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

February 1, 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

2.2.1 Directed Models: Bayesian Nets

Brief summary of Bayes Nets.–SLG

2.2.2 Undirected Models: Conditional Random Fields

Brief summary of CRFs.–SLG

2.2.3 PDB Applications to IE

Discuss efforts like Bayestore and others to put PGMs in the database. Discuss using these PGM-based PDBs for IE.–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

We seek to select tokens or sets of tokens from sequences to formulate into questions to ask the Turkers. Here we go motivate the need to selection questions which reduce a total utility function defined in terms of the entropy. THIS ENTIRE SECTION MAY BE REORGANIZED BY APPLICATION DOMAIN INSTEAD OF GRAPHICAL MODEL USED.–SLG

4.1 Total Utility Function

Describe the idea behind a total utility function and what its purpose is.–SLG

4.2 Highest Marginal Entropy

Selecting those tokens with the highest marginal entropy. Briefly describe entropy here as well.–SLG

4.3 Linear-Chain CRFs

Description of TUF in terms of sum of marginals of each node. Application Domain: Information Extraction–SLG

4.4 Application Specific Extensions to TUF

4.4.1 Skip-Chain CRFs/Frequency

Discussion of different TUF that is a modification to linear-chain. Application Domain: Information Extraction on sequences with multiple token occurrences.–SLG

4.4.2 2nd Extension (Dependent vs. Independent)?

4.5 General PGMs

There are experiments that I want to run that may use the entropy of the markov blanket of a node in any PGM. The TUF would be the sum of all of these markov blankets.–SLG

5 The Crowd

Brief overview of crowdsourcing and amazon mechanical turks. Most of the meat of this section will be saved for a future paper. We should, however, discuss how to handle answers received from the crowd and how we deal with uncertainty. Fusion of the crowd with machine learning may either go here or be expanded into an additional section.–SLG

6 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.

6.1 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.

6.2 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 databases by determining the belief we have that each label is correct. This is where we make use of the machinery of Dempster-Shafer theory.

(ALGORITHM1) displays the pseudocode for combining crowd responses with Dempster-Shafer. The input is a Question data structure that contains a Question ID and a set of Turkers that have provided a response to the question. Each Turker has fields for identification, approval rating, and their response. The actual response is a binary vector the length of the label space with a 1 in the position of their answer and 0s elsewhere.

We must first map each response to a mass function before we employ combination. The number of mass functions is generally equal to the power set of the set of possible answers, but in practice most of these are zero. The nonzero mass functions we are concerned include one for each answer (certainty) plus one associated with all answers (uncertainty), for a total size of NumPossAnswers+1.

For each Turker, we map their response to the associated mass function modified by their reliability (approval rating). Every Turker also has an uncertainty mass function which corresponds to the belief they are unreliable and any of the possible answers is valid. Once the mass functions are set, they are combined as an outer product among all intersecting elements that are not empty. K represents a renormalization of the new combination.

6.3 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.

Algorithm 1: Aggregated Belief

```
Data: Question Q, Turker T, Response R
Result: Q.LB = Total belief in each question's
        answer
begin
    for i = 1 to T.size do
        for j = 1 to Q.numAns do
            m[j] ←
            Turk[i].response[j]∗Turk[i].rating;
        end
        m[numPossAnswers+1]
        ← 1−Turker[i].rating;
        if i = 1 then
            labelBelief ← m;
        else
            labelBelief ← DSCombo(m,labelBelief);
        end
    end
end
```

```
Data: m, labelBelief
Result: labelBelief
for p = 1 to (numPossAnswers+1) do
    k = 0;
    for i = 1 to (numPossAnswers+1) do
        for j = 1 to (numPossAnswers+1) do
            if Intersect(m[i],m[j]) then
                tmpOut[p] ← tmpOut[p] +
                m[i]*labelBelief[j];
            else
                K ← K+ m[i]∗labelBelief[j];
            end
        end
    end
    tmp[p] ← 1/(1−K) ∗tmp[p];
end
labelBelief ← tmpOut;
```

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 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

7 Experiments

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

7.1 Entropy vs. Inaccuracy

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

7.2 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

8 Future Work

9 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.