Creating Probabilistic Databases from Information Extraction Models

Rahul Gupta*
IBM Research Lab
New Delhi, India
rahulgupta@in.ibm.com

Sunita Sarawagi
Indian Institute of Technology
Bombay, India
sunita@iitb.ac.in

ABSTRACT

Many real-life applications depend on databases automatically curated from unstructured sources through imperfect structure extraction tools. Such databases are best treated as imprecise representations of multiple extraction possibilities. State-of-the-art statistical models of extraction provide a sound probability distribution over extractions but are not easy to represent and query in a relational framework. In this paper we address the challenge of approximating such distributions as imprecise data models. In particular, we investigate a model that captures both row-level and column-level uncertainty and show that this representation provides significantly better approximation compared to models that use only row or only column level uncertainty. We present efficient algorithms for finding the best approximating parameters for such a model; our algorithm exploits the structure of the model to avoid enumerating the exponential number of extraction possibilities.

1. INTRODUCTION

Large text databases obtained by integrating unstructured data from multiple sources are central to many real-life applications, including citation databases like Citeseer, product comparison databases, and personal information management systems (PIM). A key step in the creation of such databases is the extraction of structured entities from unstructured sources. Typically such extraction is not a one-time activity but needs to be performed continuously to keep the database in sync with the unstructured sources. For example, in PIM databases, it is important that as new unstructured documents get added, we extract structured entities from them and integrate with existing database entities. Clearly, such synchronous, large-scale extractions cannot be performed under full user supervision and typically such systems depend on some automated means of structure extraction.

*Work done as a student at IIT Bombay

VLDB '06, September 12-15, 2006, Seoul, Korea.

Automatically extracting structured entities from unstructured text is a challenging problem, and has a long history of attempts spanning early rule-based systems like Rapier [7] to the newest statistical methods like Conditional Random Fields [29]. Unfortunately, it is impossible to guarantee perfect extraction accuracy in real-life deployment settings even with the latest extraction tools. The problem is more severe when the sources are extremely heterogeneous, making it impossible to hand tune any extraction tool to perfection. For example, the extraction of structured entities like author names, title and journal names from citations has an accuracy of close to 90% for individual fields and only 70–80% for multi-field records [22].

One method of surmounting the problem of extraction errors is to require that each extracted entity be attached with confidence scores that correlate with the probability that the extracted entities are correct. Normally, even this is a hard goal to achieve and only recently have probabilistic extraction models like Conditional Random Fields(CRFs) been found to provide sound confidence scores. We provide illustrations of this in Section 2.1. In addition, such models can output not just a single best extraction but a ranked list of extractions where each extracted record is associated with a probability of correctness. Such probabilistic results can be stored in an imprecise data management system [16, 27, 5, 15, 3, 11] for getting probabilistic answers. For example, in Figure 3 we show a list of extractions along with probability scores for an address database. We will denote each such extraction result by *segmentation* as it consists of labeled segments of the unstructured string; and call this representation the segmentation per row model [14]. While this representation of uncertainty is natural and allows for simple query execution semantics, the number of extraction results can in the worst case be exponentially large as we show in Section 2.

Thus, we need an alternative model of representing imprecision in a database that captures the original distribution, while being easy to store and query. Our goal is to choose a generic model of uncertainty so that we can leverage on the active research on efficient query processing on these models [11, 24, 8, 5]. Different representations provide varying tradeoffs in the flexibility they offer and how complex it is to process probabilistic queries over them [27]. We consider two models from the literature. First is the popular column uncertainty model where the rows of a database table

are independent of each other and the columns of each row represent uncertainty as independent probability distributions [21]. An example of such a representation appears in Figure 5. Each column in the figure is associated with an independent multinomial distribution over extracted segments. We will refer to this model as the one-row model.

The second model is a generalization of the first model where we capture both column and tuple level uncertainty by storing multiple rows per extracted record. Each row has independent column distributions and a row-level probability. We show in the paper that even a two-row model in this category can significantly improve the accuracy of approximation compared to the one-row model. Also, the number of rows required is *significantly* smaller than the number of rows required if segmentations are stored explicitly. We present efficient algorithms for directly finding the optimal parameters of the single-row and multi-row column distributions without requiring the explicit enumeration of the possibly exponential number of segmentations. Our algorithm directly operates on the model to discover a locally optimal set of informative variables that form the basis for each row's distribution.

Empirical evaluation on two real-life datasets show that our method of creating tractable database representations of the imprecision of information extraction models is efficient and accurate. To the best of our knowledge, ours is the first paper on transforming the uncertainty of complicated, yet high performing statistical models of extraction to tractable models of imprecision that can be easily stored and queried in a database.

Outline

The rest of the paper is organized as follows. We describe our problem and present a background of state of the art probabilistic models of information extraction in Section 2. In Sections 3.1 and 3.2 we present algorithms for generating best one row and multi row distributions. In Section 4 we present experimental evaluation of the accuracy and efficiency of our model. Related work and conclusions appear in Section 5 and 6 respectively.

2. SETUP

We assume that our unstructured data source is a collection of independent text records representing entities like addresses and citations. The process of extraction converts each unstructured record into a structured record in a table in a target database consisting of columns $A_1, A_2 \ldots A_K$. Examples of entity columns are house number, street names, city name and state names for addresses and author-list, title, journal name and year for citations. One or more of the columns could be missing in an unstructured record in which case we assign it a value of NULL. Thus, each entity label can appear zero or once in each unstructured record. An unstructured record could contain words that do not belong to any of the entities. We assign all these words a special label called "Other".

In such a setting the process of extraction can be viewed as a segmentation of the word sequence of an unstructured record where each segment is either one of K entities $A_1 \ldots A_K$ or part of the "Other" label. We next review existing methods of automatic extraction.

2.1 Models for automatic extraction

The problem of extracting structured entities from unstructured data is an extensively researched topic. A number of models have been proposed ranging from the earliest rule-learning models to probabilistic approaches based on graphical models like Hidden Markov Machines [28, 4, 1] and max-ent taggers [23]. A promising recently proposed model for information extraction is Conditional Random Fields (CRFs). CRFs have been shown [20, 29] to outperform most prior approaches. The state-of-the-art methods for extraction called Semi-CRFs construct a probability distribution over segmentations of the input sequence [26, 10]. We describe this approach next.

The input unstructured text is treated as a sequence of tokens $\mathbf{x} = x_1 \ldots x_n$. A segmentation $\mathbf{s}$ of an input sequence $\mathbf{x}$ is a sequence of segments $s_1 \ldots s_p$ such that the last segment ends at n, the first segment starts at 1, and segment s_{j+1} begins right after segment s_j ends. Each segment s_j consists of a *start position* t_j, an *end position* u_j, and a *label* $y_j \in Y$. In our case Y consists of the K attribute labels $\{A_1 \ldots A_K\}$ and a special label "Other". For example, a segmentation of the record "52-A Goregaon West Mumbai PIN 400 062" might be (52-A, 'House_no') (Goregaon West, 'Area') (Mumbai, 'City') (PIN 'Other') (400 062, 'Pincode') (See Figure 3).

A Semi-CRF models the conditional probability distribution over segmentation $\mathbf{s}$ for a given input sequence $\mathbf{x}$ as follows:

$$\Pr(\mathbf{s}|\mathbf{x}, \mathbf{\Lambda}) = \frac{1}{Z(\mathbf{x})} \mathbf{exp}(\mathbf{\Lambda} \cdot \sum_{\mathbf{j}} \mathbf{f}(j, \mathbf{x}, \mathbf{s})) \quad (1)$$

where $\mathbf{f}(j, \mathbf{x}, \mathbf{s})$ is a vector of local feature functions $f_1 \ldots f_N$ of $\mathbf{s}$ at the jth segment and $\mathbf{\Lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_N)$ is a weight vector that encodes the importance of each feature function in $\mathbf{f}$. $Z(\mathbf{x}) = \sum_{\mathbf{s}'} \mathbf{exp}(\mathbf{\Lambda} \cdot \sum_{\mathbf{j}} \mathbf{f}(j, \mathbf{x}, \mathbf{s}'))$ is a normalization factor. The label of a segment depends on the label of the previous segment and the properties of the tokens comprising this segment. Thus a feature for segment $s_j = (t_j, u_j, y_j)$ is a function of the form $f(y_j, y_{j-1}, \mathbf{x}, t_j, u_j)$ that returns a numeric value. Example of such features are:

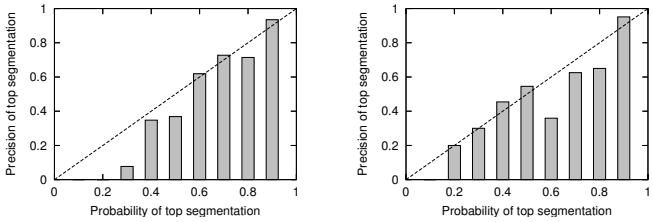
$$\begin{aligned} f_8(y_i, y_{i-1}, \mathbf{x}, 3, 5) &= [\![x_3x_4x_5 \text{ appears in a journal list}]\!] \\ &\quad \cdot [\![y_i = \text{journal}]\!] \\ f_{12}(y_i, y_{i-1}, \mathbf{x}, 19, 19) &= [\![x_{19} \text{ is an integer}]\!] \\ &\quad \cdot [\![y_i = \text{year}]\!] \cdot [\![y_{i-1} = \text{month}]\!] \end{aligned}$$

Thus, Equation 1 in the expanded form is

$$\Pr(\mathbf{s} = ((t_1, u_1, y_1), \ldots, (t_p, u_p, y_p))|\mathbf{x}, \mathbf{\Lambda}) = \frac{1}{Z(\mathbf{x})} \mathbf{exp}(\sum_{j=1}^{p} \sum_{r=1}^{N} \lambda_r f_r(y_j, y_{j-1}, \mathbf{x}, t_j, u_j))$$

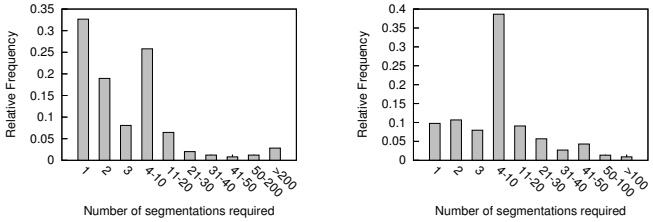
The weight vector $\mathbf{\Lambda}$ is learnt during training via a variety of methods, such as likelihood maximization [20].

During *extraction*, the goal is to find a segmentation $\mathbf{s} = s_1 \ldots s_p$ of the input sequence $\mathbf{x} = x_1 \ldots x_n$ such that $\Pr(\mathbf{s}|\mathbf{x}, \mathbf{\Lambda})$



(a) Cora (b) Address

Figure 1: Reliability plots for two datasets. The diagonal line denotes the ideal ending points of the bars.



(a) Cora, p=0.95 (b) Address, p=0.9

Figure 2: Histogram of the number of segmentations required for covering a probability mass p.

(as defined by Equation 1) is maximized.

$$\arg\max_{\mathbf{s}} \Pr(\mathbf{s}|\mathbf{x}, \mathbf{\Lambda}) = \arg\max_{\mathbf{s}} \mathbf{\Lambda} \cdot \sum_j \mathbf{f}(y_j, y_{j-1}, \mathbf{x}, t_j, u_j)$$

The right hand side can be efficiently computed using dynamic programming. Let L be an upper bound on segment length. Let $\mathbf{s}_{i:y}$ denote set of all partial segmentation starting from 1 (the first index of the sequence) to i, such that the last segment has the label y and ending position i. Let $V(i, y)$ denote the largest value of $\mathbf{\Lambda} \cdot \sum_\mathbf{j} \mathbf{f}(j, \mathbf{x}, \mathbf{s}')$ for any $\mathbf{s}' \in \mathbf{s}_{i:y}$. The following recursive calculation finds the best segmentation:

$$V(i,y) = \begin{cases} \max_{y', i'=i-L \ldots i-1} V(i', y') \\ \quad + \mathbf{\Lambda} \cdot \mathbf{f}(y, y', \mathbf{x}, i'+1, i) & \text{if } i > 0 \\ 0 & \text{if } i = 0 \\ -\infty & \text{if } i < 0 \end{cases} \quad (2)$$

The best segmentation then corresponds to the path traced by $\max_y V(|\mathbf{x}|, y)$. The probability of the best segmentation can be calculated using Equation 1. The above algorithm can be easily extended to find not just a single highest probability segmentation but the k highest scoring segmentations for any given k.

In typical real-life extraction tasks the highest scoring extraction is not necessarily the correct extraction. Fortunately, unlike in earlier generative models like HMMs, the probability of a segmentation as output by a CRF is sound in the sense that a probability of p denotes that there is a $100p\%$ chance that the extraction is correct. We illustrate this via reliability plots in Figure 1 for extraction on two datasets (descriptions of these datasets appear in Section 4) where we show binned probabilities as output by the model against true accuracy calculated as the fraction of cases that were correct in that bin. We observe that the histograms are very close to the 45 degree line which confirms the claim that the probabilities are sound.

Also, in typical extraction models k might need to be quite large to cover a significant probability mass. In Figure 2 we plot the histogram of the number of segmentations needed to cover a probability mass of 0.9. In both the datasets, a significant number of strings require more than just the top-10 segmentations.

These experiments establish that it is important to look beyond the topmost segmentation and challenges existing approaches that treat extraction as a 0/1 process of extracting just the highest scoring entities.

2.2 Database representation of extraction uncertainty

We consider three alternatives for representing the uncertainty of extraction in a database system.

In each case we show how to return probabilistic answers to a project query of the form

$$\text{select } y_1, \ldots y_l \text{ from } T \text{ where source=}\mathbf{x}$$

where T is an imprecise table and each y_i refers to one of the K columns $A_1 \ldots A_K$ of T. Probabilistic results for such a query will return a set of rows where each row r consists of l segments of $\mathbf{x}$ $(t_1, u_1), \ldots, (t_l, u_l)$ and a probability value $p_r = \Pr(y_1 = (t_1, u_1), \ldots, y_l = (t_l, u_l))$. We also allow a segment (t_i, u_i) to be $NULL$ so as to incorporate missing labels.

2.2.1 *Segmentation per row model*

A straightforward representation is to first extract from the original model all segmentations with non-zero probability and represent each segmentation $\mathbf{s}$ as a separate row with a tuple-level uncertainty value equal to the probability $\Pr(\mathbf{s})$ of that segmentation [14]. Thus, each unstructured source record r will give rise to a variable number of rows in the database; all rows of the same record are linked via a shared key that constraints their probabilities to sum to 1. An example of such a representation appears in Figure 4.

In this model the probability of a query result is calculated by summing over all segmentations that match the result row as follows:

$$\Pr(y_1 = (t_1, u_1), \ldots, y_l = (t_l, u_l)) = \sum_{\mathbf{s}: \forall i (t_i, u_i, y_i) \in \mathbf{s}} \Pr(\mathbf{s})$$

The main shortcoming of this approach is that the number of segmentations could be exponential in the size of the source

	House_no	Area	City	Pincode	Probability
s_1	52	Goregaon West	Mumbai	400 062	0.1
s_2	52-A	Goregaon	West Mumbai	400 062	0.2
s_3	52-A	Goregaon West	Mumbai	400 062	0.5
s_4	52	Goregaon	West Mumbai	400 062	0.2

Figure 3: Four segmentations of the address string '52-A Goregaon West Mumbai 400 076' along with their probabilities.

Id	House_no	Area	City	Pincode	Prob
1	52	Goregaon West	Mumbai	400 062	0.1
1	52-A	Goregaon	West Mumbai	400 062	0.2
1	52-A	Goregaon West	Mumbai	400 062	0.5
1	52	Goregaon	West Mumbai	400 062	0.2

Figure 4: Segmentation-per-row model for the example in Figure 3.

Id	House_no	Area	City	Pincode
1	52 (0.3) 52-A (0.7)	Goregaon West (0.6) Goregaon (0.4)	Mumbai (0.6) West Mumbai (0.4)	400 062 (1.0)

Figure 5: One-row model for the example in Figure 3.

Id	House_no	Area	City	Pincode	Prob
1	52 (0.167) 52-A (0.833)	Goregaon West (1.0)	Mumbai (1.0)	400 062 (1.0)	0.6
1	52 (0.5) 52-A (0.5)	Goregaon (1.0)	West Mumbai (1.0)	400 062 (1.0)	0.4

Figure 6: An optimal and exact two-row model for the example in Figure 3.

string. A naive way to deal with this problem is to take enough top segmentations that cover a big enough probability mass (say 0.95), and store these segmentations exactly. However, as Figure 2 illustrates, for quite a few strings we may end up retrieving a large number of segmentations.

Hence we are limited by two things - (a) we cannot afford to enumerate the top segmentations and (b) we are only allowed to populate a small number of rows in the imprecise database. To tackle these issues, instead of representing the uncertainty exactly, we will resort to approximate methods.

2.2.2 *One-row model*

In this representation we allocate one row per unstructured record but each column instead of being a precise value is a distribution over possible values it can take [21]. For the case of discrete text data, a natural choice for the column distribution is a multinomial distribution which assigns a probability value for each segment (i, j) of the input $\mathbf{x}$ such that the sum of the probability over all segments is equal to 1. An example of such a distribution appears in Figure 5.

Let $Q_y(i, j)$ denote the probability for segment (i, j) for column y. Then the probability of the query result $\mathbf{q} = (y_1 = (t_1, u_1) \ldots y_l = (t_l, u_l))$ is:

$$Q(\mathbf{q}) = \prod_{(t_i, u_i, y_i) \in \mathbf{q}} Q_{y_i}(t_i, u_i) \quad (3)$$

In Section 3.1 we show how to compute the best values of the $Q_y(i, j)$ parameters from the original extraction model.

While the one-row representation is storage-wise compact, it has the potential of grossly violating the true distribution because it treats the column distributions as being independent of each other. For example, the probability of the query result ((Area='Goregaon West'), (City='Mumbai')) in this model is $0.6 \times 0.6 = 0.36$ whereas the true probability from Figure 3 is $0.5 + 0.1 = 0.6$.

2.2.3 *Multi-row model*

We generalize the one-row model to storing multiple rows per extracted record with each row storing its own set of independent column distributions. In addition, each row has a row-level probability and the probability of a segmentation is obtained via a weighted sum of probability induced by each row. Let π_k denote the row probability of the k^{th} row and $Q_y^k(i, j)$ denote the multinomial parameter for segment (i, j) for column y of the k^{th} row. We present an example in Figure 6 where the number of rows m is 2.

In this case, the probability of a query result $\mathbf{q}$ is computed as

$$Q(\mathbf{q}) = \sum_{k=1}^{m} \pi_k \prod_{(t_i, u_i, y_i) \in \mathbf{q}} Q_{y_i}^k(t_i, u_i) \quad (4)$$

This model allows the approximate probability of a segmentation $\mathbf{s}$, $Q(\mathbf{s})$ to be broken down into additive components, with each component being a product of marginals rather than an unwieldy joint distribution.

2.3 Quantifying approximation quality

Since the one-row and multi-row models are approximations of the true distribution $P(\mathbf{s})$, we need a measure of the divergence between the two. A popular metric from statistics for measuring the gap between a true distribution $P(\mathbf{s})$ and

its approximation $Q(\mathbf{s})$ is KL-divergence.

$$KL(P||Q) = \sum_{\mathbf{s}} P(\mathbf{s}) \log \frac{P(\mathbf{s})}{Q(\mathbf{s})} \quad (5)$$

KL-divergence achieves its minima of zero iff $Q = P$. This measure has been found to provide more robust and meaningful values of distances between distributions than generic vector measures like L2 and L1. Further, it can be shown that $L1(P||Q) \leq (KL(P||Q)2\log 2)^{\frac{1}{2}}$. Another advantage of this measure is that it is continuous and allows the use of numerical optimization methods to choose the best approximation.

3. APPROXIMATIONS

In this section we show how to compute the parameters of the one-row and multi-row model so as to minimize the divergence with the true extraction model $P(\mathbf{s})$. Our goal is to design methods that do not require the explicit enumeration of all possible segmentations from the model because there can be an exponentially (in the length of the input) large number of them.

3.1 One row model

The multinomial parameters of a one-row model can be easily computed by directly optimizing on the KL function.

$$\begin{aligned}
&\min_Q KL(P||Q) \equiv \max_Q \sum_{\mathbf{s}} P(\mathbf{s}) \log Q(\mathbf{s}) \\
&= \sum_{\mathbf{s}} P(\mathbf{s}) \sum_{(t,u,y)\in\mathbf{s}} \log Q_y(t,u) \qquad \text{(from Eq 3)} \\
&= \sum_{(t,u,y)} \sum_{\mathbf{s}:(t,u,y)\in\mathbf{s}} P(\mathbf{s}) \log Q_y(t,u) \\
&= \sum_{(t,u,y)} P((t,u,y)) \log Q_y(t,u) \qquad (6)
\end{aligned}$$

where $P((t,u,y))$ denotes the marginal probability of segment (t,u,y) in our extraction model P. Now, for each y, the inner sum is the KL distance of Q_y from $P(.,.,y)$ (up to a constant involving only P). Thus, we can minimize the overall objective by setting

$$Q_y(t,u) = P((t,u,y)) \quad (7)$$

Figure 5 contains an example of optimal one-row parameters for the given segmentation set. $P((t,u,y))$ can be directly computed without enumerating any segmentations as we show next.

3.1.1 *Computing marginals from model* $P(\mathbf{s})$

In this section we show how to compute the marginal probabilities $P((t,u,y))$ of a segment from the original model.

Recall from Section 2 that the probability of a segmentation $\mathbf{s}$ in P is given as

$$\Pr(\mathbf{s}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \mathbf{exp}(\Lambda \cdot \sum_{\mathbf{j}} \mathbf{f}(j,\mathbf{x},\mathbf{s}))$$

The marginal probability of a segment (t,u,y) is the sum of the probability $P(\mathbf{s})$ of all segmentation $\mathbf{s}$ that contain this segment. Thus,

$$P((t,u,y)) = \sum_{\mathbf{s}:(t,u,y)\in\mathbf{s}} \frac{1}{Z(\mathbf{x})} \mathbf{exp}(\Lambda \cdot \sum_{\mathbf{j}} \mathbf{f}(\mathbf{x},\mathbf{s}))$$

The outer summation could be over exponential number of segmentations that contain a segment (t,u,y). Fortunately, the form of the distribution P can be exploited to compute this summation efficiently through a linear time forward pass followed by a backward pass. This is a standard forward-backward message passing algorithm [26] and can be skipped on first reading.

Forward pass:
Define $\alpha_i(y)$ as the value of $\sum_{\mathbf{s}'\in\mathbf{s}_{i:y}} \mathbf{exp}(\Lambda \cdot \sum_{\mathbf{j}} \mathbf{f}(j,\mathbf{s}',\mathbf{x}))$ where $\mathbf{s}_{i:y}$ denotes all segmentations from 1 to i ending at i and labeled y. The value of $\alpha_i(y)$ is computed recursively as

$$\alpha_i(y) = \sum_{i'=i-L+1}^{i} \sum_{y'\in Y} \alpha_{i'-1}(y') \mathbf{exp}(\Lambda \cdot \mathbf{f}(y,y',\mathbf{x},i',i))$$

where L = maximum segment length and the base case value $\alpha_1(y) = 1$ for all y. This gives, $Z(\mathbf{x}) = \sum_y \alpha_{|\mathbf{x}|}(y)$.

Backward pass:
In this pass we compute a set of n beta terms recursively as follows

$$\beta_i(y) = \sum_{i'=i+1}^{\min(i+L,n)} \sum_{y'\in Y} \beta_{i'}(y') \mathbf{exp}(\Lambda \cdot \mathbf{f}(y',y,\mathbf{x},i+1,i'))$$

The base case value $\beta_n(y) = 1$ for all y. Thus we need to compute nK α and β terms one for each position,label combination and the entire computation takes $O(nLK^2)$ time.

Computing marginals:
Once the α_i and β_i terms are available for each position of the sequence, we can compute the marginal probability of any segment (t,u,y) as follows:

$$P((t,u,y)) = \frac{\beta_u(y)}{Z(\mathbf{x})} \sum_{y'} \alpha_{t-1}(y') \mathbf{exp}(\Lambda \cdot \mathbf{f}(y,y',\mathbf{x},t,u)) \quad (8)$$

The marginals as computed above however have one limitation; they do not enforce the constraint we have in our extraction task that a label y can appear only once in an entire segmentation. It can be shown that any algorithm that exactly enforces such a constraint would be exponential in the number of labels. A number of algorithms exist in the graphical modeling literature for approximating the calculation of marginals in complex models [30, 9, 19]. We skip details of the approximate method in this paper because it is not crucial to the scope of this paper and the details are somewhat involved.

3.2 Multi-row model

In the multi-row model, we will consider multiple, but fixed (say m) number of rows, each one of which is a one-row model. The parameters of this model are the m row probabilities π_k and for each row k, the multinomial distribution parameters $Q_y^k(t,u)$ for each label y. We need to find the values of these parameters to minimize KL distance with $P(\mathbf{s})$

which is equivalent to maximizing the following objective.

$$\max_{\pi_k, Q^k} \sum_{\mathbf{s}} P(\mathbf{s}) \log \sum_{k=1}^{m} \pi_k Q^k(\mathbf{s}) \quad (9)$$

where $Q^k(\mathbf{s}) = \prod_{(t,u,y)\in\mathbf{s}} Q^k_y(t,u)$. Unlike in the one-row model, we cannot obtain the optimal parameter values in closed form because of the summation within the log. However, for the case where we can enumerate individual segmentations $\mathbf{s}$, this objective reduces to a well-known problem in statistics of learning a mixture model [13]. We will first discuss this enumeration-based approach and then present our algorithm that works directly on the model.

3.3 Enumeration-based approach

Let $\mathcal{S} = \mathbf{s}_1, \ldots \mathbf{s}_D$ be an enumeration of all segmentations produced from our original model $P(\mathbf{s})$. Our objective is

$$\max_{\pi_k, Q^k} \sum_{d=1}^{D} P(\mathbf{s}_d) \log \sum_{k=1}^{m} \pi_k Q^k(\mathbf{s}_d)$$

This is a well-known problem of learning the parameters of a mixture model and is solved using the classical Expectation-Maximization (EM) algorithm [13]. The algorithm starts with an initial guess of the parameter values and then iteratively performs the following two steps:

E step where for each $\mathbf{s}_d$ we find the probability $R(k|\mathbf{s}_d)$ of its belonging to component k given the current set of parameters

$$R(k|\mathbf{s}_d) = \frac{\pi_k Q^k(\mathbf{s}_d)}{\sum_i \pi_i Q^i(\mathbf{s}_d)}$$

This step can be thought of as a soft assignment of segmentations to each of the m components.

M step where we update model parameters using the above soft assignments to calculate the most likely values of parameters as follows:

$$Q^k_y(t,u) = \frac{\sum_{\mathbf{s}_d\in\mathcal{S}:(t,u,y)\in\mathbf{s}_d} P(\mathbf{s}_d) R(k|\mathbf{s}_d)}{\sum_{\mathbf{s}_d\in\mathcal{S}} P(\mathbf{s}_d) R(k|\mathbf{s}_d)}$$

$$\pi_k = \sum_{\mathbf{s}_d\in\mathcal{S}} P(\mathbf{s}_d) R(k|\mathbf{s}_d)$$

The algorithm iteratively updates parameters as above until convergence. It can be proved that in each iteration the EM algorithm improves the value of the objective and at convergence finds a locally optimal value of the parameters [13]. Although the algorithm can get stuck at local optima and is sensitive to the initial seeds, we have empirically found that for low dimensional data (e.g. $|Y| = 3--7$ dimensions in our case) and with multiple randomly seeded values of the initial soft assignment $R(k|\mathbf{s})$, the EM algorithm almost always reaches the global optimum in a few number of iterations. Since each iteration of the EM algorithm improves (a lower bound of) the objective function, so modulo any local optima, this scheme provides a good yardstick to benchmark any other algorithm.

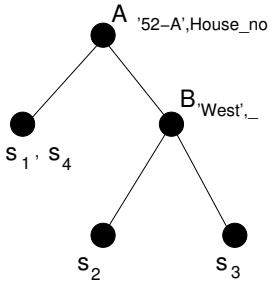


Figure 7: Example of a three-way partitioning using informative variables.

3.4 Structural approach

We now present our approach to generate multi-row models that does not entail enumeration of segmentations. The components corresponding to the rows in this case are assumed to cover disjoint sets of segmentations. We will show in Section 3.5 how to remove this limitation. Thus, for any segmentation exactly one of the rows has a non-zero probability. We achieve this disjointness by choosing a set C of boolean variables that form a binary decision tree with m leaves. Each segmentation satisfies exactly one of the paths in the tree. Our revised objective (Equation 10) with disjoint Q^k components can be written as

$$\max \sum_{k=1}^{m} \Pr(C_k) \sum_{\mathbf{s}} P(\mathbf{s}|C_k) \log \pi_k Q^k(\mathbf{s})$$

where we have removed the summation within the log by using the fact that a segmentation $\mathbf{s}$ that satisfies condition C_k will have a non-zero probability of generation only from Q^k.

Since each Q^k is equivalent to our one-row model, we can use Equation 6 to rewrite our objective as:

$$\max \sum_{k=1}^{m} \Pr(C_k)(\log \pi_k + \sum_{(t,u,y)} P(t,u,y|C_k) \log Q^k_y(t,u)) \quad (10)$$

For a given choice of C, this is maximized by choosing $Q^k_y(t,u) = P(t,u,y|C_k)$ and $\pi_k = \Pr(C_k)$. Thus, our remaining goal is to choose a partitioning C so as to maximize the objective.

$$\sum_{k=1}^{m} \Pr(C_k)(\log P(C_k) + \sum_{(t,u,y)} P(t,u,y|C_k) \log P(t,u,y|C_k)) \quad (11)$$

We achieve the m-way partitioning by sequentially choosing binary split variables to grow a tree of variables until we reach m leaves, each of which corresponds to our desired partition. Our variables correspond to various kinds of conditions on segmentations. We consider three kinds of boolean variables:

- $\mathcal{A} = \{A_{tuy} | t \geq u, y \in Y \cup \{'\_'\}\}$: Variable A_{tuy} is one if a non-NULL segment (t,u) is present and has a label y. If $y =' \_'$ then the segment's label is not conditioned on.

- $\mathcal{B} = \{B_{ty} | y \in Y \cup \{'\_'\}\}$: Variable B_{ty} is one if a segment starts at position t with label y or with no condition on label when $y =' \_'$. In this case the end boundary of the segment is not conditioned on.

- $\mathcal{C} = \{C_{uy} | y \in Y \cup \{'\_'\}\}$ Variable C_{uy} is one if a segment ends at position u. The start boundary of the segment is not conditioned on. The segment label is ignored if $y =' \_'$.

Figure 7 illustrates a three-way tree on the example string in Figure 3, created using these variables. The root variable $A_{52-A,House\_no}$ is true iff the segment '52-A' is labeled 'House_no'. Similarly $B_{West,\_}$ is true if any segment starts with the word 'West'. Here, a label is not part of the condition.

Let c denote a sequence of decision tree nodes from root to leaf, For a given condition c, let

$$H(c) = \Pr(c)(\log P(c) + \sum_{(t,u,y)} P(t,u,y|c) \log P(t,u,y|c))$$

Thus our objective from Equation 11 is $\sum_k H(C_k)$. At any stage when we choose to split node c using another variable $v \in \mathcal{A} \cup \mathcal{B} \cup \mathcal{C}$ we pick v so as to maximize the gain $I(v,c)$ of v given condition c as defined below

$$I(v,c) = H(c, v=1) + H(c, v=0) - H(c) \tag{12}$$

At each stage we pick the leaf node c from the tree and a corresponding variable v so as to get the highest gain. We stop when we have obtained the desired m number of leaf nodes.

$H(c)$ can be computed directly from model $P(\mathbf{s})$ without any enumeration using a slight variant of the algorithm presented in 3.1.1 for computing the conditional marginals $P(t,u,y|c)$ and $P(c)$. The only change is that in the forward and backward pass we disallow any segments that violate condition c. From these we can compute the conditional marginals $P(t,u,y|c)$ and the value of $Z(\mathbf{x}|c)$ analogously to the unconditional case as $Z(\mathbf{x}|c) = \sum_y \alpha(|\mathbf{x}|, y|c)$. From these $P(c)$ is available as

$$P(c) = \frac{Z(\mathbf{x}|c)}{Z(\mathbf{x})} \tag{13}$$

The above algorithm achieves the maximum gain in objective at each stage, and thus can be considered a greedy algorithm like most decision tree construction algorithms. It is possible to design an optimal algorithm that runs in $O(n^3 m^2 K^4)$ time where n is the length of the sequence and K is the number of labels. In contrast the greedy algorithm here is $O(mn^2K)$ which is practical since typically n, the number of words in an unstructured record, is no more than 100.

3.5 Merging structures

The structural approach makes a disjoint partitioning of segmentations whereas the EM algorithm in the enumeration-based approach of Section 3.3 allows a segmentation to overlap with arbitrary number of mixtures; thus the structured approach is likely to be worse than the enumeration approach in terms of approximation quality.

In this section, we present a fix to this problem. In the structural approach we generate many more than the required m partitions. In practice, we can go on making partitions till the gain obtained by a partition split is negligible. Let m_0 denote the number of initial partitions obtained in this manner.

We view the one-row model of each partition as a compound segmentation. Now we can apply the EM-based approach to cluster these m_0 compound segmentations into m rows. Using the notation of Section 3.3, we will have $|\mathcal{S}| = m_0$ with $\mathbf{s}$ denoting a compound segmentation whose distribution we denote by $P()$. Each compound segmentation $\mathbf{s}$ output by the structural approach is associated with a condition $c_{\mathbf{s}}$ which is used to calculate a total probability $P(\mathbf{s})$ (using Equation 13), and, for each label y a multinomial distribution $P_y^{\mathbf{s}}(t,u)$ (using Equation 8 after conditioning the α, β and Z terms on $c_{\mathbf{s}}$). We will merge these m_0 $P()$ distributions to m $Q()$ distributions using the EM algorithm.

We now show how to apply the EM-algorithm for compound segmentations without enumerating the constituents of any one of them. For the M-step, to compute $Q_y^k(t,u)$, instead of summing over only those $\mathbf{s}$ that contain the segment (t,u,y), we probabilistically sum over all the $\mathbf{s}$, because the membership of a segment in a compound segmentation is now probabilistic instead of being hard. Thus,

$$Q_y^k(t,u) = \frac{\sum_{\mathbf{s}_d \in \mathcal{S}} P(\mathbf{s}_d) P_y^{\mathbf{s}_d}(t,u) R(k|\mathbf{s}_d)}{\sum_{\mathbf{s}_d \in \mathcal{S}} P(\mathbf{s}_d) R(k|\mathbf{s}_d)}$$

However, the E-step is challenging because it involves computing $Q^k(\mathbf{s})$, which is the probability of generating the distribution of the compound segmentation $\mathbf{s}$ from the distribution represented by the k^{th} row. Here we exploit two insights: (a) The columns of the row are independent, so the probability of generation can be decomposed over labels. (b) Each label defines a multinomial distribution over its possible segments, in both $P(\mathbf{s})$ as well as $Q^k(\mathbf{s})$. This allows us to use the method proposed in [2] which reduces the problem of finding the probability of generating one multinomial distribution from another to a measure of divergence between the two. This distance, which turns out to be KL divergence for our scenario, is a special case of the general Bregman divergence framework of [2]. In our case, the probability of generation, $Q^k(\mathbf{s})$, can be seen to be ([2]):

$$Q^k(\mathbf{s}) \propto \exp(-\sum_y KL(P_y(\mathbf{s}) || Q_y^k)) \tag{14}$$

where $P_y(\mathbf{s})$ is the multinomial distribution for the label y in the compound segmentation $\mathbf{s}$ and Q_y^k is its counterpart in the k^{th} component of the multi-row model.

Thus, it is possible to soft-merge the m_0 partitions into m clusters using the EM algorithm without any enumeration. Since each of the original m_0 partitions was represented by a conjunction of hidden variables, every component in the final m-row model is now a weighted disjunction of such conjunctions. This representation is clearly richer and more expressive than the original conjunctive form.

We will also present empirical evidence to show that this merging step causes a significant increase in our objective value and also compares favorably with the ideal EM solution that uses enumeration to estimate its model parameters. We will also show that in practice, m_0 is not large. In fact, we will show that m_0 is much smaller than the number of segmentations we would have retrieved, had we gone for an enumeration of the top-k segmentations.

Our final algorithm for finding the best m-row model without enumerating segmentations is outlined in Algorithm 1.

Algorithm 1 StructMerge(m, gainCutOff)

{Create partitions until gain < gainCutOff}
Initialize partition tree with a single node.
while tree has unvisited leaves **do**
  $c \leftarrow$ unvisited leaf.
  Mark c visited.
  Find variable $v \in \mathcal{A} \cup \mathcal{B} \cup \mathcal{C}$ with largest gain $I(c, v)$
  if (gain > gainCutOff) **then**
    Split c on $v = 1$ and $v = 0$ to create two leaves.

{Merge partitions using EM to get m rows}
paths $\leftarrow$ path to all leaves each of which defines a partition

for all s < paths.size() **do**
  $\forall u, y$: Compute $\alpha_u(y|s)$ and $\beta_u(y|s)$ via forward-backward algorithm restricted to segments that satisfy the condition of s
  $P[s] \leftarrow \frac{Z(\mathbf{x}|s)}{Z(\mathbf{x})}$ where $Z(\mathbf{x}|s) \leftarrow \sum_y \alpha_{|\mathbf{x}|}(y|s)$
  $\forall t, u, y$: Compute $P_y^s[t, u]$ via Equation 8
$\forall k < m, \forall s < paths.size()$:
  $Seed(R[k, s])$;
loop {Until convergence}
  M step
  $\forall k < m : \pi_k \leftarrow \sum_s P[s]R[k, s]$;
  $\forall k < m, y, t, u : \quad Q_y^k[t, u] \leftarrow \frac{\sum_s P[s]R[k,s]P_y^s[t,u]}{\sum_s P[s]R[k,s]}$;
  E step
  $\forall k < m, s < paths.size()$:
    $R[k, s] \leftarrow \frac{\pi_k \exp(-\sum_y KL(P_y^s || Q_y^k))}{\sum_l \pi_l \exp(-\sum_y KL(P_y^s || Q_y^l))}$

4. EXPERIMENTS

We now present an empirical evaluation of our method of curating imprecise databases from statistical models of structure extraction. Since ours is the first work on this topic, we first present in Section 4.2 experiments to demonstrate that representing the imprecision of extraction is indeed useful in improving the quality of answers returned by the database. In other words, we show that the current practice of storing the topmost extraction in a conventional database can incur higher error than a database that captures extraction uncertainty. We next go on to evaluate in Section 4.3 the two primary methods of representing imprecision via row uncertainty and column uncertainty and show that the multi-row representation that combines both forms of uncertainty provides better approximation for a fixed number of stored parameters. Finally we present results of evaluating our core contribution in the paper — the multi-row parameter approximation algorithm. We evaluate the algorithm both in terms of the quality of approximation and efficiency of execution.

Dataset	Model	Number of labels in query 1	2	3	4
Address (Weak)	Top-1	0.31	0.48	0.59	0.66
	Top-k	0.22	0.32	0.35	0.37
Address (Strong)	Top-1	0.10	0.17	0.22	0.26
	Top-k	0.08	0.14	0.17	0.19
Cora (Weak)	Top-1	0.16	0.29	0.38	0.45
	Top-k	0.10	0.17	0.22	0.25
Cora (Strong)	Top-1	0.07	0.13	0.18	0.21
	Top-k	0.05	0.09	0.12	0.14

Table 1: Square errors on projection queries of varying size over the top-1 and top-k models.

4.1 Datasets

We used the following two real-life datasets for our experiments.

Cora: Cora is a popular citations benchmark [22] that consists of 500 citations collected from the reference section of several academic papers. The columns of the corresponding citation table were paper title, author-list, booktitle, journal, page number, date, and volume. We use a subset of 125 citations as our test-set.

Address: The Address dataset consists of 770 home addresses of students in IIT Bombay India. The table columns were Street names, area, city name, state name and zipcode. Indian addresses are less regular than US addresses, and extracting even fields like city names is challenging [4]. A subset of 440 addresses was used as the test-set.

For each dataset, we train a strong and a weak CRF model by varying the amount of training data. To train the strong model, we use 30% of the data for Cora and 50% for Address. To train both the weak models, we use 10% of the data. See [25] for details of features used in training these models.

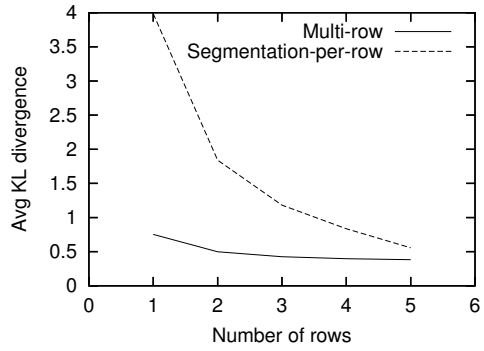
In each experiment, we retrieve enough segmentations to cover a sufficient probability mass (0.92 in the case of Cora and 0.96 for the Address dataset). This forms our test set and the divergence of a model is computed over this test set.

4.2 Error reduction with probabilistic data

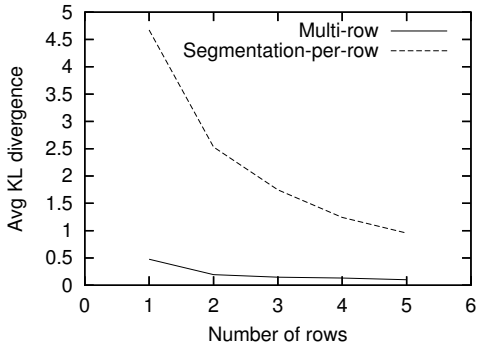
We demonstrate the usefulness of capturing the uncertainties of extraction by showing that even for an imperfect extraction model, the current practice of storing the top-most extraction result can incur higher error than a probabilistic query processing system. Our queries project various subsets of the column labels and we measure the error in the query results with respect to the correct segmentation.

We report the square error, which is $(1 - p^*)^2$, where p^* is the probability of the correct answer according to the stored model.

In Table 1 we show the errors for two cases: First, Top-1, where we store the single best extraction. In this case if



(a) Cora (Weak)



(b) Address (Weak)

Figure 8: Comparing divergence of multi-row models with increasing m, against Segmentation-per-row model with the same number of parameters.

the single stored segmentation is correct, we get an error of 0, otherwise it is 1. Second, Top-k, where we store *all* segmentations with their probabilities so as to capture the Semi-CRF distribution exactly. This allows us to compare the best possible imprecise model with the current practice of storing extractions. We report errors for strongly as well as weakly trained extraction models.

Table 1 shows a significant drop in error from Top-1 to Top-k. This establishes that representing uncertainty in the database leads to more accurate query results and as we will see, even a simple two row approximation can achieve the same reduction in error as the ideal case of representing the entire distribution.

4.3 Comparing models of imprecision

We study the quality of approximation with increasing amounts of storage used for representing imprecision in the database. In Figure 8 we compare the approximation quality of a multi-row model over m rows with a segmentation-per-row model where we store top-k segmentations with k chosen so that both approaches store the same number of parameters.

We observe that the multi-row model has a significantly smaller divergence from the true model compared to the segmentation-per-row model when both are constrained to use the same number of parameters. For example for the Address dataset, with 2 rows the multi-row model can achieve a close to 0 divergence with the true model whereas the Segmentation-per-row approach using equivalent number of parameters (which amounts to roughly 4 rows) has a divergence of around 2.5.

The case of $m = 1$ corresponds to the one-row model and we observe that going from one to two rows results in more than a factor of two reduction in divergence and increasing m further does not cause as dramatic a decrease.

These experiments show that the multi-row model that combines both row-level and column-level uncertainty provides an effective mechanism of capturing the uncertainty of an otherwise complex extraction model.

4.4 Evaluating the multi-row algorithm

In this section, we compare our multi-row algorithm (structural+merging) with the enumeration-based approach and the simpler structural-only approach. Figure 9 plots the KL distances obtained by the structural approach, structural+merging approach (with a gain cutoff of 0.5) and the enumeration approach all for $m = 2$. For reference we also show the one-row model. For the structural+merging approach, as well as the enumeration approach, the EM output was chosen by executing five different randomly seeded runs and selecting the one with the least KL distance over the test-set. The figures are plotted by sorting the data in increasing order of KL distances obtained by the enumeration algorithm.

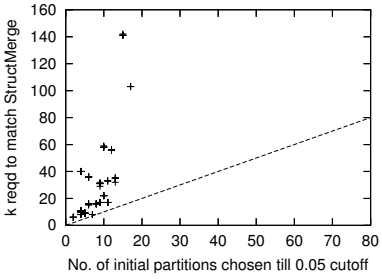
We observe from the graphs that the structural+merging approach is better than the structural approach and very close to the ideal enumeration approach.

The enumeration approach however runs EM over significantly more segmentations than the compound segmentations m_0 generated in the structural+merging approach as we show in the next experiment.

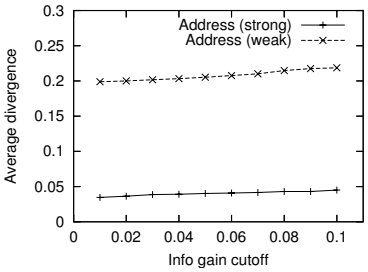
For each multi-row model generated by the structural+merging approach, we record the number m_0 of initial partitions and then find the smallest k for which the enumeration approach using top-k segmentations would have smaller KL distance. Figure 10 displays the variation of k with m_0 for the Address dataset. We also display the $y = x$ line to point out that k is always more than m_0 and sometimes the gap is very large. This means that the structural+merging approach achieves almost the same approximation quality as the enumeration approach while computing a much fewer number of parameters.

Dependence on input parameters

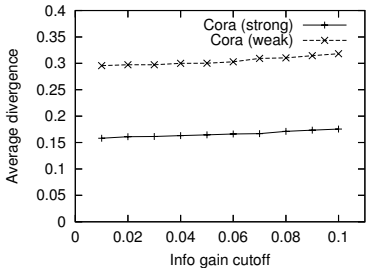
The two input parameters to the structural+merging approach are the number of rows m, and the gain cutoff ϵ. We already studied the effect of m on the quality of approximation in Section 4.3. In general, m is easy to determine based on the quality of approximation achieved for each m. The choice of the gain cutoff is less direct but as we show



(a) Address (Weak)



(a) Address



(b) Cora

Figure 10: Variation of k with m_0. ($\epsilon = 0.05$ and $m = 2$).

Figure 11: Dependence of the KL distance obtained via the structural+merging approach on gain cutoff. m is set to 2.

in Figure 11 the gain cutoff is easy to determine for two reasons: first, there is a wide range of values where the approximation quality does not change much and second, the gain cutoff is an absolute quantity that measures properties of distributions and is not dependent on particular datasets, as we also observe in Figures 11(a) and 11(b).

Impact on query results

So far we have seen that a low KL divergence is good from the perspective of approximating the extraction model. However from a practitioner's point of view, a model is good iff it impacts his query results. Here, we demonstrate that the structural+merging approach is much better than the one-row approach in terms of preserving the (probability sorted) rank order of query results. For this purpose, we constructed projection queries over four randomly chosen columns of the Address dataset. We then compared the 'golden' order obtained by sorting on the exact probabilities of the segmentation-per-row approach with the approximate probabilities of the one-row approach and the structural+merging approach. We retrieved top-10 results for each string and for each pair in this 'golden' ordering, we checked whether the order between the two result rows was preserved in the one-row and structural+merging models. In Figure 12(a), we report the fraction of pairs whose ranking was inverted. Figure 12(a) shows that the one-row model keeps the ordering unchanged for only 8% of the cases whereas the structural+merging approach keeps the ordering unchanged for about 25% of the cases.

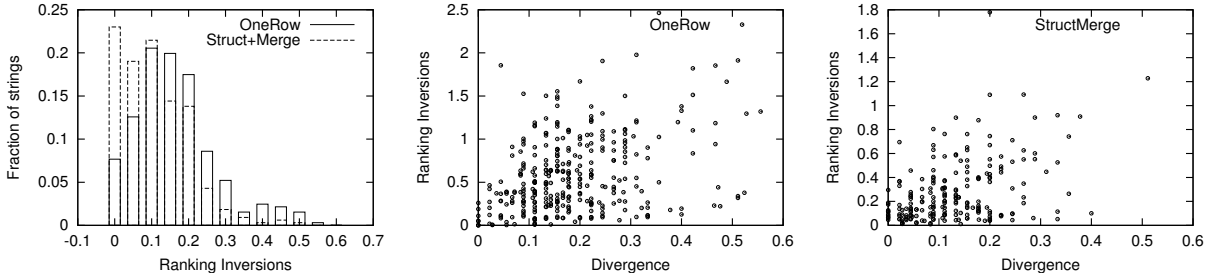
While we have established the practical advantage of using the structural+merging approach, it is necessary that our original objective of minimizing KL divergence also translate to minimizing ranking inversions. Figures 12(b) and 12(c) illustrate that this is indeed the case for one-row as well as structural+merging. The scatter plots show a strong correlation between KL divergence and the inversion scores. Moreover, the scatter plot of the structural+merging approach is concentrated in the bottom-left corner, thus reaffirming that this approach is highly suitable for both the criteria.

5. RELATED WORK

The work presented in this paper is related to work in two disconnected communities; management of imprecise information from databases and approximation of statistical models from machine learning.

While the topic of imprecise databases has been of constant interest to database researchers [15, 3], a recent revival [27, 17, 12, 11, 24, 8, 6] has lead to better consolidation and more insights on the representation and processing of uncertain data. The focus mostly has been on designing generic models of imprecision and processing queries over them efficiently. There has been little thought on how to populate imprecise data with sound probabilistic quantification in the first place. Admittedly, much of this is application-specific. However, even for popular applications of imprecise data, like sensor data, there is surprisingly little work on matching an external, more powerful probability distribution to database models of imprecision. Existing work on approximate representation in the imprecise database literature [12] focus primarily on converting from one database model of uncertainty to another without any probabilistic underpinning, whereas our focus is on converting an external,complex probability distribution to a probabilistic database model. The latter presents a very different set of challenges than the former. To the best of our knowledge, ours is the first paper on transforming the uncertainty of complicated, yet high performing statistical models of extraction to tractable models of imprecision that can be easily stored and queried in a database.

The CRF model for information extraction is a special case of probabilistic graphical models. The problem of approximating a complex graphical model $P(\mathbf{x})$ to a simpler model $Q(\mathbf{x})$ for faster inference is an actively researched problem in machine learning [9, 30, 19, 18]. However, the techniques there are not directly applicable to our case because of two reasons. First, most work there assumes that the original $P(\mathbf{x})$ distribution is too complicated even for computing marginal distributions. They address this problem by changing their objective to minimize $KL(Q||P)$ instead of the desired objective of minimizing $KL(P||Q)$. Note that KL is an asymmetric measure of divergence and when P is the starting distribution it is more meaningful to minimize divergence with P than with Q. The segmentation



(a) Histogram of ranking inversions

(b) Scatter plot for One-Row

(c) Scatter plot for StructMerge

Figure 12: Figure 12(a) plots the inversion scores of the one-row and structural+merging models on the Address (Weak) dataset. Figures 12(b) and 12(c) plot the correlation between KL divergence and inversion scores. For the structural+merging approach, $m = 2$ and $\epsilon = 0.005$.

model used in this paper is more tractable and thus we can solve the original objective efficiently. Second, approximating using mixture model $Q(\mathbf{x})$ raises peculiar optimization challenges not present in the more extensively studied case of $Q(\mathbf{x})$ having a subset of the dependencies in $P(\mathbf{x})$. There is surprisingly little work on approximating to a mixture model and what exists solves the less direct objective of minimizing $KL(Q||P)$ [18].

6. CONCLUSIONS

In this paper we presented a method of curating imprecise databases from statistical models of structure extraction. We investigated three models of representing imprecision in a database: a segmentation-per-row approach that only allows row-level uncertainty, a one-row model that allows only column-level uncertainty and a multi-row model that allows both row and column-level uncertainty. We showed that the multi-row model provides significantly better approximation for the number of parameters it uses. We designed algorithms for finding such parameters efficiently without requiring an enumeration of all possible segmentations from the source model. Empirical results on real-life datasets show that our algorithm achieves very close to the ideal KL distance and does so with much fewer partitions than segmentations needed by direct enumeration.

There is much scope for future work in this topic, including handling of multi-table imprecision, extending imprecision to not just extraction but also integration, and designing fast algorithms in each case.

Acknowledgements

We would like to thank the anonymous reviewers for their valuable comments. The work reported here was supported by research grants from Microsoft Research and an IBM Faculty award. We thank them for their generous support.

7. REFERENCES

[1] E. Agichtein and V. Ganti. Mining reference tables for automatic text segmentation. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, USA*, 2004.

[2] A. Banerjee, S. Merugu, I. S. Dhillon, and J. Ghosh. Clustering with bregman divergences. *Journal of Machine Learning Research (JMLR)*, 6:1705–1749, Oct 2005.

[3] D. Barbar, H. Garcia-Molina, and D. Porter. The management of probabilistic data. *IEEE Transactions on Knowledge and Data Engineering*, 4(5):487–502, 1992.

[4] V. R. Borkar, K. Deshmukh, and S. Sarawagi. Automatic text segmentation for extracting structured records. In *Proc. ACM SIGMOD International Conf. on Management of Data*, Santa Barabara,USA, 2001.

[5] J. Boulos, N. Dalvi, B. Mandhani, S. Mathur, C. Re, and D. Suciu. Mystiq: a system for finding more answers by using probabilities. In *ACM SIGMOD*, 2005.

[6] D. Burdick, P. M. Deshpande, T. S. Jayram, R. Ramakrishnan, and S. Vaithyanathan. OLAP over uncertain and imprecise data. In *VLDB '05: Proceedings of the 31st international conference on Very large data bases*, pages 970–981. VLDB Endowment, 2005.

[7] M. E. Califf and R. J. Mooney. Relational learning of pattern-match rules for information extraction. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99)*, pages 328–334, July 1999.

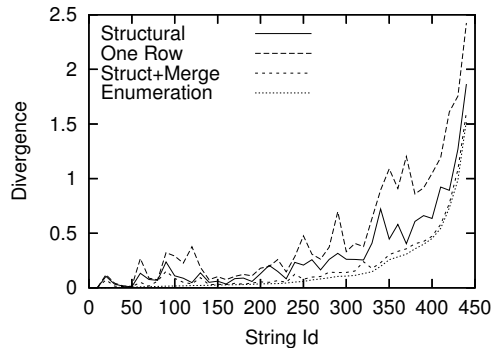
[8] R. Cheng, D. V. Kalashnikov, and S. Prabhakar. Evaluating probabilistic queries over imprecise data. In *SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 551–562, New York, NY, USA, 2003. ACM Press.

[9] C. K. Chow and C. N. Liu. Approximating discrete probability distributions with dependence trees. In *IEEE Transactions on Information Theory IT-14*, volume 3, pages 462–467, 1968.

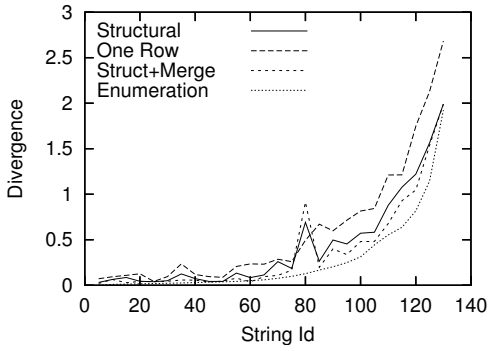
[10] W. W. Cohen and S. Sarawagi. Exploiting dictionaries in named entity extraction: Combining semi-markov extraction processes and data integration methods. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, USA*, 2004.

[11] N. N. Dalvi and D. Suciu. Efficient query evaluation on probabilistic databases. In *VLDB*, pages 864–875, 2004.

[12] A. Das Sarma, S. U. Nabar, and J. Widom. Representing uncertain data: Uniqueness, equivalence, minimization, and approximation. Technical report, Stanford University, 2005.



(a) Address (Weak)



(b) Cora (Weak)

Figure 9: Comparison of KL distances of various approaches. The strings are sorted by their KL distance as given by the enumeration approach.

[13] A. P. Dempster, N. M. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, 1(39):1–38, 1977.

[14] D. Dey and S. Sarkar. A probabilistic relational model and algebra. *ACM TODS*, 21(3):339–369, 1996.

[15] N. Fuhr. A probabilistic framework for vague queries and imprecise information in databases. In *Proceedings of the sixteenth international conference on Very large databases*, pages 696–707, San Francisco, CA, USA, 1990. Morgan Kaufmann Publishers Inc.

[16] T. J. Green and V. Tannen. Models for incomplete and probabilistic information. *IEEE Data Engineering Bulletin*, 29(1), March 2006.

[17] E. Hung, L. Getoor, and V. S. Subrahmanian. Pxml: A probabilistic semistructured data model and algebra. In *ICDE*, 2003.

[18] T. S. Jaakkola and M. I. Jordan. Improving the mean field approximation via the use of mixture distributions. In M. I. Jordan, editor, *Learning in Graphical Models*. MIT Press, 1999.

[19] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. Saul. An introduction to variational methods for graphical models. In M. I. Jordan, editor, *Learning in Graphical Models*. MIT Press, 1999.

[20] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning (ICML-2001)*, Williams, MA, 2001.

[21] L. V. S. Lakshmanan, N. Leone, R. Ross, and V. S. Subrahmanian. Probview: a flexible probabilistic database system. *ACM TODS*, 22(3):419–469, 1997.

[22] F. Peng and A. McCallum. Accurate information extraction from research papers using conditional random fields. In *HLT-NAACL*, pages 329–336, 2004.

[23] A. Ratnaparkhi. Learning to parse natural language with maximum entropy models. *Machine Learning*, 34, 1999.

[24] R. Ross, V. S. Subrahmanian, and J. Grant. Aggregate operators in probabilistic databases. *J. ACM*, 52(1):54–101, 2005.

[25] S. Sarawagi. Efficient inference on sequence segmentation models. In *Proceedings of the 23rd International Conference on Machine Learning (ICML), Pittsburgh, PA, USA*, 2006.

[26] S. Sarawagi and W. W. Cohen. Semi-markov conditional random fields for information extraction. In *NIPS*, 2004.

[27] A. D. Sarma, O. Benjelloun, A. Halevy, and J. Widom. Working models for uncertain data. In *ICDE*, 2006.

[28] K. Seymore, A. McCallum, and R. Rosenfeld. Learning Hidden Markov Model structure for information extraction. In *Papers from the AAAI-99 Workshop on Machine Learning for Information Extraction*, pages 37–42, 1999.

[29] F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *In Proceedings of HLT-NAACL*, 2003.

[30] M. J. Wainwright, T. Jaakkola, and A. S. Willsky. Tree-based reparameterization framework for analysis of sum-product and related algorithms. *IEEE Transactions on Information Theory*, 45(9):1120–1146, 2001.