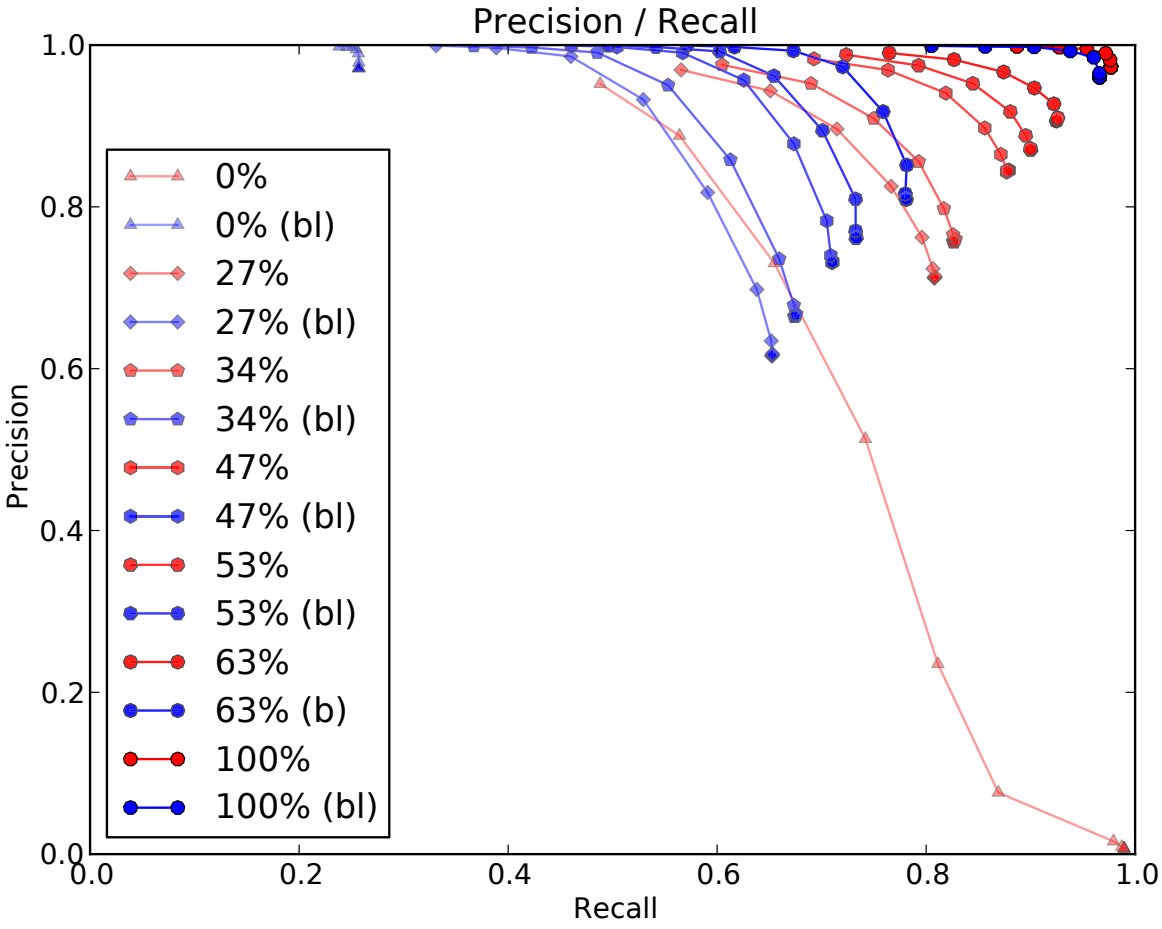

Figure 4.2: Precision and recall at different degrees of supervision for the proposed name phylogeny model and a baseline which does not stipulate any intermediary name forms. The proposed model consistently outperforms the baseline.

The dataset consists of five splits (by entity), the smallest of which is 604 mentions and the largest is 1374. We reserve the largest split for development purposes, and report our results on the remaining four.

Baselines. We use the discriminative entity clustering algorithm of Green et al. [2012] as our baseline; their approach was found to outperform another generative model which produced a flat clustering of mentions via a Dirichlet process mixture model. Their method uses Jaro-Winkler string similarity to match names, then clusters mentions with matching names (for disambiguation) by comparing their unigram context distributions using the Jenson-Shannon metric. We also compare to the EXACT-match baseline, which assigns all strings with the same name to the same entity.

Procedure. We run four test experiments in which one split is used to pick model hyperparameters and the remaining three are used for test. For the discriminative baseline, we tune the string match threshold, context threshold, and the weight of the context model prior (all via grid search). For our model, we tune *only* the fixed weight of the root feature, which determines the precision/recall trade-off (larger values of this feature result in more attachments to $\diamondsuit$ and hence more entities). We leave other hyperparameters fixed: 16 latent topics, and Gaussian priors $\mathcal{N}(0, 1)$ on all log-linear parameters. For PHYLO, the entity clustering is the result of (1) training the model using EM, (2) sampling from the posterior to obtain a distribution over clusterings, and (3) finding a consensus clustering. We use 20 iterations of EM with 100 samples per E-step for training, and use 1000 samples after training to estimate

	Precision	Recall	$\mathbf{B}^3$
EXACT-MATCH	99.6	53.7	69.8
Green et al. [2012]	92.1	69.8	79.3
PHYLO	85.3	91.4	88.7
PHYLO+TOPIC	92.8	90.8	91.8
PHYLO+TOPIC+MBR	92.9	90.9	91.9

Table 4.1: Results for the Twitter dataset, averaged over four data splits. Higher $\mathbf{B}^3$ scores are better.

the posterior. We report results using three variations of our model: PHYLO does not consider mention context (all mentions effectively have the same topic) and determines mention entities from a single sample of $\boldsymbol{p}$ (the last); PHYLO+TOPIC adds context (§4.7.2); PHYLO+TOPIC+MBR uses the full posterior and consensus clustering to pick the output clustering (§4.9). Our results are shown in Table 4.1.

4.10.3 Newswire

Data. We use the ACE 2008 dataset, which is described in detail in Green et al. [2012]. It was designed specifically to evaluate co-reference resolution systems, and includes entities with similar names, and entities with aliases (which our model does not explicitely account for), as well as variation in how the same names are spelled.

It is split into a development portion and a test portion. The baseline system took the first mention from each (gold) within-document coreference chain as the canonical mention, ignoring other mentions in the chain; we follow the same procedure in our experiments. That is, each within-document coreference chain is mapped to a single

mention as a preprocessing step.

Baselines & Procedure. We use the same baselines as in §4.10.2. On development data, modeling pragmatics as in §4.3.1 gave large improvements for organizations (8 points in F-measure), correcting the tendency to assume that short names like CIA were coincidental homonyms. Hence we allowed $\gamma > 0$ and tuned it on development data For these experiments, we used only a simplified version of the pragmatic model, approximating $w(p', x)$ as 1 or 0 according to whether $p'.n = x.n$. We also omitted the IMH step from §4.7.3. The other results we report do not use pragmatics at all, since we found that it gave only a slight improvement. Results are in Table 4.2.

		Precision	**Recall**	**B^3**
PER	EXACT-MATCH	98.0	81.2	88.8
	Green et al. [2012]	95.0	88.9	91.9
	PHYLO+TOPIC+MBR	97.2	88.6	92.7
ORG	EXACT-MATCH	98.2	78.3	87.1
	Green et al. [2012]	92.1	88.5	90.3
	PHYLO+TOPIC+MBR	95.5	80.9	87.6

Table 4.2: Results for the ACE dataset. Higher scores are better.

4.10.4 Blogs

Data. The CMU political blogs dataset consists of 3000 documents about U.S. politics [Yano et al., 2009]. Preprocessed as described in Yogatama et al. [2012], the data consists of 10647 entity mentions. Unlike our other datasets, mentions are not annotated with entities: the reference consists of a table of 126 entities, where each

row is the *canonical name* of one entity.

Baselines. We compare to the system results reported in Figure 2 of Yogatama et al. [2012]. This includes a baseline hierarchical clustering approach, the "EEA" name canonicalization system of Eisenstein et al. [2011], as well the model proposed by Yogatama et al. [2012]. Like the output of our model, the output of their hierarchical clustering baseline is a mention clustering, and therefore must be mapped to a table of canonical entity names to compare to the reference table.

Procedure & Results We tune our method as in previous experiments, on the initialization data used by Yogatama et al. [2012] which consists of a subset of 700 documents of the full dataset. The tuned model then produced a mention clustering on the full political blog corpus. As the mapping from clusters to a table is not fully detailed in Yogatama et al. [2012], we used a simple heuristic: the *most frequent* name in each cluster is taken as the canonical name, augmented by any titles from a predefined list appearing in any other name in the cluster. The resulting table is then evaluated against the reference, as described in Yogatama et al. [2012]. We achieved a response score of 0.17 and a reference score of 0.61. Though not state-of-the-art, this result is close to the score of the "EEA" system of Eisenstein et al. [2011], as reported in Figure 2 of Yogatama et al. [2012], which is specifically designed for the task of canonicalization.

4.10.5 Discussion

On the Twitter dataset, we obtained a 12.6-point F1 improvement over the baseline. To understand our model's behavior, we looked at the sampled phylogenetic trees on development data. One reason our model does well in this noisy domain is that it is able to relate seemingly dissimilar names via successive steps. For instance, our model learned to relate many variations of LL Cool J:

Cool James	LLCoJ	El-El Cool John
LL	LL COOL JAMES	LLCOOLJ

In the sample we inspected, these mentions were also assigned the same topic, further boosting the probability of the configuration.

The ACE dataset, consisting of editorialized newswire, naturally contains less name variation than Twitter data. Nonetheless, we find that the variation that does appear is often properly handled by our model. For instance, we see several instances of variation due to transliteration that were all correctly grouped together, such as Megawati Soekarnoputri and Megawati Sukarnoputri. The pragmatic model was also effective in grouping common acronyms into the same entity.

We found that multiple samples tend to give different phylogenies (so the sampler is mobile), but essentially the same clustering into entities (which is why consensus clustering did not improve much over simply using the last sample). Random restarts of EM might create more variety by choosing different locally optimal parameter settings. It may also be beneficial to explore other sampling techniques [Bouchard-Côté, 2014].

We suspect that our greedy EM optimization procedure is the culprit for the peaked posterior, and a possible improvement would be to run multiple sampler chains from different initializations of the parameters; we leave this as future work however.

Our method assembles observed names into an evolutionary tree. However, the true tree must include many names that fall outside our small observed corpora, so our model would be a more appropriate fit for a far larger corpus. Larger corpora also offer stronger signals that might enable our Monte Carlo methods to mix faster and detect regularities more accurately. Nonetheless, in this paper we have shown that these unsupervised techniques can achieve good results even on small datasets.

A common error of our system is to connect mentions that share long substrings, such as different PERSONs who share a last name, or different ORGANIZATIONs that contain University of. A more powerful name mutation than the one we use here would recognize entire words, for example inserting a common title or replacing a first name with its common nickname.

4.11 Related Work

Several previous papers have also considered learning transducers or other models of word pairs when the pairing between inputs and outputs is not given. Most commonly, one observes parallel or comparable corpora in two languages, and must reconstruct a matching from one language's words to the other's before training on the resulting

pairs [Schafer, 2006, Klementiev and Roth, 2006, Haghighi et al., 2008, Snyder et al., 2010, Sajjad et al., 2011]. Haghighi et al. [2008] learn a joint generative model of input features and output features, although it does not take the form of a transducer.

Prior work on learning transducers without training pairs has relied on a given partition of the observed strings into unaligned "inputs" and "outputs." For instance, Irvine et al. [2010] use Wikipedia to construct supervision for transliteration models between many languages.

This work differs in that we assume no such partitioning, and our latent variable is not a bipartite matching between the input and output strings. Instead, given a single corpus of unaligned strings, our latent variable is a tree relating different string types. Any string may be generated as the output of one edge and subsequently serve as the input to other edges.

Hall and Klein [2010] extend this setting to more than two languages, where the phylogenetic tree is known. A given lexeme (abstract word) can be realized in each language by at most one word (string type), derived from the parent language's realization of the same lexeme. The system must match words that share an underlying lexeme (i.e., cognates), creating a matching of each language's vocabulary to its parent language's vocabulary. A further challenge is that the parent words are *unobserved* ancestral forms.

Hall and Klein [2010] use the term "cognate group" instead of "lexeme." In the more scalable PARSIM model of [Hall and Klein, 2011], it is a semantic slot ("gloss")

that is realized once per language, and this slot is observed for each word type. For a given slot, the child language may use the same lexeme as the parent (mutation) or create a new one (innovation).

Similarly, Dreyer and Eisner [2011b] organize words into morphological paradigms of a given structure. Again words with the same underlying lexeme (i.e., morphemes) must be identified. A lexeme can be realized in each grammatical inflection (such as "first person plural present") by exactly one word type, related to other inflected forms of the same lexeme, which as above may be unobserved. Their inference setting is closer to ours because the input is an unorganized collection of words—input words are not tagged with their grammatical inflections. This contrasts with the usual multilingual setting where each word is tagged with its true language.

In one way, our problem differs significantly from the above problems. We are interested in random variation that may occur *within* a language as well as across languages. A person name may have *unboundedly* many different variants. This is unlike the above problems, in which a lexeme has at most K realizations, where K is the (small) number of languages or inflections. In the above problems, one learns a set of $O(K)$ or $O(K^2)$ specialized transducers that relate Latin to Italian, singular to plural, etc. We instead use one global mutation model that applies to all names.

We cannot assign the observed strings to positions in an existing structure that is shared across all lexemes, such as a given phylogenetic tree whose K nodes represent languages, or a given inflectional grid whose K cells represent grammatical inflections.

Rather, we must organize them into a idiosyncratic phylogenetic tree whose nodes are the string types or tokens themselves.

Names and words are not the only non-biological objects that are copied with mutation. Documents, database records, bibliographic entries, code, and images can evolve in the same way. Reconstructing these relationships has been considered by a number of papers on authorship attribution, near-duplicate detection, deduplication, record linkage, and plagiarism detection. A few such papers reconstruct a phylogeny, as in the case of chain letters [Bennett et al., 2003], malware [Karim et al., 2005], or images [Dias et al., 2012]. In fact, the last of these uses the same minimum spanning tree method that we apply in §4.4.4. However, these papers do not *train* a similarity measure as we do. To our knowledge, these two techniques have not been combined outside biology.

In molecular evolutionary analysis, phylogenetic techniques *have* often been combined with estimation of some parametric model of mutation [Tamura et al., 2011]. However, names mutate differently from biological sequences, and our mutation model for names (§4.3) reflects that. We also posit a specific process (§4.2.1) that generates the name phylogeny.

One view of our phylogenetic model comes from the literature on random graphs (e.g., for modeling social networks or the link structure of the web). In a *preferential attachment* model, a graph's vertices are added one by one, and each vertex selects some previous vertices as its neighbors. Our phylogeny is a *preferential attachment*

tree, a random *directed* graph in which each vertex selects a *single* previous vertex as its parent. Specifically, it is a *random recursive tree*[Smythe and Mahmoud, 1995] whose vertices are the tokens. This is not the tree shown in Figure 4.1, whose vertices are types rather than tokens. To this simple random topology we have added a random labeling process with mutation.

Cross-document coreference resolution (CDCR) was first introduced by Bagga and Baldwin [1998a]. Most approaches since then are based on the intuitions that coreferent names tend to have "similar" spellings and tend to appear in "similar" contexts. The distinguishing feature of our system is that both notions of similarity are learned together without supervision. We adopt a "phylogenetic" generative model of coreference. The basic insight is that coreference is created when an author thinks of an entity that was mentioned earlier in a similar context, and mentions it again in a similar way. The author may alter the name mention string when copying it, but both names refer to the same entity. Either name may later be copied further, leading to an evolutionary tree of mentions—a phylogeny. Phylogenetic models are new to information extraction. In computational historical linguistics, Bouchard-Côté et al. [2013] have also modeled the mutation of strings along the edges of a phylogeny; but for them the phylogeny is observed and most mentions are not, while we observe the mentions only.

Name similarity is also an important component of *within*-document coreference resolution, and efforts in that area bear resemblance to our approach. Haghighi and

Klein [2010] describe an "entity-centered" model where a distance-dependent Chinese restaurant process is used to pick previous coreferent mentions *within* a document. Similarly, Durrett and Klein [2013] learn a mention similarity model based on labeled data. Our *cross*-document setting has no observed mention ordering and no observed entities: we must sum over all possibilities, a challenging inference problem.

Chapter 5

Conclusion

Generative models, such as hidden Markov models, are commonly used for problems where either no annotated data is available, or where only partial supervision exists. Unfortunately, it is difficult to incorporate expressive features in these models while retaining tractable inference algorithms. In this thesis, we have proposed joint models that make minimal assumptions about the underlying generative process, and incorporate rich character-level features.

Our primary technical contributions are as follows:

In §2, we consider character-level sequence models. We propose a stacked LSTM architecture for jointly modeling multiple string types in a single network, with parameters shared between all types. We explore different approaches for incorporating entity attributes, yielding name models which are sensitive to various properties such as nationality and gender. We also experimented with discriminative training objec-

tives as an alternative to log-likelihood training, and found that this led to further improvements in classification performance.

In §3, we propose a hidden non-Markov model for inducing latent state sequences and labeled segmentation problems. We describe two variants of the model: (1) a fully nonparametric variant, where Bayesian sequence memoizers are used at both the word-level and character-levels and (2) a hybrid model in which a more expressive neural model is used at the character level. We propose a general framework for inference in such models, based on particle MCMC, and apply this framework to unsupervised part-of-speech induction with dictionaries, and to named-entity recognition with gazetteers.

In §4, we proposed an evolutionary phylogenetic process to discover relations between strings in an unorganized collection. Our primary contribution consists of new modeling ideas, and associated inference techniques, for the problem of cross-document coreference resolution. We have described how writers systematically plunder and then systematically modify the work of past writers. Inference under such models could also play a role in tracking evolving memes and social influence, not merely in establishing strict coreference. Our perspective descends from noisy channel modeling. Working with highly noisy corpora such as Twitter requires distinguishing the underlying signal from the noise (e.g., identifying entities that are referred to inconsistently).

5.1 Extensions

There are many opportunities for extensions of the proposed methods. We mention some here.

- In §3, we used a convenient factorization of the hidden non-Markov model into independent transition and emission factors, which correspond to models over (latent) words and observed characters. However, this is an unrealistic assumption, since the context in which a word occurs may influence its spelling. A simple example of a contextual dependency for NER is where a full name is used at the beginning of a news article. Another example is where a nickname is used in an informal context, and a full name in a formal context. It would be desirable for the emission distribution to incorporate contextual information. One way to accomplish this is by directly parametrizing the joint distribution over labeled characters. This may be understoodd as modeling sequences of *tuples*, consisting of a label and a character. The probability of emitting each tuple is dependent on all previous tuples. We may then use an RNN to score paths of such tuples. This model poses the same inference challenges as the one proposed in §3; however, particle Gibbs again provides an efficient solution for approximate inference.

- We have advocated using an unstructured approach—character-level sequence models—to model name structure. Of course, names often *do* exhibit a consis-

tent structure; for instance, English person names typically have a first, middle, and last name. Our proposed models *implicitely* capture this structure in the continuous dynamics of the RNN, which avoids encoding any fixed assumptions into the model. However, in unsupervised or low-resource settings, it may be helpful to imbue the model with more latent variables, enabling it to capture higher-order phenomena such as re-orderings or word-level variation more efficiently [Elsner et al., 2009, Charniak, 2001]. Grammar induction techniques, usually applied to learning syntax, could be adapted to avoid manually specifying this structure [Cohn et al., 2010, Gormley and Eisner, 2013].

- The transducer described in §4.3 could be parametrized using RNNs in several different ways. One approach involves treating the problem as a sequence-to-sequence RNN [Sutskever et al., 2014], in which the input string is encoded into a continuous vector-space, which is then used to render the output string. This approach is appealing as it does not rely on a fixed alignment between the input and output strings, and may provide a better account for re-orderings. Another option is to use a transducer model where the probability of different edits are parametrized using RNNs.

- In §3.3, we propose an inference method based on sequential Monte Carlo. The proposal q to sample the state at time t used all previous and current information. However, it may also be beneficial to incorporate *future* information into the

proposal, in order to minimize particle degeneracy. One promising approach for doing so is to use a recurrent neural network as an adaptive proposal [Gu et al., 2015]. The proposal may be adapted (optimized) by minimizing the inclusive KL divergence between the target posterior $p(\boldsymbol{y} \mid \boldsymbol{x})$ and the proposal q.

5.2 Outlook

At the time of writing this thesis, there has been a resurgence in interest in neural methods. A key strength of neural networks is their ability for feature learning. One of the first successful attempts at natural language processing "from scratch" using neural networks was Collobert et al. [2011]. Given enough supervised data, neural networks are able to learn expressive continuous representations of natural language inputs. Seminal work in machine translation has since shown the ability of recurrent neural networks (*cf.* §2, §3) to encode entire input sentences into a fixed dimensional vectors, and to produce translations based on those encodings competitive with highly engineered approaches [Sutskever et al., 2014]. The merit of representation learning in fully supervised settings is well-established at this point. However, there remain many interesting research directions in settings where there are no resources or more limited resources, and therefore require unsupervised or semi-supervised learning. A weakness of neural networks is that they are data inefficient, requiring many examples of inputs and desired outputs. For instance, a recent attempt at using recurrent neural networks

for constituency parsing relied on an entirely separate generative model to produce additional (noisy) supervision for the neural model from unlabeled data [Vinyals et al., 2015]. Our view is that directly enriching generative models with neural factors, and performing joint inference in the resulting model, may ultimately yield superior results in various structured prediction tasks. In §3, we take some modest first steps in this direction, and we discuss some extensions in §5.1 that we believe are quite promising.

Bibliography

D. B. Rubin A. P. Dempster, N. M. Laird. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977. ISSN 00359246. URL `http://www.jstor.org/stable/2984875`.

David J. Aldous. *École d'Été de Probabilités de Saint-Flour XIII — 1983*, chapter Exchangeability and related topics, pages 1–198. Springer Berlin Heidelberg, Berlin, Heidelberg, 1985. ISBN 978-3-540-39316-0. doi: 10.1007/BFb0099421. URL `http://dx.doi.org/10.1007/BFb0099421`.

Waleed Ammar, Chris Dyer, and Noah A Smith. Conditional random field autoencoders for unsupervised structured prediction. In *Advances in Neural Information Processing Systems*, pages 3311–3319, 2014.

Nicholas Andrews and Jason Eisner. Transformation process priors. In *NIPS 2011 Workshop on Bayesian Nonparametrics: Hope or Hype?*, Sierra Nevada, Spain,

December 2011. URL http://cs.jhu.edu/~jason/papers/#nipsw11-transf. Extended abstract (3 pages).

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

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. *Dbpedia: A nucleus for a web of open data*. Springer, 2007.

Amit Bagga and Breck Baldwin. Entity-based cross-document coreferencing using the vector space model. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 1*, ACL '98, pages 79–85, Stroudsburg, PA, USA, 1998a. Association for Computational Linguistics. doi: 10.3115/980845.980859.

Amit Bagga and Breck Baldwin. Algorithms for scoring coreference chains. In *In The First International Conference on Language Resources and Evaluation Workshop on Linguistics Coreference*, pages 563–566, 1998b.

Alex Baron and Marjorie Freedman. Who is who and what is what: Experiments in cross-document co-reference. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 274–283, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1613715.1613754.

BIBLIOGRAPHY

Regina Barzilay and Lillian Lee. Learning to paraphrase: an unsupervised approach using multiple-sequence alignment. In *Proc. of NAACL-HLT*, pages 16–23, Stroudsburg, PA, USA, 2003. doi: 10.3115/1073445.1073448.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on*, 5 (2):157–166, 1994.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, March 2003. ISSN 1532-4435. URL `http://dl.acm.org/citation.cfm?id=944919.944966`.

Yoshua Bengio, Holger Schwenk, Jean-Sébastien Senécal, Fréderic Morin, and Jean-Luc Gauvain. *Innovations in Machine Learning: Theory and Applications*, chapter Neural Probabilistic Language Models, pages 137–186. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. ISBN 978-3-540-33486-6. doi: 10.1007/3-540-33486-6_6. URL `http://dx.doi.org/10.1007/3-540-33486-6_6`.

C. H. Bennett, M. Li, , and B. Ma. Chain letters and evolutionary histories. *Scientific American*, 288(3):76–81, June 2003. More mathematical version available at `http://www.cs.uwaterloo.ca/~mli/chain.html`.

Taylor Berg-Kirkpatrick, Alexandre Bouchard-Côté, John DeNero, and Dan Klein. Painless unsupervised learning with features. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association*

for Computational Linguistics, HLT '10, pages 582–590, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. ISBN 1-932432-65-5. URL http://dl.acm.org/citation.cfm?id=1857999.1858082.

Indrajit Bhattacharya and Lise Getoor. *A Latent Dirichlet Model for Unsupervised Entity Resolution*, chapter 5, pages 47–58. doi: 10.1137/1.9781611972764.5. URL http://epubs.siam.org/doi/abs/10.1137/1.9781611972764.5.

Peter J. Bickel and Kjell A. Doksum. *Mathematical Statistics : Basic Ideas and Selected Topics*. Holden-Day, Inc., 1977.

D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003. URL /home/rayder441/Documents/PaperArchive/3-993-blei.pdf.

David M. Blei and Peter I. Frazier. Distance dependent chinese restaurant processes. *J. Mach. Learn. Res.*, 12:2461–2488, November 2011. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=1953048.2078184.

Alexandre Bouchard-Côté. Sequential Monte Carlo (SMC) for Bayesian phylogenetics. *Bayesian phylogenetics: methods, algorithms, and applications*, 2014.

Alexandre Bouchard-Côté, David Hall, Thomas L. Griffiths, and Dan Klein. Automated reconstruction of ancient languages using probabilistic models of sound change. *Proceedings of the National Academy of Sciences*, 2013.

BIBLIOGRAPHY

Eugene Charniak. Unsupervised learning of name structure from coreference data. In *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics on Language Technologies*, NAACL '01, pages 1–7, Stroudsburg, PA, USA, 2001. Association for Computational Linguistics. doi: 10.3115/1073336.1073343. URL `http://dx.doi.org/10.3115/1073336.1073343`.

David Cheriton and Robert Endre Tarjan. Finding minimum spanning trees. *SIAM Journal on Computing*, 5(4):724–742, 1976.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Christos Christodoulopoulos, Sharon Goldwater, and Mark Steedman. Two decades of unsupervised pos induction: How far have we come? In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 575–584. Association for Computational Linguistics, 2010.

Trevor Cohn, Phil Blunsom, and Sharon Goldwater. Inducing tree-substitution grammars. *J. Mach. Learn. Res.*, 11:3053–3096, December 2010. ISSN 1532-4435. URL `http://dl.acm.org/citation.cfm?id=1756006.1953031`.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu,

and Pavel Kuksa. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537, November 2011. ISSN 1532-4435. URL `http://dl.acm.org/citation.cfm?id=1953048.2078186`.

Aron Culotta, Michael Wick, Robert Hall, Matthew Marzilli, and Andrew McCallum. Canonicalization of database records using adaptive similarity measures. In *Proc. of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '07, pages 201–209, 2007. ISBN 978-1-59593-609-7. doi: 10.1145/1281192.1281217.

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

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977. ISSN 00359246. doi: 10.2307/2984875. URL `http://web.mit.edu/6.435/www/Dempster77.pdf`.

Z. Dias, A. Rocha, and S. Goldenstein. Image phylogeny by minimal spanning trees. *IEEE Trans. on Information Forensics and Security*, 7(2):774–788, April 2012.

C.H.Q. Ding, Xiaofeng He, Hongyuan Zha, Ming Gu, and H.D. Simon. A min-max cut algorithm for graph partitioning and data clustering. In *Data Mining, 2001.*

ICDM 2001, Proceedings IEEE International Conference on, pages 107–114, 2001. doi: 10.1109/ICDM.2001.989507.

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

Markus Dreyer and Jason Eisner. Discovering morphological paradigms from plain text using a dirichlet process mixture model. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 616–627, Stroudsburg, PA, USA, 2011a. Association for Computational Linguistics. ISBN 978-1-937284-11-4. URL `http://dl.acm.org/citation.cfm?id=2145432.2145504`.

Markus Dreyer and Jason Eisner. Discovering morphological paradigms from plain text using a Dirichlet process mixture model. In *Proc. of EMNLP*, pages 616–627, 2011b. Supplementary material (9 pages) also available.

Markus Dreyer, Jason Smith, and Jason Eisner. Latent-variable modeling of string transductions with finite-state methods. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 1080–1089, Honolulu, Hawaii, October 2008a. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/D08-1113`.

Markus Dreyer, Jason Smith, and Jason Eisner. Latent-variable modeling of string transductions with finite-state methods. In *Proc. of EMNLP*, pages 1080–1089,

Honolulu, Hawaii, October 2008b. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D08-1113.

Gregory Dubbin and Phil Blunsom. Unsupervised bayesian part of speech inference with particle gibbs. In *Machine Learning and Knowledge Discovery in Databases*, pages 760–773. Springer, 2012.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, July 2011. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=1953048.2021068.

Greg Durrett and Dan Klein. Easy victories and uphill battles in coreference resolution. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1971–1982. Association for Computational Linguistics, 2013. URL http://aclweb.org/anthology/D13-1203.

Jacob Eisenstein, Tae Yano, William W. Cohen, Noah A. Smith, and Eric P. Xing. Structured databases of named entities from Bayesian nonparametrics. In *Proceedings of the First Workshop on Unsupervised Learning in NLP*, EMNLP '11, pages 2–12, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. ISBN 978-1-937284-13-8. URL http://dl.acm.org/citation.cfm?id=2140458.2140460.

Jason Eisner. Transformational priors over grammars. In *Proceedings of the Confer-*

ence on Empirical Methods in Natural Language Processing (EMNLP), Philadelphia, July 2002. URL `http://cs.jhu.edu/~jason/papers/#emnlp02`.

Micha Elsner, Eugene Charniak, and Mark Johnson. Structured generative models for unsupervised named-entity clustering. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 164–172. Association for Computational Linguistics, 2009.

T. Finin, Z. Syed, J. Mayfield, P. McNamee, and C. Piatko. Using Wikitology for cross-document entity coreference resolution. In *AAAI Spring Symposium on Learning by Reading and Learning to Read*, 2009. URL `/home/rayder441/Documents/PaperArchive/finin-coref09.pdf`.

Jurgen Van Gael, Yunus Saatci, Yee Whye Teh, and Zoubin Ghahramani. Beam sampling for the infinite hidden markov model. In *Proceedings of the 25th international conference on Machine learning*, pages 1088–1095. ACM, 2008.

Dan Garrette and Jason Baldridge. Learning a part-of-speech tagger from two hours of annotation. In *HLT-NAACL*, pages 138–147. Citeseer, 2013.

Jan Gasthaus and Yee Whye Teh. Improvements to the sequence memoizer. In *NIPS*, pages 685–693, 2010.

F.A. Gers and J. Schmidhuber. Lstm recurrent networks learn simple context-free

and context-sensitive languages. *Neural Networks, IEEE Transactions on*, 12(6): 1333–1340, Nov 2001. ISSN 1045-9227. doi: 10.1109/72.963769.

Walter R Gilks. *Markov chain monte carlo*. Wiley Online Library, 2005.

Sharon Goldwater and Tom Griffiths. A fully bayesian approach to unsupervised part-of-speech tagging. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 744–751, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P07-1094`.

Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. Contextual dependencies in unsupervised word segmentation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, ACL-44, pages 673–680, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics. doi: 10.3115/1220175.1220260. URL `http://dx.doi.org/10.3115/1220175.1220260`.

Javier Gonzalez-Dominguez, Ignacio Lopez-Moreno, Hasim Sak, Joaquin Gonzalez-Rodriguez, and Pedro J Moreno. Automatic language identification using long short-term memory recurrent neural networks. 2014.

Noah Goodman, Vikash Mansinghka, Daniel M Roy, Keith Bonawitz, and Joshua B Tenenbaum. Church: a language for generative models. *arXiv preprint arXiv:1206.3255*, 2012.

BIBLIOGRAPHY

Matthew R. Gormley and Jason Eisner. Nonconvex global optimization for latent-variable models. In *Proceedings of ACL*, August 2013.

Spence Green, Nicholas Andrews, Matthew R. Gormley, Mark Dredze, and Christopher D. Manning. Entity clustering across languages. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12, pages 60–69, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics. ISBN 978-1-937284-20-6. URL `http://dl.acm.org/citation.cfm?id=2382029.2382039`.

Thomas L Griffiths and Mark Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences of the United States of America*, 101(Suppl 1): 5228–5235, 2004.

Shixiang Gu, Zoubin Ghahramani, and Richard E Turner. Neural adaptive sequential monte carlo. In C. Cortes, N.D. Lawrence, D.D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2611–2619. Curran Associates, Inc., 2015. URL `http://papers.nips.cc/paper/5961-neural-adaptive-sequential-monte-carlo.pdf`.

Dan Gusfield. *Algorithms on Strings, Trees, and Sequences—Computer Science and Computational Biology*. Cambridge University Press, 1997. ISBN 0-521-58519-8.

Aria Haghighi and Dan Klein. Coreference resolution in a modular, entity-centered

model. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 385–393, Los Angeles, California, June 2010. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/N10-1061.

Aria Haghighi, Percy Liang, Taylor Berg-Kirkpatrick, and Dan Klein. Learning bilingual lexicons from monolingual corpora. In *Proc. of ACL-08: HLT*, pages 771–779, 2008. URL http://www.aclweb.org/anthology/P/P08/P08-1088.

David Hall and Dan Klein. Finding cognates using phylogenies. In *Association for Computational Linguistics (ACL)*, 2010.

David Hall and Dan Klein. Large-scale cognate recovery. In *Proc. of EMNLP*, Edinburgh, Scotland, July 2011.

Rob Hall, Charles Sutton, and Andrew McCallum. Unsupervised deduplication using cross-field dependencies. In *Proc. of the ACM SIGKDD International Conference On Knowledge Discovery and Data Mining*, KDD '08, pages 310–317, 2008. ISBN 978-1-60558-193-4. doi: http://doi.acm.org/10.1145/1401890.1401931. URL http://doi.acm.org/10.1145/1401890.1401931.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

BIBLIOGRAPHY

S Hochreiter and J Schmidhuber. Long short-term memory. *Neural Computation*, 9 (8):1735–1780, Nov 1997a. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735.

Sepp Hochreiter and Jürgen Schmidhuber. Lstm can solve hard long time lag problems. *Advances in neural information processing systems*, pages 473–479, 1997b.

Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jürgen Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

Fred M. Hoppe. Pólya-like urns and the ewens' sampling formula. *Journal of Mathematical Biology*, 20(1):91–94, 1984. ISSN 1432-1416. doi: 10.1007/BF00275863. URL `http://dx.doi.org/10.1007/BF00275863`.

Ann Irvine, Chris Callison-Burch, and Alexandre Klementiev. Transliterating from all languages. In *Proceedings of The Ninth Biennial Conference of the Association for Machine Translation in the Americas*, Denver, Colorado, 2010. URL `http://cis.upenn.edu/~ccb/publications/transliterating-from-all-languages.pdf`.

Hemant Ishwaran and Lancelot F James. Gibbs sampling methods for stick-breaking priors. *Journal of the American Statistical Association*, 96(453), 2001.

Md. Enamul. Karim, Andrew Walenstein, Arun Lakhotia, and Laxmi Parida. Malware phylogeny generation using permutations of code. *Journal in Computer Virology*, 1(1–2):13–23, 2005.

BIBLIOGRAPHY

Junichi Kazama and Kentaro Torisawa. Exploiting wikipedia as external knowledge for named entity recognition. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 698–707, 2007.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL `http://arxiv.org/abs/1412.6980`.

Alexandre Klementiev and Dan Roth. Weakly supervised named entity transliteration and discovery from multilingual comparable corpora. In *Proc. of COLING-ACL*, pages 817–824, 2006. URL `http://www.aclweb.org/anthology/P/P06/P06-1103`.

K. Knight and J. Graehl. Machine transliteration. *Computational Linguistics*, 24: 599–612, 1998.

Terry Koo, Amir Globerson, Xavier Carreras, and Michael Collins. Structured prediction models via the matrix-tree theorem. In *Proc. of EMNLP-CoNLL*, pages 141–150, 2007. URL `http://www.aclweb.org/anthology/D/D07/D07-1015`.

Zornitsa Kozareva. Bootstrapping named entity recognition with automatically generated gazetteer lists. In *Proceedings of the eleventh conference of the European chapter of the association for computational linguistics: student research workshop*, pages 15–21. Association for Computational Linguistics, 2006.

BIBLIOGRAPHY

Rémi Lebret and Ronan Collobert. Word embeddings through hellinger pca. *EACL 2014*, page 482, 2014.

Heeyoung Lee, Marta Recasens, Angel Chang, Mihai Surdeanu, and Dan Jurafsky. Joint entity and event coreference resolution across documents. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, 2012.

Richard A. Levine and George Casella. Implementations of the Monte Carlo EM Algorithm. *Journal of Computational and Graphical Statistics*, 10(3):422–439, 2001. ISSN 10618600. doi: 10.2307/1391097. URL `http://dx.doi.org/10.2307/1391097`.

David D Lewis. Naive (bayes) at forty: The independence assumption in information retrieval. In *Machine learning: ECML-98*, pages 4–15. Springer, 1998.

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

Fredrik Lindsten, Thomas Schön, and Michael I Jordan. Ancestor sampling for particle gibbs. In *Advances in Neural Information Processing Systems*, pages 2591–2599, 2012.

BIBLIOGRAPHY

Jun S Liu. The collapsed gibbs sampler in bayesian computations with applications to a gene regulation problem. *Journal of the American Statistical Association*, 89 (427):958–966, 1994.

Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. Text classification using string kernels. *J. Mach. Learn. Res.*, 2:419–444, March 2002. ISSN 1532-4435. doi: 10.1162/153244302760200687. URL http://dx.doi.org/10.1162/153244302760200687.

Marco Lui and Timothy Baldwin. langid. py: An off-the-shelf language identification tool. In *Proceedings of the ACL 2012 system demonstrations*, pages 25–30. Association for Computational Linguistics, 2012.

Larry M. Manevitz and Malik Yousef. One-class svms for document classification. *J. Mach. Learn. Res.*, 2:139–154, March 2002. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=944790.944808.

Andrew McCallum, Chris Pal, Greg Druck, and Xuerui Wang. Multi-conditional learning: Generative/discriminative training for clustering and classification. In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 1*, AAAI'06, pages 433–439. AAAI Press, 2006. ISBN 978-1-57735-281-5. URL http://dl.acm.org/citation.cfm?id=1597538.1597608.

Daichi Mochihashi, Takeshi Yamada, and Naonori Ueda. Bayesian unsupervised word segmentation with nested pitman-yor language modeling. In *Proceedings of*

the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1, pages 100–108. Association for Computational Linguistics, 2009.

Karthika Mohan, Judea Pearl, and Jin Tian. Graphical models for inference with missing data. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 1277–1285. Curran Associates, Inc., 2013. URL `http://papers.nips.cc/paper/4899-graphical-models-for-inference-with-missing-data.pdf`.

David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007.

Tetsuji Nakagawa, Kentaro Inui, and Sadao Kurohashi. Dependency tree-based sentiment classification using crfs with hidden variables. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 786–794, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. ISBN 1-932432-65-5. URL `http://dl.acm.org/citation.cfm?id=1857999.1858119`.

Feiping Nie, Chris H. Q. Ding, Dijun Luo, and Heng Huang. Improved minmax cut graph clustering with nonnegative relaxation. In José L. Balcázar, Francesco Bonchi, Aristides Gionis, and Michèle Sebag, editors, *ECML/PKDD (2)*, volume

6322 of *Lecture Notes in Computer Science*, pages 451–466. Springer, 2010. ISBN 978-3-642-15882-7.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of The 30th International Conference on Machine Learning*, pages 1310–1318, 2013.

Mihael Perman, Jim Pitman, and Marc Yor. Size-biased sampling of poisson point processes and excursions. *Probability Theory and Related Fields*, 92(1):21–39, 1992.

David MW Powers. Applications and explanations of zipf's law. In *Proceedings of the joint conferences on new methods in language processing and computational natural language learning*, pages 151–160. Association for Computational Linguistics, 1998.

Daniel Ramage, David Hall, Ramesh Nallapati, and Christopher D. Manning. Labeled lda: A supervised topic model for credit attribution in multi-labeled corpora. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1*, EMNLP '09, pages 248–256, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics. ISBN 978-1-932432-59-6. URL `http://dl.acm.org/citation.cfm?id=1699510.1699543`.

William M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971. doi:

10.1080/01621459.1971.10482356. URL http://www.tandfonline.com/doi/abs/10.1080/01621459.1971.10482356.

Delip Rao, Paul McNamee, and Mark Dredze. Streaming cross document entity coreference resolution. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, COLING '10, pages 1050–1058, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1944566.1944687.

Jason DM Rennie and Ryan Rifkin. Improving multiclass text classification with the support vector machine. 2001.

Eric Sven Ristad and Peter N. Yianilos. Learning string edit distance. Technical Report CS-TR-532-96, Princeton University, Department of Computer Science, 1996.

Eric Sven Ristad and Peter N. Yianilos. Learning string edit distance. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 20(5):522–532, May 1998.

Hassan Sajjad, Alexander Fraser, and Helmut Schmid. An algorithm for unsupervised transliteration mining with an application to word alignment. In *Proc. of ACL*, pages 430–439, 2011. ISBN 978-1-932432-87-9. URL http://dl.acm.org/citation.cfm?id=2002472.2002527.

BIBLIOGRAPHY

Charles Schafer. *Translation Discovery Using Diverse Smilarity Measures*. PhD thesis, Johns Hopkins University, 2006.

Sameer Singh, Amarnag Subramanya, Fernando Pereira, and Andrew McCallum. Large-scale cross-document coreference using distributed inference and hierarchical models. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 793–803, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P11-1080`.

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

David A. Smith and Noah A. Smith. Probabilistic models of nonprojective dependency trees. In *Proc. of EMNLP-CoNLL*, pages 132–140, 2007.

R. T. Smythe and H. M. Mahmoud. A survey of recursive trees. *Theory of Probability and Mathematical Statistics*, 51(1–27), 1995.

Benjamin Snyder, Regina Barzilay, and Kevin Knight. A statistical model for lost language decipherment. In *Proc. of ACL*, pages 1048–1057, 2010. URL `http://www.aclweb.org/anthology/P10-1107`.

BIBLIOGRAPHY

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

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

J.A.K. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9(3):293–300, 1999. ISSN 1370-4621. doi: 10.1023/A:1018628609742. URL `http://dx.doi.org/10.1023/A%3A1018628609742`.

Koichiro Tamura, Daniel Peterson, Nicholas Peterson, Glen Stecher, Masatoshi Nei, and Sudhir Kumar. Mega5: Molecular evolutionary genetics analysis using maximum likelihood, evolutionary distance, and maximum parsimony methods. *Molecular Biology and Evolution*, 28(10):2731–2739, 2011. doi: 10.1093/molbev/msr121. URL `http://mbe.oxfordjournals.org/content/28/10/2731.abstract`.

Duyu Tang, Bing Qin, and Ting Liu. Document modeling with gated recurrent

neural network for sentiment classification. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1422–1432, Lisbon, Portugal, September 2015. Association for Computational Linguistics. URL `http://aclweb.org/anthology/D15-1167`.

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

Luke Tierney. Markov Chains for Exploring Posterior Distributions. *The Annals of Statistics*, 22(4):1701–1728, 1994. ISSN 00905364.

Erik F Tjong Kim Sang and Fien De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 142–147. Association for Computational Linguistics, 2003.

Antonio Toral and Rafael Munoz. A proposal to automatically build and maintain gazetteers for named entity recognition by using wikipedia. In *Proceedings of EACL*, pages 56–61, 2006.

Pedro A Torres-Carrasquillo, Elliot Singer, Mary A Kohler, Richard J Greene, Douglas A Reynolds, and John R Deller Jr. Approaches to language identification

using gaussian mixture models and shifted delta cepstral features. In *INTERSPEECH*, 2002.

W. Tutte. *Graph Theory*. Addison-Wesley, 1984.

Anand Venkataraman. A statistical model for word discovery in transcribed speech. *Comput. Linguist.*, 27(3):352–372, September 2001. ISSN 0891-2017. URL `http://dl.acm.org/citation.cfm?id=972655.972657`.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*, pages 2755–2763, 2015.

Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.

Hanna Wallach, David Mimno, and Andrew McCallum. Rethinking lda: Why priors matter. In *Advances in Neural Information Processing Systems*, pages 1973–1981, 2009.

Mengqiu Wang, Wanxiang Che, and Christopher D. Manning. Joint word alignment and bilingual named entity recognition using dual decomposition. In *Association for Computational Linguistics (ACL)*, 2013. URL `http://nlp.stanford.edu/pubs/wang-etal-acl13.pdf`.

BIBLIOGRAPHY

Wikipedia. Wikipedia, the free encyclopedia, 2015. URL http://en.wikipedia.org/. [Online; 2015].

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

Tae Yano, William W. Cohen, and Noah A. Smith. Predicting response to political blog posts with topic models. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pages 477–485, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics. ISBN 978-1-932432-41-1. URL http://dl.acm.org/citation.cfm?id=1620754.1620824.

Dani Yogatama, Yanchuan Sim, and Noah A. Smith. A probabilistic model for canonicalizing named entity mentions. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 685–693, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=2390524.2390621.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.

GuoDong Zhou and Jian Su. Named entity recognition using an hmm-based chunk tagger. In *Proceedings of the 40th Annual Meeting on Association for Compu-*

tational Linguistics, ACL '02, pages 473–480, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073163. URL http://dx.doi.org/10.3115/1073083.1073163.

Vita

Nicholas was born in Dallas, Texas, in 1984, to British parents. He grew up in Geneva, Switzerland before returning to the United States in 2000. He obtained Bachelors' degrees in Mathematics and Computer Science from Virginia Tech in Blacksburg, Virginia in 2008. Prior to joining the doctoral program at Johns Hopkins University, Nicholas worked as an associate scientist at BBN Technologies for one year, where he was converted to the Church of Emacs. He defended his Ph.D. in Computer Science at Johns Hopkins University in 2015, advised by Prof. Jason Eisner and Prof. Mark Dredze. He was affiliated with the Center for Language and Speech Processing and Human Language Technologies Center of Excellence. Nicholas resides in Washington D.C. with his girlfriend Maria-Veronica and their dog, Bubba.