Uncertain Data Queries Processing in a Probabilistic Framework

Ming He
College of Computer Science, Beijing University of Technology, Beijing
Email: heming@bjut.edu.cn

Yong-ping Du
College of Computer Science, Beijing University of Technology, Beijing
Email: ypdu@bjut.edu.cn

***Abstract*—Many applications today need to manage data that is uncertain, such as information extraction (IE), data integration, sensor RFID networks, and scientific experiments. Top-k queries are often natural and useful in analyzing uncertain data in those applications. In this paper, we study the problem of answering top-k queries in a probabilistic framework from a state-of-the-art statistical IE model-semi-Conditional Random Fields (CRFs)-in the setting of Probabilistic Databases that treat statistical models as first-class data objects. We investigate the problem of ranking the answers to Probabilistic Databases query. We present efficient algorithm for finding the best approximating parameters in such a framework to efficiently retrieve the top-k ranked results. An empirical study using real data sets demonstrates the effectiveness of probabilistic top-k queries and the efficiency of our method.**

***Index Terms*—probabilistic databases, uncertain data, information extraction, conditional random fields**

I. INTRODUCTION

In recent years, uncertain data management arises in many applications. The reasons for uncertainty in data are as various as the applications themselves: in sensor and RFID data, uncertainty as a result of measurement errors [1, 2]; in information extraction, uncertainty comes from the inherent ambiguity in natural-language text [3, 4]; and in business intelligence, uncertainty is used to decrease the cost of data cleaning [5]. The field of uncertain data management poses a number of unique challenges on several fronts. The two broad issues are those of modeling the uncertain data, and then leveraging it to work with a variety of applications. The second issue is that of adapting data management and mining applications to work with the uncertain data. The main areas of research in the field are including modeling of uncertain data, uncertain data management and uncertain data mining [6]. Unfortunately, traditional precise database management systems (DBMSs) do not support uncertain data due to data records are typically represented by probability distributions rather than deterministic values. Hence, it is necessary to develop data management techniques for managing probabilistic data.

The goal of Information Extraction is to extract structured data from a collection of unstructured text documents. Usually the schema is given in advance by the user, and the extractor is tailored to that specific schema [7]. In information extraction, imprecision comes from the inherent ambiguity in natural-language text. Automatically extracting structured entities from unstructured text is a challenging problem, and has a history of attempts spanning early rule-based systems like Rapier [8] to the lately statistical methods like Conditional Random Fields (CRFs) [3]. Gupta and Sarawagi [9] recently describe how to use a probabilistic database to store the result of text segmentation with Conditional Random Fields.

In this paper we introduce a state-of-art method called semi-CRFs for information extraction and provide a probabilistic framework enabling opportunities for top-k queries. Our technique is based on two major observations: First, All approaches to information extraction are imprecise, and most often can associate a probability score to item extracted. In the database community, information extraction has been a major motivating application for the recent groundswell of work on probabilistic database (PDB), which can model the uncertainty inherent in information extraction outputs, and enable users to get probabilistic answers. Second, top-k queries are often natural and useful in analyzing uncertain data, and CRFs can be very naturally modeled as first-class data in a relational database, in the spirit of recent PDB like BayesStore [10] and the work of Sen and Deshpande [11]. Similarly, text data can be captured relationally via the inverted file representation commonly used in information retrieval.

This paper is organized as follows: In Section 2, we will examine the issue of uncertain data representation and modeling. In Section 3, we describe our problem and present a background of state of the art probabilistic models of information extraction. In section 4, we describe the design of a probabilistic database that can support probabilistic information extraction models, and a framework is presented for inference queries. In Section 5 we present experimental evaluation of the accuracy and

doi:10.4304/jcp.5.11.1663-1669

efficiency of our model. The conclusions and future work are discussed in Section 6.

II. UNCERTAIN DATA REPRESENTATION AND MODELING

A. Probabilistic Database Definitions

A probabilistic database is defined [12] as follows:

Definition1. A *probabilistic-information database* is a finite probability space whose outcomes are all possible database instances consistent with a given schema. This can be represented as the pair (*X, p*), where *X* is a finite set of possible database instances consistent with a given schema, and *p* (*I*) is the probability associated with any instance $I \in X$. We note that since *p* (·) represents the probability vector over all instances in *X*, we have $\sum_{I \in X} p(I) = 1$.This representation is a formalism of the "possible world model" [13]. To represent the uncertainties in data, some approaches associate the uncertainties to tuples, while others associate them to values.

Probabilistic ?-tables [14], [15] are a simple way of representing probabilistic data. In this case, one models the probability that a particular tuple is present in the database. Thus, the probability of a particular instantiation of the database can be defined as the product of the probabilities of the corresponding set of tuples to be present in the database with the product of the probabilities of the complementary set of tuples to be absent from the database.

A closely related probabilistic representation is that of probabilistic or-set tables. While the probabilistic ?-table is concerned with the presence or absence of a particular tuple, the p-or-set table is concerned with modeling the probabilistic behavior of each attribute for a tuple that is known to be present in the database. In this case, each attribute is represented as an "or" over various possibilities along with corresponding probability values. An instantiation of the database is constructed by picking each outcome for an attribute independently. The ProbView model presented in [13] is a kind of or-set table. The only significant difference is that the ProbView model uses confidence values instead of probabilities.

III. MODELS FOR INFORMATION EXTRACTION

A Conditional Random Fields (CRFs) models a single joint distribution **Pr** (**y**| **x**) over the predicted labels **y** = $y_1 \cdots y_n$ of the tokens of **x**=$x_1 \cdots x_n$. The tractability of the joint distribution is ensured by using a Markov random field to express the conditional independencies that hold between elements y_i of **y**. In typical extraction tasks, a chain is adequate for capturing label dependencies. This implies that the label y_i of the *i*th token is directly influenced only by the labels of tokens that are adjacent to it. In other words, once the label y_{i-1} is fixed, label y_{i-2} has no influence on label y_i.

The dependency between the labels of adjacent tokens is captured by a scoring function (yi_{-1}, yi, **x**, *i*) between nodes y_{i-1} and y_i. This score is defined in terms of weighted functions of features as follows:

$$\psi(y_i, \mathbf{x}, i, y_{i-1}) = exp^{\sum_{k=1}^{K} \omega_k f_k(y_i, \mathbf{x}, i, y_{i-1})} = exp^{\mathbf{W} \cdot \mathbf{f}(y_i, \mathbf{x}, i, y_{i-1})} \quad (1)$$

Where **w** is a weight vector for **f** that is learnt during training via a variety of methods. With these per-edge scores, the conditional distribution of a label sequence **y** given a token **x** is given as

$$\mathbf{Pr}(\mathbf{y} \mid \mathbf{x}, \mathbf{w}) = \frac{1}{Z(\mathbf{x})} \prod_{i=1}^{n} \psi(y_{i-1}, y_i, \mathbf{x}, i) = \frac{1}{Z(\mathbf{x})} exp^{\sum_{i=1}^{n} \mathbf{W} \cdot \mathbf{f}(y_i, \mathbf{x}, i, y_{i-1})} \quad (2)$$

The term Z(**x**) is a normalization factor and is equal to $\sum_{y} exp^{\mathbf{w} \cdot \mathbf{f}(\mathbf{x}, \mathbf{y})}$, where $f(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} \mathbf{f}(y_i, x, i, y_{i-1})$ the sum of the feature vector over all token positions of the sequence. Some subset of these features can be simplified further to depend only on the current state and are independent of the previous state. We will refer these as state features and denote these by *f* (y_i, **x**, *i*) when we want to make the distinction explicit. The term transition features refers to the remaining features that are not independent of the previous label.

To extend CRFs to semi-CRFs model, in which the output is a sequence of segments, with each segment defining an entity, rather than a sequence of labels as in earlier CRFs models. More formally, a segmentation **s** of an input sequence of length *n* is a sequence of segments $s_1 \cdots s_p$ such that the last segment ends at *n*, the first segment starts at 1, and segment s_j+1begins right after segment s_j ends. Each segment s_j consists of a start position l_j, an end position u_j , and a label $y_j \in \mathbf{y}$.

In a Semi-CRFs model, features are defined over segments comprising of multiple tokens forming an entire entity string. This allows for the use of more powerful features than token-level models because features can now capture joint properties over all the tokens forming part of an entity. Like in the case of sequence models, 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 = (y_j, l_j, u_j) is of the form *f* (y_j, y_{j-1}, **x**, l_j, u_j).

A Semi-CRF models the conditional probability distribution over segmentation **s** for given input sequence **x** as follows:

$$\mathbf{Pr}(\mathbf{s} \mid \mathbf{x}, \mathbf{w}) = \frac{1}{Z(\mathbf{x})} \prod_{i=1}^{n} \psi(y_{i-1}, y_i, \mathbf{x}, \mathbf{s}) = \frac{1}{Z(\mathbf{x})} exp^{\sum_{i=1}^{n} \mathbf{W} \cdot \mathbf{f}(y_i, \mathbf{x}, \mathbf{s}, y_{i-1})} \quad (3)$$

During extraction, the goal is to find a segmentation **s**=$s_1 \cdots s_p$ of the input sequence **x**=$x_1 \cdots x_n$ such that Pr(**s** | **x**, **w**) as defined by Equation 3 is maximized, which implies that

$$argmax_{\mathbf{s}} \mathbf{Pr}(\mathbf{s} \mid \mathbf{x}, \mathbf{w}) = argmax_{\mathbf{s}} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, \mathbf{s}) = argmax_{\mathbf{s}} \mathbf{w} \cdot \sum_{j} \mathbf{f}(y_i, y_{i-1}, \mathbf{x}, l_j, u_j) \quad (4)$$

The right hand side can be efficiently computed by dynamic programming. Let *L* be an upper bound on segment length. Let $\mathbf{s}_{i:y}$ denote set of all partial segmentation starting form 1(the first index of 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

$w \cdot \sum_j \mathbf{f}(j, \mathbf{x}, \mathbf{s}')$ for any $\mathbf{s}' \in \mathbf{s}_{i:y}$. The following recursive calculation implement finds the best segmentation:

$$V(i,y) = \begin{cases} max_{y',d=i-L\ldots i-1} V(d,y') + \mathbf{w} \cdot \mathbf{f}(y,y',x,d+1,i) & if\ (i>0) \\ 0 & if\ (i=0) \\ -\infty & if\ (i<0) \end{cases} \quad (5)$$

The algorithm makes a forward scan of the input tokens, and for each token position *i* and entity label y computes the best segmentation from 1 to *i* by taking the maximum over all possible segment lengths of the last segment ending at *i* and all possible labels of the segment before the last. 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 3. This requires $O(nm^2L)$ time where $m = |\mathbf{y}|$ and $O(mnL)$ space. Since L and m are small integers, this makes the algorithm linear in the length of the input and has been traditionally considered to be a fast algorithm.

Since conventional CRFs not consider maximizing over possible segment lengths *d*, inference for semi-CRFs is more expensive. However, Equation 4 shows that the additional cost is only linear in L. Since in the worst case $L \leq |\mathbf{x}|$, the algorithm is always polynomial. For fixed L, it can be shown that semi-CRFs are no more expressive than order-L CRFs. For order-L CRFs, however the additional computational cost is exponential in L. The difference is that semi-CRFs only consider sequences in which the same label is assigned to all L positions, rather than all $|\mathbf{y}|^b \cdot L$ length-L sequences. This is a useful restriction, as it leads to faster inference.

Semi-CRFs are also a natural restriction, as it is often convenient to express features in terms of segments. For example, let d_j denote the length of a segment, and let . be the average length of all segments with label *I*. Now consider the segment feature $f_k(j, \mathbf{x}, \mathbf{s}) = (d_j -)^2 \cdot [[y_i = I]]$, where $[[P]]=1$ when predicate P is true and 0 otherwise. After training, the contribution of this feature toward $\mathbf{Pr}(\mathbf{s}|\mathbf{x})$ associated with a length-*d* entity will be proportional to $exp^{w_k \cdot (d-\delta)^2}$.

IV. SEMI-CRFS IN PROBABILISTIC DATABASES AND INFERENCE QUERIES

Although information extraction has been cited as a key driving application for probabilistic database research, to date there has been a gap between probabilistic information extraction and ***PDB***.

In this section, we describe the design of a probabilistic database ***PDB***=(***R***, ***F***) that can support probabilistic information extraction models, such as semi-CRFs by (1) storing documents in an incomplete relation ***R*** as an inverted file with a probabilistic label-*p* attribute; and (2) storing the exact probability distribution ***F*** over the extraction possibilities from semi-CRFs through a factor table. A ***PDB*** consists of two key components: (1) a collection of incomplete relations ***R*** with missing or uncertain data, and (2) a probability distribution ***F*** on all possible database instances, which we call possible worlds, and denote *pw* (***PDB***).

*A. The Incomplete Relation **R**: Taken Table:*

The *token table* is an incomplete relation ***R*** in ***PDB***, which stores documents as relations in a database, in a manner similar to the inverted files commonly used in information retrieval. As shown in Figure 1, each tuple in taken table records a unique occurrence of a token, which is identified by the document ID (docID), the start position (spos) and the end position (epos) the token is taken from. A token table has the following schema:

TokenTable (docID, spos, epos, token, label-*p*)

The token table contains one probabilistic attribute label-*p*, which can contain missing values, whose probability distribution will be computed from the semi-CRF model. The deterministic attributes of the token table are populated by parsing the input documents ***D***, with label values marked as missing by default. However, the token table can also be updated by users, who can provide deterministic label values for some of the records.

docID	spos	epos	token	label-*p*
1	1	1	*HouseNo*	
1	2	3	*Area*	
1	4	4	*City*	
1	5	5	*Other*	
1	6	7	*Zip*	

Figure 1. A sample of token table

B. Indpendent tuples model

We need an alternative model of representing imprecision in a database that captures the original distribution, while being easy to store and query. Our method to deal with the problem of extraction uncertainty is to require that each extracted entity be attached with confidence scores that correlate with the probability that the extracted entities are correct. The semi-CRFs 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 ***PDB*** for getting probabilistic answers. Figure 2 illustrates four possible segmentations of address extractions together with their probabilities scores for an address database. Again, these four rows together provide a more informative summary of the imprecision in the data than possible with the highest probability row alone. 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 independent tuples model.

	HouseNo	Area	City	Zip	Prob
s1	52	Goregaon West	Mumbai	400 062	0.1
s2	52-A	Goregaon	West Mumbai	400 062	0.2
s3	52-A	Goregaon West	Mumbai	400 062	0.5
s4	52	Goregaon	West Mumbai	400 062	0.2

Figure 2. Four possible sgementation $\mathbf{s_1}$, $\mathbf{s_2}$, $\mathbf{s_3}$, $\mathbf{s_4}$

C. P- Top-k Queries On Semi-CRFs- based PDF

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 segmentation as output by semi-CRFs is sound in the sense that a probability of p denotes that there is a 100p% chance that the extraction is correct [4].

We now describe *probabilistic* top-k queries over a probabilistic database ***PDB*** supporting the semi-CRF model. It determines the segmentations with the top-k highest probabilities given a token sequence $\mathbf{x}$ from a document.

We consider an approach for representing the uncertainty of extraction in a database system and show how to return probabilistic answers to a project query of the form

SELECT $y_1 \cdots y_k$
FROM ***T***
LIMIT k
WHERE *source*=**x**

Where ***T*** is an imprecise table in ***PDB*** and each y_i refers to one of the k column $c_1 \cdots c_k$ of ***T***. Probabilistic results for such a query will return k rows where each row r consists of l segments of $\mathbf{x} = \{(l_1, u_1), \cdots, (l_l, u_l)\}$and a probability value $p_r = \mathbf{Pr}(s_1=(l_1, u_1), \cdots, s_l=(l_1, u_1)| \mathbf{x})$. We also allow a segment (l_j, u_j) to be *NULL* so as to incorporate missing labels.

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 $\mathbf{Pr}(\mathbf{s}|\mathbf{x})$ of that segmentation. 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, such a representation appears in figure3. In independent tuples model the probability of a query result is calculated by summing over all segmentations that match the result row as follows:

$$\mathbf{Pr}(y_1 = (l_1,u_1),\cdots,(l_l,u_l) \mid \mathbf{x}) = \sum_{\mathbf{s}:\forall (l_i,u_i,y_i)\in \mathbf{s}} \mathbf{Pr}(\mathbf{s} \mid \mathbf{x}) \qquad (6)$$

We study a slight variation of Top-k queries, namely, the probabilistic top-k or simply P-top-k, that return k tuples with the highest probabilities being one of the top-k ranked tuples in a random possible world pw(***PDB***). Formally, the probabilistic top-k is defined as follows:

Definition2. *Probabilistic top-k query*: A probabilistic k top-k query on ***PDB*** return a set of k tuples $\mathbf{S}= \{\mathbf{s}_1, \cdots, \mathbf{s}_k\}$, satisfying $\mathbf{Pr}(\mathbf{s}_i \mid \mathbf{x}) \geqslant \mathbf{Pr}(\mathbf{s}_j \mid \mathbf{x})$, for any $\mathbf{s}_i \in \mathbf{S}$ and $\mathbf{s}_j \notin \mathbf{S}$.

The P-top-k queries return the k most probable segmentation of being the top-k among all. The dynamic programming algorithm is a key implementation technique for top-k inference in information extraction applications. It is feasible to extend sequence models like semi-CRFs to return a set of k highest probability extractions instead of a single most likely extraction. We need to adjust the dynamic programming algorithm to maintain top-k highest scoring solution at each position. To implement this algorithm, we define a new V matrix of quadruple $V(i, y_1, j, y_2)$, where each cell contains the top-k partial segmentation between position i and position j, where y_1 is assigned to position i and y_2 is assigned to position j.

$$V(i, y_1, j, y_2) = \begin{cases} max_{y_{1'}, y_{2'}} (V(i+1, y_{1'}, j-1, y_{2'}) \\ \quad + \sum_k \mathbf{w} \cdot \mathbf{f}(y_1, y_{1'}, l_i, u_i) \\ \quad + \sum_k \mathbf{w} \cdot \mathbf{f}(y_2, y_{2'}, y_{i+1}) \quad if\ (-1 \le i < j \le n) \\ 0 \qquad\qquad\qquad\qquad\qquad if\ (i = j = 0) \end{cases} \qquad (7)$$

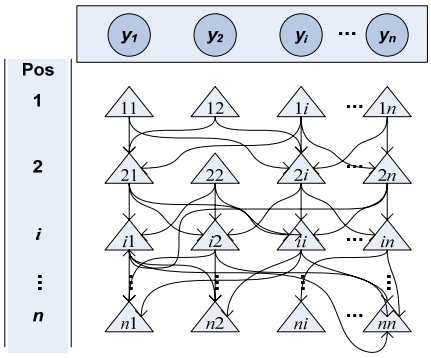

Figure 3. Illustration of computation of ***V*** matrix in the dynamic programming algorithm

As illustrated in Figure 3, we project the four dimensions matrix V to two-dimensional one V' for computing the top-k segmentation using dynamic programming data structure called V' matrix. The first row contains the possible labels $\mathbf{y} = y_1 \cdots y_n$ and the first column contains the position from 1 to n for tokens of $\mathbf{x}=x_1 \cdots x_n$. Let us show that we are computing top-k quires. Each cell in V' (i, y) stores the score and the path of top-k .partial segmentation up until position i ending with label y. The V' matrix at position 0, i.e. V' $(0, y, k)$ is initialized as 0 referring to the equation 7, and if i <0 denotes that the previous label is NULL. The V' matrix at position i >0 is recursively defined form the V' matrix at position i-1 by picking the partial segmentation with ranking k scores. The path of the top-k partial segmentation are may be corresponds to the k edges of path traced. Our algorithm for finding the best m-row model in Algorithm2, it extends maintained states based on each seen tuple. When a new tuple is retrieved, it is used to extend all states causing all possible ranks of this tuple to be recognized

D. Processing Framework

In this section, we propose a processing framework in Figure 4 that leverages PDB storage, indexing, and query processing techniques to compute uncertain P-top–k query answers. Our proposed processing framework consists of two main layers:

Storage Layer: The main functionalities provided in this layer including: tuple retrieval, indexing, query processing (including score-based ranking) and generation rules for uncertain data. In addition, different data access methods are provided to allow the upper processing layer to retrieve uncertain tuples.

Processing Layer: The processing layer converts queries to immediate form, optimizes queries and devises an execution plan for retrieving most probable top–k answers from the underlying storage layer.

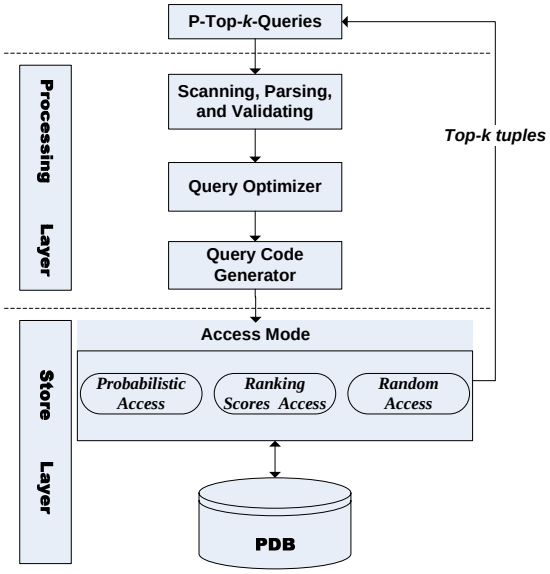

Figure 4. Processing Framework

V. EXPERIMENTS

In this section, we will present an empirical evaluation of our method of constructing probabilistic database from statistical models of structure extraction. We implemented a probabilistic database supporting the semi-CRF model and conducted a systematic empirical study using two data sets on a PC computer with a 3.0 GHz Intel Pentium4 CPU, 1.0 GB main memory, and a 160GB hard disk, running the Microsoft Windows XP Professional Edition operating system. Our algorithms were implemented in Microsoft Visual C++ 6.0. All experiments are run 3 times and take the average running time.

A. DataSets

We use two datasets in the evaluation. The first dataset is **Address** that consists of 770 home addresses of students in IIT Bombay India. There are 5 possible labels used to tag this dataset, such as street names, area, city name, state name and zipcode, respectively. Indian addresses are less regular than others, and extracting even fields like city names is challenging. A subset of 385 addresses was used as the test-set. The second dataset is **Cora** citations data set that consists of 500 bibliography entries from academic papers. There are 13 possible labels for each token in each entry: author, book title, date, editor, institution, journal, location, note, pages, publisher, tech, title, and volume. Although bibliography entries are generally short and follow some conventions, they are interesting because they can display large variations in total length and the length of each section within an entry. In addition, each entry does not always include all possible sections, and there can be differences in the ordering of sections. We use a subset of 350 citations as our test-set. Three example sequences are shown in Figure 5.

1) A. Cau, R. Kuiper, and W.-P. de Roever. Formalising Dijkstra's development strategy within Stark's formalism. In C. B. Jones, R. C. Shaw, and T. Denvir, editors , Proc. 5th. BCS-FACS Refinement Workshop, 1992.
2) M. Kitsuregawa, H. Tanaka, and T. Moto-oka. Application of hash to data base machine and its architecture. New Generation Computing, 1(1), 1983.
3) W. Cohen. Learning from textbook knowledge: A case study. In AAAI-90, 1990.

Figure 5. Three example items from the Cora citations data set

B. Setup

For each dataset, we train a strong and a weak semi-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. In each experiment, we retrieve enough segmentation to cover a sufficient probability mass (0:93 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. A fixed value of L was chosen for each dataset based on observed entity lengths. This ranged between 4 and 6 for the two different datasets.

Features: We used indicator features for the word itself and various surface patterns (capturing capitalization, digit pattern and delimiters) at the word and one position to its left and right. These features apply on the start, end and in-between part of segments to get features equivalent to those in SequenceBCEU. Transition features are used with a history size of 1. These are all word-level features. The segment-level features we added were length feature for each possible segment-length value and for each available external lexicon we added features corresponding to the highest match with whole entities in the lexicon. We measure similarity using the popular TF-IDF similarity function which is known to work well for name-matchingin between two text records r_1 and r_2. The TF-IDF similarity is defined as follows:

$$\text{TF - IDF}(r_1, r_2) = \sum_{t \in r_1 \cap r_2} V(t, r_1) V(t, r_2)$$

$$V(t, r) = \frac{V'(t, r)}{\sum_{t' \in r} V'(t', r)^2} \tag{8}$$

$$V(t, r) = \log(\text{TF}(t, r) + 1) \log(\text{IDF}(t))$$

In the above, the IDF term makes the weight of a token inversely proportional to its frequency in the database and the TF term makes it proportional to its frequency in the record. Intuitively, this assigns low scores to frequent tokens (stop-tokens) and high scores to rare tokens. We use a threshold to limit matches only to values greater than because features with very low scores are not useful and unnecessarily increase running time and have even been found to reduce accuracy in some cases. Even distinctly dissimilar record pairs have non-zero positive similarity scores due to the presence of stop-tokens. For the benefit of the sequence models we also added word level dictionary match features using the same similarity

functions. Thus, we believe that an equivalent set of features are available to the sequence and semi-CRFs models.

The observation features that we used included 23 regular expression features and list-based features (such as person and place names), as well as 1,374 vocabulary features created by extracting whole words from the training data. The regular expression and list-based features that we used are shown in Figure 6. All features are binary and, except for the "EndsIn" features, ignore trailing commas, periods, colons, and semicolons. The observation feature set is fairly simple and could be developed further.

C. Results

We first propose experiments to verify that representing the imprecision of extraction is an available approach to promote the quality of answers returned by the database. That is to say, we show that the current practice of storing the topmost extraction in a conventional database can cause higher error than a database that acquires extraction uncertainty. Our queries project various subsets of the column labels and we survey the error in the query results with reference to the correct segmentation. We report the square error *Se*, which is $(1\text{-}p_c)^2$, where p_c is the probability of the correct answer according to the stored model.

In Figure 7, we illustrate the errors for two situations: there are Top-1 and Top-k queries, respectively. For the first condition we store the single best extraction, where if the single stored segmentation is correct, we get an error of 0, otherwise it is 1. In the second case we store all segmentations with their probabilities so as to acquire the Semi-CRF distribution exactly. This allows us to compare the best possible probabilities model with the current practice of storing extractions. We report errors for weakly as well as strongly trained extraction models.

The models were compared using two metrics: the average per-sequence prediction accuracy and the average F1 score. Per-sequence accuracy is defined to be the number of correctly labeled elements divided by the total number of elements:

$$accuracy(Y,Y') = \frac{\sum_{t=1}^{T}[y_t = y_t]}{T} \tag{9}$$

The F1 score or F measure is defined to be the harmonic mean of the precision and the recall. With respect to a specific label *l*, let *A* be the number of true positives, *B* is the number of false negatives, *C* is the number of false positives, and *D* is the number of true negatives. Precision is the fraction of tokens identified as *l* that really are *l*.

$$P = \frac{A}{A+C} \tag{10}$$

The recall of a method is the number of true positives divided by the total number of positive examples:

$$R = \frac{A}{A+B} \tag{11}$$

NAME	DESCRIPTION
InitCap	Starts with a capitalized letter.
AllCaps	All characters are capitalized.
AllDigits	All characters are digits.
ContainsDigits	Contains at least one digit.
ContainsDots	Contains at least one period.
ContainsDash	Contains at least one dash.
LonelyInitial	A single letter followed by a period.
SingleChar	One character only.
CapLetter	One capitalized character only.
URL	Regular expression for a URL
InParen	In parentheses.
Year	Regular expression for a year.
Punc	Only punctuation (period, comma, colon, semi-colon).
EndsInComma	Ends in a comma.
EndsInDot	Ends in a period.
EndsInColon	Ends in a colon.
EndsInSemiColon	Ends in a semi-colon.
EndsInQuote	Ends in a quotation mark.
StartsWithQuote	Starts with a quotation mark.
Name	Appears in a list of names.
Place	Appears in a list of places.
Acronyms	Appears in a list of acronyms.
Months	Appears in a list of month and day names.
Word	Matches the word.

Figure 6. Observation Features of Cora Dataset

Dataset	Query Schema	The number of label in P-top-*k*			
		1	2	3	4
Cora (Weak Model)	Top-1	0.16	0.29	0.38	0.45
	Top-*k*	0.11	0.2	0.27	0.32
Cora (Strong Model)	Top-1	0.07	0.13	0.18	0.21
	Top-k	0.06	0.11	0.13	0.16
Address (Weak Model)	Top-1	0.31	0.48	0.59	0.66
	Top-k	0.25	0.34	0.38	0.41
Address (Strong Model)	Top-1	0.1	0.17	0.22	0.26
	Top-k	0.08	0.12	0.18	0.19

Figure 7. Se on projection of varying models over top-k queries

The F1 score is thus

$$F1 = \frac{2PR}{P+R} = \frac{2A}{2A+B+C} \tag{12}$$

F1 is one when $B = C = 0$. It essentially measures the number of true positives compared to the number of true positives plus mistakes, ignoring the number of true negatives.

We next evaluate our method for typical named entity recognition (NER) problem. In Figure 8 we compare our model with the popular sequential labeling model with the begin-continue-end-unique (BCEU) encoding of labels (SequenceBCEU). We observe that the average of recall, precision and running time of our model for diffident *L* outperforms SequenceBCEU method. This supports the conclusions in earlier work that our model is better suited for information extraction tasks.

Figure 9 shows the results of recall, precision and F1 reference $L = 4$, $L = 5$ and $L = 6$, respectively for two datasets. We observe that the maximum length 6 for the **Address** dataset with higher recall, precision and F1, and another one corresponding **Cora** is 5.

Dataset	Model	Recall(%)	Precision(%)	F1(%)	Time(msec)
Address	SequenceBCEU	81.8	84.6	83.1	397
	OurModel(Top1)	85.6	88.1	86.8	289
Cora	SequenceBCEU	90.9	90.7	90.8	696
	OurModel(Top2)	91.5	91.1	91.3	592

Figure 8. Comparison of models on three information extraction tasks

Dataset	Recall(%)			Precision(%)			F1(%)
	L=4	L=5	L=6	L=4	L=5	L=6	
Address	84.8	85.6	86.4	86.1	88.1	86.6	86.8
Cora	91.5	92.7	90.3	91.1	92.4	89.8	91.3

Figure 9. Results of recall, precision, F1 for different *L*

VI. CONCLUSIONS AND FUTURE WORK

In this paper we develop a probabilistic database approach to statistical models-based information extraction. We investigated a model of representing imprecision in a database: an independent tuples model that allows row-level uncertainty. Our work here incorporates a specific family of probabilistic modes– linear-chain semi-CRFs models- as a first-class citizen in a probabilistic database. We proposed a framework for efficient computation of probabilistic top-k queries in uncertain databases. The algorithms in this paper take advantage of this specific model's structure to allow features which measure properties of segments, rather than individual elements. Empirical results on real-life datasets show that the performance of our framework is efficient.

In future work we hope to extend our study to in this topic, including handing of multi-table imprecision, extending imprecision to not just information extraction but also integration, and designing fast algorithms in each case.

ACKNOWLEDGMENT

This work was supported in part by the National Science Foundation of China under grants No. 60803086, Science-Technology Plan of Beijing Municipal Commission of Education of China under grants No. KM200910005009 and Youth Foundation of Beijing University of Technology grants No.X1007012201002.

REFERENCES

[1] A. Deshpande, C. Guestrin, S. Madden, J.M.Hellerstein, and W.Hong. "Model-driven data acquisiton in sensor networks," Proc. 30th International Conference on Very Large Data Bases (VLDB 04), 2004, pp. 588–599.

[2] C.Re, J.Letchner, M.Balazinska, and D.Suciu, "Event queries on correlated probilistic streams," Proc. 27th ACM SIGMOD international conference on Management of data(SIGMOD 08), 2008, pp.715-728.

[3] J.Lafferty , A.McCallum , and F.Pereira,"Conditional random fields: Probabilistic Models for Segmenting and Labeling Sequence Data," Proc. In International Conference on Machine Learning(ICML 01), 2001, pp.282-289.

[4] R. Gupta and S. Sarawagi, "Creating probabilistic databases from information extraction models," Proc. 32nd International Conference on Very Large Data Bases (VLDB 06), 2006, pp. 965–976.

[5] D. Burdick, P. Deshpande, T. S. Jayram, R. Ramakrishnan, and S. Vaithyanathan, "Efficient allocation algorithms for olap over imprecise data,". Proc. 32nd International Conference on Very Large Data Bases (VLDB 06), 2006, pp. 391–402.

[6] Aggarwal. C.C. Yu.P.S, " A Survey of Uncertain Data Algorithms and Applications," IEEE Transl. , vol. 21, pp. 609–623, May 2009.

[7] S.Sarawagi, "Automation in information extraction and data integration (Tutorial)," Proc. 28th International Conference on Very Large Data Bases (VLDB 02), 2002.

[8] M. E. Califf, R. J. Mooney, "Relational learning of pattern-match rules for information extraction," Proc. the Sixteenth National Conference on Artificial Intelligence, 1999,pp. 328-334.

[9] R. Gupta, S. Sarawagi, "Creating probabilistic databases from information extraction models," Proc. 32nd International Conference on Very Large Data Bases(VLDB 06), 2006, pp: 965–976.

[10] D. Wang, E. Michelakis, M. Garofalakis, and J. Hellerstein, "BayesStore: Managing Large, Uncertain Data Repositories with Probabilistic Graphical Models," Proc. the 34th International Conference on Very Large Data Bases(VLDB 08), 2008, pp.340-351.

[11] P. Sen, A. Deshpande, "Representing and Querying Correlated Tuples in Probabilistic Databases," Proc. 23rd IEEE International Conference on Data Engineering , (ICDE 07),2007,pp.596-605.

[12] T. Green and V. Tannen, "Models for Incomplete and Probabilistic Information," Data Eng. Bull., vol. 29, no. 1, 2006.

[13] S. Abiteboul, P. Kanellakis, and G. Grahne, "On the Representation and Querying of Sets of Possible Worlds," Proc. ACM SIGMOD , vol.16, no.3, 1987,pp.34-48.

[14] N. Fuhr and T. Rolleke, "A Probabilistic Relational Algebra for the Integration of Information Retrieval and Database Systems," ACM Transactions on Information Systems, 1997, vol.15, no.1, pp.32-66.

[15] L.V.S. Lakshmanan, N. Leone, R. Ross, and V.S. Subrahmanian, "ProbView: A Flexible Database System," ACM Transactions on Database Systems, 1997, vol. 22, no. 3, pp. 419-469.

Ming He was born in 1975. He received his BS, MS and PHD degree in computer science and technology from Xi'an Jiaotong University. He is a lecturer in college of computer science of Beijing University of Technology. His current research interests include database systems, information retrieval, and intelligent information processing.

Yong-ping Du was born in 1977. She received his BS, MS and PHD degree in computer science and technology from Fudan University. She is an associate professor in college of computer science of Beijing University of Technology. Her current research interests include information retrieval and natural language processing.