Structured Databases of Named Entities from Bayesian Nonparametrics

Jacob Eisenstein  
*Carnegie Mellon University*

Tae Yano  
*Carnegie Mellon University*

William W. Cohen  
*Carnegie Mellon University, wcohen@cs.cmu.edu*

Noah A. Smith  
*Carnegie Mellon University, nasmith@cs.cmu.edu*

Eric P. Xing  
*Carnegie Mellon University, epxing@cs.cmu.edu*

Follow this and additional works at: http://repository.cmu.edu/lti

Part of the Computer Sciences Commons

Published In  
Proceedings of the EMNLP Workshop on Unsupervised Learning in NLP (UNSUP), 2-12.
Structured Databases of Named Entities from Bayesian Nonparametrics

Jacob Eisenstein  Tae Yano  William W. Cohen  Noah A. Smith  Eric P. Xing
School of Computer Science  Carnegie Mellon University  Pittsburgh, PA 15213, USA
{jacobeis,taey,wcohen,nasmith,epxing}@cs.cmu.edu

Abstract

We present a nonparametric Bayesian approach to extract a structured database of entities from text. Neither the number of entities nor the fields that characterize each entity are provided in advance; the only supervision is a set of five prototype examples. Our method jointly accomplishes three tasks: (i) identifying a set of canonical entities, (ii) inferring a schema for the fields that describe each entity, and (iii) matching entities to their references in raw text. Empirical evaluation shows that the approach learns an accurate database of entities and a sensible model of name structure.

1 Introduction

Consider the task of building a set of structured records from a collection of text: for example, extracting the names of people or businesses from blog posts, where each full name decomposes into fields corresponding to first-name, last-name, title, etc. To instruct a person to perform this task, one might begin with a few examples of the records to be obtained; assuming that the mapping from text to records is relatively straightforward, no additional instruction would be necessary. In this paper, we present a method for training information extraction software in the same way: starting from a small table of partially-complete “prototype” records (Table 1), our system learns to add new entries and fields to the table, while simultaneously aligning the records to text.

We assume that the dimensionality of the database is unknown, so that neither the number of entries nor the number of fields is specified in advance. To accommodate this uncertainty, we apply a Bayesian model which is nonparametric along three dimensions: the assignment of text mentions to entities (making popular entries more likely while always allowing new entries); the alignment of individual text tokens to fields (encouraging the re-use of common fields, but permitting the creation of new fields); and the assignment of values to entries in the database itself (encouraging the reuse of values across entries in a given field). By adaptively updating the concentration parameter of stick-breaking distribution controlling the assignment of values to entries in the database, our model can learn domain-specific information about each field: for example, that titles are often repeated, while names are more varied.

Our system’s input consists of a very small prototype table and a corpus of text which has been automatically segmented to identify names. Our desired output is a set of structured records in which each field contains a single string — not a distribution over strings, which would be more difficult to interpret. This requirement induces a tight probabilistic coupling between the assignment of text to cells in the table, so special care is required to ob-
tain efficient inference. Our procedure alternates between two phases. In the first phase, we perform collapsed Gibbs sampling on the assignments of string mentions to rows and columns in the table, while marginalizing the values of the table itself. In the second phase, we apply Metropolis-Hastings to swap the values of columns in the table, while simultaneously relabeling the affected strings in the text.

Our model performs three tasks: it constructs a set of entities from raw text, matches mentions in text with the entities to which they refer, and discovers general categories of tokens that appear in names (such as titles and first names). We are aware of no existing system that performs all three of these tasks jointly. We evaluate on a dataset of political blogs, measuring our system’s ability to discover a set of reference entities (recall) while maintaining a compact number of rows and columns (precision). With as few as five partially-complete prototype examples, our approach gives accurate tables that match well against a manually-annotated reference list. Our method outperforms a baseline single-link clustering approach inspired by one of the most successful entries (Elmacioglu et al., 2007) in the SEMEVAL “Web People Search” shared task (Artiles et al., 2007).

2 Task Definition

In this work, we assume that a bag of \(M\) mentions in text have been identified. The \(m\)th mention \(w_m\) is a sequence of contiguous word tokens (its length is denoted \(N_m\)) understood to refer to a real-world entity. The entities (and the mapping of mentions to entities) are not known in advance. While our focus in this paper is names of people, the task is defined in a more generic way.

Formally, the task is to construct a table \(x\) where rows correspond to entities and columns to functional fields. The number of entities and the number of fields are not prespecified. \(x_{i,j}\) denotes the \(j\)th column of \(x\), and \(x_{i,j}\) is a single word type filling the cell in row \(i\), column \(j\). An example is Table 1, where the fields are first-name, last-name, title, middle-name, and so on. In addition to the table, we require that each mention be mapped to an entity (i.e., a row in the table). Success at this task therefore requires (i) identifying entities, (ii) discovering the internal structure of mentions (effectively canonicalizing them), and (iii) mapping mentions to entities (therefore resolving coreference relationships among mentions). Note that this task differs from previous work on knowledge base population (e.g., McNamee, 2009) because the schema is not formally defined in advance; rather, the number of fields and their meaning must be induced from just a few prototype examples.

To incorporate partial supervision, a subset of the table \(x\) is specified manually by an annotator. We denote this subset of “prototypes” by \(\tilde{x}\); for entries that are unspecified by the user, we write \(\tilde{x}_{i,j} = \emptyset\). Prototypes are not assumed to provide complete information for any entity.

3 Model

We now craft a nonparametric generative story that explains both the latent table and the observed mentions. The model incorporates three nonparametric components, allowing an unbounded number of rows (entities) and columns (fields), as well as an unbounded number of values per column (field values). A plate diagram for the graphical model is shown in Figure 1.

A key point is that the column distributions \(\phi\) range over possible values at the entity level, not over mentions in text. For example, \(\phi_2\) might be the distribution over possible last names and \(\phi_3\) the distribution over elected office titles. Note that \(\phi_2\) would contain a low value for the last name Obama — which indicates that few people have this last name — even though a very high proportion of mentions in our data include the string Obama.

The user-generated entries (\(\tilde{x}\)) can still be treated as the outcome of the generative process: using exchangeability, we treat these entries as the first samples drawn in each column. In this work, we treat them as fully observed, but it is possible to treat them as noisy and incorporate a stochastic dependency between \(x_{i,j}\) and \(\tilde{x}_{i,j}\).

4 Inference

We now develop sampling-based inference for the model described in the previous section. We begin with a token-based collapsed Gibbs sampler, and then add larger-scale Metropolis-Hastings moves.
• **Generate the table entries.** For each column \( j \),
  - Draw a concentration parameter \( \alpha_j \) from a log-normal distribution, \( \log \alpha_j \sim \mathcal{N}(\mu, \sigma^2) \).
  - Draw a distribution over strings from a Dirichlet process \( \phi_j \sim DP(\alpha_j, G_0) \), where the base distribution \( G_0 \) is a uniform distribution over strings in a fixed character alphabet, up to an arbitrary finite length.
  - For each row \( i \), draw the entry \( x_{i,j} \sim \phi_j \).

• **Generate the text mentions.**
  - Draw a prior distribution over rows from a stick-breaking distribution, \( \tilde{\theta}_t \sim \text{Stick}(\eta_t) \).
  - Draw a prior distribution over columns from a stick-breaking distribution, \( \tilde{\theta}_t \sim \text{Stick}(\eta_c) \).
  - For each mention \( w_m \),
    * Draw a row in the table \( r_m \sim \theta_t \).
    * For each word token \( w_{m,n} \ (n \in \{1, \ldots, N_m\}) \),
      - Draw a column in the table \( c_{m,n} \sim \theta_c \).
      - Set the text \( w_m = x_{r_m,c_{m,n}} \).

4.1 Gibbs sampling

A key aspect of the generative process is that the word token \( w_{m,n} \) is completely determined by the table \( x \) and the row and column indicators \( r_m \) and \( c_{m,n} \): given that a token was generated by row \( i \) and column \( j \) of the table, it must be identical to the value of \( x_{i,j} \). Using Bayes’ rule, we can reverse this deterministic dependence: given the values for the row and column indices, the entries in the table are restricted to exact matches with the text mentions that they generate. This allows us to marginalize the unobserved entries in the table. We can also marginalize the distributions \( \theta_t, \theta_c \), and \( \phi_j \), using the standard collapsed Gibbs sampling equations for Dirichlet processes. Thus, sampling the row and column indices is all that is required to explore the entire space of model configurations.

4.1.1 Conditional probability for word tokens

The conditional sampling distributions for both rows and columns will marginalize the table (besides the prototypes \( \tilde{x} \)). To do this, we must be able to compute \( P(w_{m,n} \mid r_m = i, c_{m,n} = j, \tilde{x}, w_{-(m,n)}, r_{-m}, c_{-(m,n)}, \alpha_j) \), which represents the probability of generating word \( w_{m,n} \), given \( r_m = i \) and \( c_{m,n} = j \). The notation \( w_{-(m,n)}, r_{-m}, \) and \( c_{-(m,n)} \) represent the words, row indices, and column indices for all mentions besides \( w_{m,n} \). For simplicity, we will elide these variables in much of the subsequent notation.

We first consider the case where we have a user-specified entry for the row and column \( (i, j) \) — that is, if \( x_{ij} \neq \emptyset \). Then the probability is simply,

\[
P(w_{m,n} \mid r_m = i, c_{m,n} = j, \tilde{x}, \ldots) = \begin{cases} 1, & \text{if } x_{ij} = w_{m,n} \\ 0, & \text{if } x_{ij} \neq w_{m,n} \end{cases}
\]

(1)

Because the table cell \( x_{ij} \) is observed, we do not marginalize over it; we have a generative probability of one if the word matches, and zero otherwise. If the table cell \( x_{ij} \) is not specified by the user, then we marginalize over its possible values. For any given \( x_{ij} \), the probability \( P(w_{m,n} \mid x_{ij}, r_m = i, c_{m,n} = j) \) is still a delta function, so we have:

\[
\int P(w_{m,n} \mid r_m, c_{m,n})P(x_{r_m,c_{m,n}} \mid \ldots) \, dx_{r_m,c_{m,n}} = P(x = w_{m,n} \mid w_{-(m,n)}, r_{-m}, c_{-(m,n)}, \tilde{x}, \ldots)
\]

The integral is equal to the probability of the value of the cell \( x_{r_m,c_{m,n}} \) being identical to the string \( w_{m,n} \), given assignments to all other variables. To compute this probability, we again must consider two cases: if the cell \( x_{i,j} \) has generated some other string \( w'_{m',n'} \) then its value must be identical to that
string; otherwise it is unknown. More formally, for any cell \((i, j)\), if \(\exists w_{m',n'} : r_{m'} = i \land c_{m',n'} = j \land \langle m', n' \rangle \neq \langle m, n \rangle\), then \(P(x_{i,j} = w_{m',n'}) = 1\); all other strings have zero probability. If \(x_{i,j}\) has not generated any other entry, then its probability is conditioned on the other elements of the table \(x\). The known elements of this table are themselves determined by either the user entries \(\bar{x}\) or the observations \(w_{(m,n)}\). We can define these known elements as \(\bar{x}\), where \(\bar{x}_{ij} = \emptyset\) if \(\bar{x}_{ij} = \emptyset \land \not\exists\langle m, n \rangle : r_{m} = i \land c_{m,n} = j\). Then we can apply the standard Chinese restaurant process marginalization to obtain:

\[
P(x_{i,j} | \bar{x}_{-(i,j)}, \alpha) = \begin{cases} 
\frac{\text{N}(\bar{x}_{(i,j)} = x_{i,j})}{\text{N}(\bar{x}_{(i,j)} = x_{i,j}) + \alpha}, & \text{N}(\bar{x}_{(i,j)} = x_{i,j}) > 0 \\
\frac{\text{N}(\bar{x}_{(i,j)} = x_{i,j})}{\text{N}(\bar{x}_{(i,j)} = x_{i,j}) + \alpha}, & \text{N}(\bar{x}_{(i,j)} = x_{i,j}) = 0
\end{cases}
\]

(2)

In our implementation, we maintain the table \(\bar{x}\), updating it as we resample the row and column assignments. To construct the conditional distribution for any given entry, we first consult this table, and then compute the probability in Equation 2 for entries where \(\bar{x}_{ij} = \emptyset\).

### 4.1.2 Sampling columns

We can now derive sampling equations for the column indices \(c_{m,n}\). We first apply Bayes’ rule to obtain \(P(c_{m,n} | w_{m,n}, r_{m}, \ldots) \propto P(c_{m,n} | c_{-(m,n)}, \eta_{c}) \times P(w_{m,n} | c_{m,n}, r_{m, \bar{x}}, \ldots)\). The likelihood term \(P(w_{m,n} | c_{m,n}, r_{m}, \ldots)\) is defined in the previous section; we can compute the first factor using the standard Dirichlet process marginalization over \(\theta_{c}\). Writing \(\text{N}(c_{(m,n)} = j)\) for the count of occurrences of column \(j\) in the set \(c_{-(m,n)}\), we obtain

\[
P(c_{m,n} = j | c_{-(m,n)}, \eta_{c}) = \begin{cases} 
\frac{\text{N}(c_{-(m,n)} = j)}{\text{N}(c_{-(m,n)} = j) + \eta_{c}}, & \text{if } \text{N}(c_{-(m,n)} = j) > 0 \\
\frac{\text{N}(c_{-(m,n)} = j)}{\text{N}(c_{-(m,n)} = j) + \eta_{c}}, & \text{if } \text{N}(c_{-(m,n)} = j) = 0
\end{cases}
\]

(3)

### 4.1.3 Sampling rows

In principle the row indicators can be sampled identically to the columns, with the caveat that the generative probability \(P(w_{m} | r_{m}, \ldots)\) is a product across all \(N_m\) tokens in \(w_{m}\).\(^1\) However, because of

\(^1\)This relies on the assumption that the values of \(\{c_{m,n}\}\) are mutually independent given \(c_{-(m,n)}\). Future work might apply the tight probabilistic coupling between the row and column indicators, straightforward Gibbs sampling mixes slowly. Instead, we marginalize the column indicators while sampling \(r\). Only the likelihood term is affected by this change:

\[
P(w_{m} | r_{m}, w_{-m}, r_{-m}, \ldots) = \sum_{j} P(c = j | c_{-m}, \eta_{c})P(w_{m,n} | c_{m,n} = j, r_{m}, \bar{x}, \alpha).
\]

(4)

The tokens are conditionally independent given the row, so we factor and then explicitly marginalize over each \(c_{m,n}\). The chain rule gives the form in Equation 4, which contains terms for the prior over columns and the likelihood of the word; these are defined in Equations 2 and 3. Note that neither the inferred table \(\bar{x}\) nor the heldout column counts \(c_{-m}\) include counts from any of the cells in row \(m\).

### 4.2 Column swaps

Suppose that during initialization, we encounter the string Barack Obama before encountering Barry Obama. We would then put Barry in the first-name column, and put Barack in some other column for nicknames. After making these initial decisions, they would be very difficult to undo using Gibbs sampling — we would have to first shift all instances of Barry to another column, then move an instance of Barack to the first-name column, and then move the instances of Barack to the nickname column. To rectify this issue, we perform sampling on the table itself, swapping the columns of entries in the table, while simultaneously updating the relevant column indices of the mentions.

In the proposal, we select at random a row \(t\) and indices \(i\) and \(j\). In the table, we will swap \(x_{t,i}\) with \(x_{t,j}\); in the text we will swap the values of each \(c_{m,n}\) whenever \(r_{m} = t\) and \(c_{m,n} = i\) or \(j\). This proposal is symmetric, so no Hastings correction is required. Because we are simultaneously updating the table and the column indices, the generative likelihood of the words is unchanged; the only changes

\(1\) a more structured model of the ways that fields are combined when mentioning an entity. For example, a first-order Markov model could learn that family names often follow given names, but the reverse rarely occurs (in English).
in the overall likelihood come from the column indices and the values of the cells in the table. Letting \( x^*, c^* \) indicate the state of the table and column indices after the proposed move, we will accept with probability,

\[
P_{\text{accept}}(x \rightarrow x^*) = \min \left( 1, \frac{P(c^*) P(x^*)}{P(c) P(x)} \right)
\]

We first consider the ratio of the table probabilities, \( \frac{P(x^* | c)}{P(x | c)} \). Recall that each column of \( x \) is drawn from a Dirichlet process; appealing to exchangeability, we can treat the row \( t \) as the last element drawn, and compute the probabilities \( P(x_{t,i} \mid x_{-t,i}, \alpha_t) \), with \( x_{-t,i} \) indicating the elements of the column \( i \) excluding row \( t \). This probability is given by Equation 2. For a swap of columns \( i \) and \( j \), we compute the ratio:

\[
P(x_{t,i} \mid x_{-t,j}, \alpha_j) P(x_{t,j} \mid x_{-t,i}, \alpha_i) \quad \frac{P(x_{t,i} \mid x_{-t,i}, \alpha_i) P(x_{t,j} \mid x_{-t,j}, \alpha_j)}{P(x_{t,i} \mid x_{-t,j}, \alpha_j) P(x_{t,j} \mid x_{-t,i}, \alpha_i)}
\]

Next we consider the ratio of the column probabilities, \( \frac{P(c^*)}{P(c)} \). Again we can apply exchangeability, \( P(c) = P(\{c_m : r_m = t\} \mid \{c_{m'} : r_{m'} \neq t\}) P(\{c_{m'} : r_{m'} \neq t\}) \). The second term \( P(\{c_{m'} : r_{m'} \neq t\}) \) is unaffected by the move, and so is identical in both the numerator and denominator of the likelihood ratio; probabilities from columns other than \( i \) and \( j \) also cancel in this way. The remaining ratio can be simplified to,

\[
\left( \frac{P(c = j \mid c_{-t}, \eta_c)}{P(c = i \mid c_{-t}, \eta_c)} \right)^{N(r = t \cap c = j) - N(r = t \cap c = j)}
\]

where the counts \( N(\cdot) \) are from the state of the sampler before executing the proposed move. The probability \( P(c = i \mid c_{-t}, \eta_c) \) is defined in Equation 3, and the overall acceptance ratio for column swaps is the product of (6) and (7).

4.3 Hyperparameters

The concentration parameters \( \eta_r \) and \( \eta_c \) help to control the number of rows and columns in the table, respectively. These parameters are updated to their maximum likelihood values using gradient-based optimization, so our overall inference procedure is a form of Monte Carlo Expectation-Maximization (Wei and Tanner, 1990).

The concentration parameters \( \alpha_j \) control the diversity of each column in the table: if \( \alpha_j \) is low then we expect a high degree of repetition, as with titles; if \( \alpha_j \) is high then we expect a high degree of diversity. When the sampling procedure adds a new column, there is very little information for how to set its concentration parameter, as the conditional likelihood will be flat. Consequently, greater care must be taken to handle these priors appropriately.

We place a log-normal hyperprior on the column concentration parameters, \( \log \alpha_j \sim \mathcal{N}(\mu, \sigma^2) \). The parameters of the log-normal are shared across columns, which provides additional information to constrain the concentration parameters of newly-created columns. We then use Metropolis-Hastings to sample the values of each \( \alpha_j \), using the joint likelihood,

\[
P(\alpha_j, \bar{x}^{(j)} \mid \mu, \sigma^2) \propto \exp\left(\frac{-(\log \alpha_j - \mu)^2}{2\sigma^2}\right) \alpha_j^{k_j} \Gamma(\alpha_j),
\]

where \( \bar{x}^{(j)} \) is column \( j \) of the inferred table, \( n_j \) is the number of specified entries in column \( j \) of the table \( \bar{x} \) and \( k_j \) is the number of unique entries in the column; see Rasmussen (2000) for a derivation. After repeatedly sampling several values of \( \alpha_j \) for each column in the table, we update \( \mu \) and \( \sigma^2 \) to their maximum-likelihood estimates.

5 Temporal Prominence

Andy Warhol predicted, “in the future, everyone will be world-famous for fifteen minutes.” A model of temporal dynamics that accounts for the fleeting and fickle nature of fame might yield better performance for transient entities, like Joe the Plumber. Among several alternatives for modeling temporal dynamics in latent variable models, we choose a simple nonparametric approach: the recurrent Chinese restaurant process (RCRP; Ahmed and Xing, 2008). The core idea of the RCRP is that time is partitioned into epochs, with a unique Chinese restaurant process in each epoch. Each RCP has a prior which takes the form of pseudo-counts computed from the counts in previous epochs. We employ the simplest version of the RCRP, a first-order Markov model in which the prior for epoch \( t \) is equal to the vector of counts for epoch \( t - 1 \):
A system's evaluation metrics according to Doddington et al. (2004) suggest an appeal to evaluating cross-document coreference as a system's performance. We ran the Stanford Named Entity Recognition system (Finkel et al., 2005), and performed a qualitative analysis. Our preprocessing pipeline includes tokenization and named entity recognition. Gold standard evaluation was performed on a corpus of blogs discussing United States politics in 2008 (Eisenstein and Xing, 2010). We developed a reference set of 100 unique mention strings for evaluating the unique name strings in the training set by frequency, and manually merging strings that reference the same entity. We also manually discarded strings from the reference set if they resulted from errors in the preprocessing pipeline (tokenization and named entity recognition). Each entity is represented by the set of all word tokens that appear in its references; there are a total of 231 tokens for the 100 entities. Most entities only include first and last names, though the most frequent entities have many more; for example, the entity Barack Obama has known names: \{Barack, Obama, Sen., Mr.\}.

## Metrics
We evaluate the recall and precision of a system’s response set by matching against the reference set. The first step is to create a bipartite matching between response and reference entities. Using a cost function that quantifies the similarity of each mention string with the entities and fields to which they refer, we are aware of no prior work that performs these tasks jointly, nor any dataset that is annotated for all three tasks. Consequently, we focus our quantitative evaluation on what we take to be the most important subtask: identifying the entities which are mentioned in raw text. We annotate a new dataset of blog text for this purpose, and design precision and recall metrics to reward systems that recover as much of the reference set as possible, while avoiding spurious entities and fields. We also perform a qualitative analysis, noting the areas where our method outperforms string matching approaches, and where there is need for further improvement.

### Data
Evaluation was performed on a corpus of blogs describing United States politics in 2008 (Eisenstein and Xing, 2010). We ran the Stanford Named Entity Recognition system (Finkel et al., 2005) to obtain a set of 25,000 candidate mentions which the system judged to be names of people. We then pruned strings that appeared fewer than four times and eliminated strings with more than seven tokens (these were usually errors). The resulting dataset has 19,247 mentions comprising 45,466 word tokens, and 813 unique mention strings.

### Gold standard
We develop a reference set of 100 entities for evaluation. This set was created by sorting the unique name strings in the training set by frequency, and manually merging strings that reference the same entity. We also manually discarded strings from the reference set if they resulted from errors in the preprocessing pipeline (tokenization and named entity recognition). Each entity is represented by the set of all word tokens that appear in its references; there are a total of 231 tokens for the 100 entities. Most entities only include first and last names, though the most frequent entities have many more; for example, the entity Barack Obama has known names: \{Barack, Obama, Sen., Mr.\}.

### Metrics
We evaluate the recall and precision of a system’s response set by matching against the reference set. The first step is to create a bipartite matching between response and reference entities. Using a cost function that quantifies the simil-
ilarity of response and reference entities, we opti-
mize the matching using the Kuhn-Munkres algo-
rithm (Kuhn, 1955). For recall, the cost function
counts the number of shared word tokens, divided
by the number of word tokens in the reference enti-
ties; the recall is one minus the average cost of the
best matching (with a cost of one for reference enti-
ties that are not matched, and no cost for unmatched
response entities). Precision is computed identically,
but we normalize by the number of word tokens in
the response entity. Precision assigns a penalty of
one to unmatched response entities and no penalty
for unmatched reference entities.

Note that this metric grossly underrates the preci-
sion of all systems: the reference set is limited to 100
entities, but it is clear that our text mentions many
other people. This is harsh but fair: all systems are
penalized equally for identifying entities that are not
present in the reference set, and the ideal system will
recover the fifty reference entities (thus maximizing
recall) while keeping the table as compact as possi-
ble (thus maximizing precision). However, the raw
precision values have little meaning outside the con-
text of a direct comparison under identical experi-
mental conditions.

**Systems** The initial seed set for our system con-
sists of a partial annotation of five entities (Table 1)
— larger seed sets did not improve performance. We
run the inference procedure described in the previ-
ous section for 20,000 iterations, and then obtain a
final database by taking the intersection of the in-
ferrer tables X obtained at every 100 iterations, start-
ing with iteration 15,000. To account for variance
across Markov chains, we perform three different
runs. We evaluate a non-temporal version of our
model (as described in Sections 3 and 4), and a tem-
poral version with 5 epochs. For the non-temporal
version, a non-parallel C implementation had a wall
clock sampling time of roughly 16 hours; the tem-
poral version required 24 hours.

We compare against a baseline that incrementally
clusters strings into entities using a string edit dis-
tance metric, based on the work of Elmacioglu et
al. (2007). Starting from a configuration in which
each unique string forms its own cluster, we incre-
mentally merge clusters using the single-link crite-
rian, based on the minimum Jaccard edit distance

![Figure 2: The precision and recall of our models, as compared to the curve defined by the incremental clustering baseline. Each point indicates a unique sampling run.](image)

<table>
<thead>
<tr>
<th>Bill</th>
<th>Clinton</th>
<th>Benazir</th>
<th>Bhum [2007]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nancy</td>
<td>Pelosi</td>
<td>Speaker</td>
<td>Roberts</td>
</tr>
<tr>
<td>Martin</td>
<td>King</td>
<td>Dr.</td>
<td>Luther</td>
</tr>
<tr>
<td>Bill</td>
<td>Nelson</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2: A subset of the entity database discovered by our model, hand selected to show highlight interesting success and failure cases.

between each pair of clusters. This yields a series of
outputs that move along the precision-recall curve,
with precision increasing as the clusters encompass
more strings. There is prior work on heuristics for
selecting a stopping point, but we compare our re-
sults against the entire precision-recall curve (Man-
ing et al., 2008).

**7 Results**

The results of our evaluation are shown in Figure 2.
All sampling runs from our models lie well beyond
the precision-recall curve defined by the baseline
system, demonstrating the ability to achieve reason-
able recall with a far more compact database. The
baseline system can achieve nearly perfect recall by
creating one entity per unique string, but as it merges
strings to improve precision, its recall suffers sig-
nificantly. As noted above, perfect precision is not
possible on this task, because the reference set cov-
ers only a subset of the entities that appear in the
data. However, the numbers do measure the ability
to recover the reference entities in the most compact
table possible, allowing a quantitative comparison of
our models and the baseline approach.
Table 2 shows a database identified by the atemporal version of our model. The most densely-populated columns in the table correspond to well-defined name parts: columns 1 and 2 are almost exclusively populated with first and last names respectively, and column 3 is mainly populated by titles. The remaining columns are more of a grab bag. Column 4 correctly captures Jr for Martin Luther King; column 5 correctly captures Luther, but mistakenly contains Roberts (thus merging the John Kerry and John Roberts entities), and Bhutto (thus helping to merge the Bill Clinton and Benazir Bhutto entities).

The model successfully distinguishes some, but not all, of the entities that share tokens. For example, the model separates Bill Clinton from Bill Nelson; it also separates John McCain from John Kerry (whom it mistakenly merges with John Roberts). The ability to distinguish individuals who share first names is due in part to the model attributing a low concentration parameter to first names, meaning that some repetition in the first name column is expected. The model correctly identifies several titles and alternative names, including the rare title Speaker for Nancy Pelosi; however, it misses others, such as the Senator title for Bill Nelson. This may be due in part to the sample merging procedure used to generate this table, which requires that a cell contain the same value in at least 80% of the samples.

Many errors may be attributed to slow mixing. After mistakenly merging Bhutto and Clinton at an early stage, the Gibbs sampler — which treats each mention independently — is unable to separate them. Given that several other mentions of Bhutto are already in the row occupied by Clinton, the overall likelihood would benefit little from creating a new row for a single mention, though moving all such mentions simultaneously would result in an improvement. Larger scale Metropolis-Hastings moves, such as split-merge or type-based sampling (Liang et al., 2010) may help.

8 Related Work

Information Extraction A tradition of research in information extraction focuses on processing raw text to fill in the fields of manually-defined templates, thus populating databases of events or relations (McNamee and Dang, 2009). While early approaches focused on surface-level methods such as wrapper induction (Kushmerick et al., 1997), more recent work in this area includes Bayesian nonparametrics to select the number of rows in the database (Haghighi and Klein, 2010a). However, even in such nonparametric work, the form of the template and the number of slots are fixed in advance. Our approach differs in that the number of fields and their meaning is learned from data. Recent work by Chambers and Jurafsky (2011) approaches a related problem, applying agglomerative clustering over sentences to detect events, and then clustering syntactic constituents to induce the relevant fields of each event entity. As described in Section 6, our method performs well against an agglomerative clustering baseline, though a more comprehensive comparison of the two approaches is an important step for future work.

Name Segmentation and Structure A related stream of research focuses specifically on names: identifying them in raw text, discovering their structure, and matching names that refer to the same entity. We do not undertake the problem of named entity recognition (Tjong Kim Sang, 2002), but rather apply an existing NER system as a preprocessing step (Finkel et al., 2005). Typical NER systems do not attempt to discover the internal structure of names or a database of canonical names, although they often use prefabricated “gazetteers” of names and name parts as features to improve performance (Borthwick et al., 1998; Sarawagi and Cohen, 2005).

Charniak (2001) shows that it is possible to learn a model of name structure, either by using coreference information as labeled data, or by leveraging a small set of hand-crafted constraints. Elsner et al. (2009) develop a nonparametric Bayesian model of name structure using adaptor grammars, which they use to distinguish types of names (e.g., people, places, and organizations). Li et al. (2004) use a set of manually-crafted “transformations” of name parts to build a model of how a name might be rendered in multiple different ways. While each of these approaches bears on one or more facets of the problem that we consider here, none provides a holistic treatment of name disambiguation and structure.
Resolving Mentions to Entities  The problem of resolving mentions to entities has been approached from a variety of different perspectives. There is an extensive literature on probabilistic record linkage, in which database records are compared to determine if they are likely to have the same real-world referents (e.g., Felligi and Sunter, 1969; Bilenko et al., 2003). Most approaches focus on pairwise assessments of whether two records are the same, whereas our method attempts to infer a single coherent model of the underlying relational data. Some more recent work in record linkage has explicitly formulated the task of inferring a latent relational model of a set of observed datasets (e.g., Cohen et al., 2000; Pasula et al., 2002; Bhattacharya and Getoor, 2007); however, to our knowledge, these prior models have all exploited some predefined database schema (i.e., set of columns), which our model does not require. Many of these prior models have been applied to bibliographic data, where different conventions and abbreviations lead to imperfect matches in different references to the same publication. In our task, we consider name mentions in raw text; such mentions are short, and may not offer as many redundant clues for linkage as bibliographic references.

In natural language processing, coreference resolution is the task of grouping entity mentions (strings), in one or more documents, based on their common referents in the world. Although much of coreference resolution has on the single document setting, there has been some recent work on cross-document coreference resolution (Li et al., 2004; Haghghi and Klein, 2007; Poon and Domingos, 2008; Singh et al., 2011). The problem we consider is related to cross-document coreference, although we take on the additional challenge of providing a canonicalized name for each referent (the corresponding table row), and in inferring a structured representation of entity names (the table columns). For this reason, our evaluation focuses on the induced table of entities, rather than the clustering of mention strings. The best coreference systems depend on carefully crafted, problem-specific linguistic features (Bengtson and Roth, 2008) and external knowledge (Haghghi and Klein, 2010b). Future work might consider how to exploit such features for the more holistic information extraction setting.

9 Conclusion

This paper presents a Bayesian nonparametric approach to recover structured records from text. Using only a small set of prototype records, we are able to recover an accurate table that jointly identifies entities and internal name structure. In our view, the main advantage of a Bayesian approach compared to more heuristic alternatives is that it facilitates incorporation of additional information sources when available. In this paper, we have considered one such additional source, incorporating temporal context using the recurrent Chinese restaurant process.

We envision enhancing the model in several other respects. One promising direction is the incorporation of name structure, which could be captured using a first-order Markov model of the transitions between name parts. In the nonparametric setting, a transition matrix is unbounded along both dimensions, and this can be handled by a hierarchical Dirichlet process (HDP; Teh et al 2006).4 We envision other potential applications of the HDP: for example, learning “topics” of entities which tend to appear together (i.e., given a mention of Mahmoud Abbas in the American press, a mention of Benjamin Netanyahu is likely), and handling document-specific burstiness (i.e., given that an entity is mentioned once in a document, it is much more likely to be mentioned again). Finally, we would like to incorporate lexical context from the sentences in which each entity is mentioned, which might help to distinguish, say, computer science researchers who share names with former defense secretaries or professional basketball players.

Acknowledgments This research was enabled by AFOSR FA95501010247, DARPA grant N10AP20042, ONR N000140910758, NSF DBI-0546594, IIS-0713379, IIS-0915187, IIS-0811562, an Alfred P. Sloan Fellowship, and Google’s support of the Worldly Knowledge project at CMU. We thank the reviewers for their thoughtful feedback.

4One of the reviewers proposed to draw entire column sequences from a Dirichlet process. Given the relatively small number of columns and canonical name forms, this may be a straightforward and effective alternative to the HDP.
References


George Doddington, Alexis Mitchell, Mark Przybokci, Lance Ramshaw, Stephanie Strassel, and Ralph Weischedel. 2004. The automatic content extraction (ace) program: Tasks, data, and evaluation. In *4th international conference on language resources and evaluation (LREC’04)*.


