CrowdPillar: Reducing Uncertainty in Probabilistic Databases by Leveraging Support from the Crowd

Sean Goldberg
University of Florida
sean@cise.ufl.edu

Daisy Zhe Wang
University of Florida
daisyw@cise.ufl.edu

Sunny Khatri
University of Florida
skhatri@cise.ufl.edu

Nilu Nalini
University of Florida
nilu.nalini@gmail.com

Tim Kraska
UC Berkeley
kraska@eecs.berkeley.edu

February 7, 2012

1 Introduction

Discuss general nature of PDBs and common nature of uncertainty. Introduce our goal of reducing the uncertainty in PDBs by adding interaction with the crowd. Motivate information extraction problem. Discuss related work with IE and PDBs.–SLG

2 Background

2.1 Probabilistic Databases

General overview of probabilistic databases and different types of PDBs. Discuss various ways of handling uncertainty in the database and motivate advantages of a reduction in uncertainty.–SLG

2.2 Probabilistic Graphical Models

Small overview of directed and undirected models–SLG

2.3 PGMs in PDBs

Present overview of PGMs implemented into either DBs or PDBs.–SLG

2.4 Entropy

Present entropy as a quantization of uncertainty. Equation and brief example.–SLG

2.5 Amazon Mechanical Turks

Discuss basic idea behind crowdsourcing and give AMT basics.–SLG

3 System Overview

Present main problem definition. Describe high level system components and flow of information. 1. Graphical Model-based PDB. 2. High-level Question formulation. 3. Asking those questions to the crowd and handling the results. Very high level as a layout for the rest of the paper.–SLG

The CrowdPillar system design is shown in (FIGURE). The core of the system is designed as a modification to probabilistic databases that utilize PGMs as their data model. Crowdpillar analyzes the uncertainty in the output and generates questions for crowd submission to reduce some Total Utility Function (TUF). This is discussed in greater detail in (SECTION). The response of the crowd is combined in a principled way with the original PGM output using Dempter-Shafer belief theory, as described in (SECTION).

We consider a sample application task and follow it through the major components of the system: the probabilistic database, question formulation, crowd submission, and data fusion.

Consider an extraction system for automatically reading published citations and storing them in the database. In order to fit an incoming citation into the schema, the string has to be appropriately chunked to determine which tokens belong to which attributes. A string such as: insert citation–SLG needs to be labeled with its appropriate title,

author, conference, etc.

Conditional random fields are particularly well suited to this information extraction task, though many IE methods that can derive the appropriately labelings may be used. There are advantages to storing the CRF model directly in the database. In addition to scalability, parameters of the model may be accessed to decide both which questions to ask and how they should be presented to the crowd.

Once the data has been labeled, we wish to optimally select a number of sequences of high uncertainty and query the crowd for a subset of each sequence's tokens. How many questions can be asked depends on the user's cost and time budget. We discuss a number of sequence selection criteria in (SECTION) based on the type of PGM used.

After determining which sequences should be sent to the crowd, we generate a question in XML format for each sequence. Questions may be in the form of multiple choice, fill in the label, yes/no, etc. Given the possible uncertainty in the crowd response, we resolve conflict and combine responses using Dempster-Shafer theory based on the Turkers' approval ratings to generate a probabilistic collection of responses to each question.

The crowd submissions are sent back to the database, where they are combined with the original output CRF data using (TECHNIQUE)/seanPossibly also DS. The system is designed to be run in batch mode, either whenever a collection of new data is entered into the system or a budget for another round of questions becomes available. How often to specifically query for answers will be dependent on the user's specifications and needs.

4 Question Formulation

Given a graphical model with uncertainty associated with a set of unobserved nodes, the main task of CrowdPillar is designating which nodes to observe and by performing inference reduce the uncertainty of the entire system. Observation is the act of clamping a random variable to a specific value. For the case of information extraction, this is the truth value of the particular token.

This section is concerned with precisely how to select the optimal node to observe from a graphical model. Since the entropy of the system is our main barometer, the node whose resolution best reduces this uncertainty is the one sent to the crowd. Before we can form a discussion about optimal node selection, we must first introduce the metric by which uncertainty is measured in our sysem.

4.1 Total Utility Function

In order to make a decision about which node to query, we develop the concept of a Total Utility Function (TUF) associated with each type of graphical model. The TUF fulfills two prominent and necessary roles: quantification and selection. At its core, it represents a quantification of the uncertainty so that we may measure differences in total uncertainty between different configurations of our model. The second functionality is that is designed in such a way as to rapidly promote uncertainty reduction by identifying the key node hubs whose resolution gives the greatest effect. There are node features that need to be taken into account such as frequency and dependency which will be discussed in greater detail later in the section

The Total Utility Function works by assigning to each unobserved node some kind of score. There are two paradigms that may be used here. The first is to assign each node a score in a simple manner and combine them in some complex model-dependent way. The other is to leave the score combination simple. but create complex ways to attach a score to each node. We choose the latter approach because it is more easily generalizable to various PGMs.

The simplest way to combine scores is to take the sum. This makes node selection for reduction simple. By observing the node with the highest score and reducing its score to zero, the entire function is maximally reduced. The tradeoff is that the method of assigning a score may be much more complex. We devote the remainder of this section to discussion of the various techniques for assigning a score and settle on a combined metric than can be used for any probabilistic graphical model.

4.2 Simple Methods of Assigning Scores

4.2.1 Highest Marginal Entropy

Simplest possible score. Select node with highest marginal. Model: Linear-chain CRF.–SLG

4.2.2 Most Frequent

Select node pertaining to token that appears most in the observation. Model: Skip-Chain CRF.–SLG

4.2.3 Most Dependency

Select node with most connections, ie. upon which others are most dependent. Model: Generalized PGM–SLG

4.3 Neighborhoods: Total Score Metric

Combine all methods above to produce an optimal score metric across any generalized PGM–SLG

5 Probabilistic Data Integration

In an ideal world, all the responses coming from the crowd will be correct values equivalent to the ground truth. We formulate question, get an answer, and replace fields that were requested in the question. In practice, there are numerous reasons a single person from the crowd may not supply the correct result.

Amazon Mechanical Turks is subject to spammers that provide random responses in order to reap the benefit of the HIT while doing little of the work. Much previous research has been done to reduce the effect of spammers. Even honest Turkers, however, may not always give the correct response for reasons such as lack of knowledge, inexperience, or increased difficulty of the questions. The golden standard has been to ask the question multiple times and take a majority vote among the users.

Here we model a new way of looking at the crowd response viewing it as an application of probabilistic data integration. In traditional data integration, data is combined from multiple heterogeneous sources to give the user a single, unified view. Probabilistic data integration attaches probabilities to the combined result. Generally, there is ambiguity in the compatibility of certain relations which leads to a probability value as a result of combination itself, despite the data being indivudally deterministic.

It's also possible for the data coming from different sources to include probability as a first class citizen, that is, the data is naturally probabilistic, such as the combination of probabilistic sensor data. Combining results from the crowd becomes an application in probabilistic data integration if we take each Turker to be a source of data and attach probabilities to their answers based on their Turker Approval Rating.

We employ the machinary of the Dempster-Shafer model of belief functions to combine probabilistic evidence from multiple sources. The goal is to combine data from different Turkers to present a single unified crowd response for merging with the original CRF output. This second step of combining crowd and CRF can be accomplished using the same method, but treating the total crowd and CRF as different probabilistic sources.

5.1 Dempster-Shafer Belief Model

The Dempster-Shafer model [1] [2] employs a mathematical object called a belief function that measures the degree of belief (or mass) someone has about an event. The uncertainty associated with a belief corresponds with missing or incomplete data, as opposed to fuzzy or possibilistic data. Collecting evidence from different sources, one can establish a degree of belief about the reliability of the source itself and therefore also the evidence presented.

Consider an event X. Dempster-Shafer theory maps each element of the power set of X to the interval $[0, 1]$. This mapping is referred to as the /textitmass function. The fundamental properties of the mass function are:

$$m(\emptyset) = 0$$

$$\sum_{A \in 2^X} m(A) = 1$$

That is, the mass of the empty set is always zero and the remaining members of the power set have to sum to 1.

To motivate back to our original problem of information extraction, let's assume there are 2 possible labelings for a token, $X = \{1, 2\}$. Of the four elements of the power set of X, only 3 have mass functions: m($\{1\}$), m($\{2\}$), and m($\{1,2\}$). They correspond to the mass that the proper label is 1, that the proper label is 2, or that we are uncertain and the proper label can still be either 1 or 2.

The *belief* in a set is the sum of all elements of the power set that contain that set. Therefore the belief in label 1 is defined as $m(\{1\}) + m(\{1, 2\})$, with a similar function for the belief in label 2.

Mass functions from multiple sources may be combined in a principled manner using Dempster's Rule of Combination. Let's presume we have two different mass functions m_1 and m_2. The *joint mass* is found with:

$$
\begin{aligned}
m_{1,2}(\emptyset) &= 0 \\
m_{1,2}(A) &= (m_1 \oplus m_2)(A) \\
&= \frac{1}{1-K} \sum_{B \cap C = A \neq \emptyset} m_1(B) m_2(C)
\end{aligned}
$$

where

$$
K = \sum_{B \bigcap C = \emptyset} m_1(B) m_2(C)
$$

The combination is essentially a normalized outer product. All combinations with an intersection equal to the event in question are summed.

5.2 Combining Crowd Data

If multiple Turkers are treated as different sources and their responses and evidence, we can map those responses to belief functions and combine them using the Rule of Combination. The usual technique is to ask the same question to multiple Turkers and take a majority vote. While it has been shown in an image annotation task (CITATION) that repeated labeling and majority vote of the crowd can approach annotation accuracy of experiments, the nature of the combination removes the uncertainty from the result.

Let's say that we limit responses to only those Turkers with a 75% approval rating or above and do a best of 5 majority vote. If three Turkers with an approval rating of 75% answer label 1 and two turkers with an approval rating of 100% answer label 2, the majority vote takes label 1 and passes no other information about who responded with which label. If a particularly difficult question has Turkers divided, this is generally not included in the annotation. This makes sense if the annotation storage component has no method to store probabilities, but this is clearly not the case for probabilistic databases.

If the crowd gives conflicting responses to a question, this discrepancy can be illustrated within the database by determining the belief we have that each answer is correct. This is where we make use of the machinery of Dempster-Shafer theory. For each question submitted, we maintain a running mass function for the correctness of each possible answer. This function is updated as new responses come in from the crowd. If there are n possible answers, we are concerned with $n + 1$ masses, one for each answer plus the uncertainty mass that the Turker is unreliable and any answer is possible.

Algorithm 1 displays the pseudocode for updating a question's current mass function with a new response provided by the Turker. For now, we assume the response takes the form of a multiple choice answer and ranges from 1 to n. The mass function associated with the Turker is zero for every answer except the one they chose, which gets mapped to their approval rating in $[0, 1]$. This defines the likelihood they picked the correct answer. Their unreliability gets mapped to the set of all possible answers $m[n + 1]$.

If there is no current mass associated with the question, the current Turker mass is taken as the Question's mass. Otherwise, we need to combine the two masses using Dempster's Rule of Combination. Remember that the combination for each answer is an outer product over all sets that have an intersection equivalent to that answer. Here we assume mutual exclusivity among all choices and so this intersection occurs only when m and $Q.m$ are the same answer or either is the set of all possible answers. The final result is a newly combined mass function which can easily be converted into a probability distribution for that question.

UNFINISHED–SLG

5.3 Combining Crowd and Machine

After combining multiple answers from the crowd, the total crowd response needs to be aggregated with the original graphical model output. One method is to completely eliminate the machine response and simply put the crowd data in its place and given the increased reliability of human authors can lead to desirable results.

We take a differing view, however, like a detective collecting evidence from different sources. The PGM and crowd represent merely two different sources from which conclusions are being drawn. Rather than throwing away information, we seek to integrate everything we have in a principled way. On questions in which the crowd is divided amongst itself, we can use the original PGM to "break the tie", so to speak.

The combination of crowd and machine is shown in (ALGORITHM2). It is very similar to the within crowd combination, but a couple notable exceptions. The distribution of probability is quite different for the graphical

Algorithm 1: Update Question Belief (Daisy, please let me know if you would change the style of this code)–SLG

Data: Question Q, Turker T, Response $R \in [1, n]$
Result: $Q.m$ = Total mass for Q
begin
  $n \equiv$ number of possible answers to Q;
  $Q.m \equiv$ current mass for Q;
  Initialize masses $m[1]$,...,$m[n+1]$ to 0;
  `// Map response and uncertainty to mass functions`
  $m[R] \leftarrow T.rating$;
  $m[n+1] \leftarrow 1 - T.rating$;
  `// Check if this is the first response`
  if $Q.m = NULL$ **then**
    return $Q.m \leftarrow m$
  else
    Initialize $sum[1]$,...,$sum[n+1]$ to 0;
    for $p = 1$ **to** $n+1$ **do**
      Initialize $K \leftarrow 0$;
      for $i = 1$ **to** $n+1$ **do**
        for $j = 1$ **to** $n+1$ **do**
          `// Take outer product`
          $Outer \leftarrow m[i] * Q.m[j]$;
          `// Check for intersection`
          if $i = j$ *or* $i = n+1$ *or* $j = n+1$ **then**
            $sum[p] \leftarrow sum[p] + Outer$;
          else
            $K \leftarrow K + Outer$;
          end
        end
      end
      `// Renormalization`
      $sum[p] \leftarrow (\frac{1}{1-K}) * sum[p]$;
    end
    return $Q.m \leftarrow sum$
  end
end

model output.

There is no "reliability" or "approval rating" associated with the machine, so we take the assumption that the machine is always reliable in its probabilities. This may seem like a faulty assumption and indeed it would be if we were asking questions that the PGM were very confident in even though they were wrong. Due to the nature of our TUF, only those questions with high entropy and low certainty are presented. With the probabilities widely distributed, it's safer to give the machine 100% reliability and combine its label probabilities as is.

Instead of a binary response vector, the PGM has probabilities associated with each label and we use those directly. Each label response is associated with a mass function in the same way as before, only now there is no mass function for the uncertainty.

Apart from those changes, the outer product is taken in the same way and produces a total combined belief over all of our sources. UNFINISHED–SLG

6 Experiments

Use this section to show correctness of our method with respect to CRF IE task.–SLG

6.1 Entropy vs. Inaccuracy

Experiments showing correlation of high marginal entropy with inaccuracy of CRF.–SLG

6.2 Uncertainty Reduction

Experiments showing uncertainty reduction in DB after applying each of our methods. Will probably put a couple different application domain experiments here. We already have one for citation labeling. Ideally, we need an experiment that utilizes Bayes Nets and another that utilizes skip-chain CRFs.–SLG

7 Future Work

8 Conclusion

References

[1] Arthur P. Dempster: *Upper and Lower Probabilities Induced by a Multivalued Mapping*. Classic Works of the Dempster-Shafer Theory of Belief Functions 2008: 57-72.

[2] Glenn Shafer: *A Mathematical Theory of Evidence*, Princeton University Press, 1976, ISBN 0-608-02508-9.

[3] Alexander J. Quinn, Benjamin Bederson, Tom Yeh, Jimmy Lin, *Crowdflow: Integrating Machine Learning with Mechanical Turk for Speed-Cost-Quality Flexibility*, Technical Report HCIL-2010-09, University of Maryland, 2010

[4] J. Lafferty, A. McCallum, and F. Pereira. *Conditional random Fields: Probabilistic models for segmenting and labeling sequence data*. In Proc. ICML-01, pages 282289, 2001.