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

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

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

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

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

Tim Kraska
UC Berkeley
kraska@eecs.berkeley.edu

ABSTRACT

The need to process and store large amounts of uncertain and imprecise data has necessitated the use of Probabilistic Databases (PDBs) which maintain and allow queries over data that carry a degree of uncertainty. An integral part of the data cleaning process is finding efficient ways to reduce this uncertainty. We propose CrowdPillar, a PDB system with an embedded graphical model structure that can employ crowdsourcing techniques such as Amazon Mechanical Turk (AMT) to resolve the most uncertain data entries. CrowdPillar utilizes functions associated with the entropy over nodes in the graph to select which fields should be submitted to the crowd for correction. This paper lays out the CrowdPillar system, discusses formulating questions for submission to AMT, and showcases the use of belief theory to combine responses for integration back into the PDB.

1 Introduction

As the ability to acquire, maintain, and store large amounts of data becomes increasingly easier and more affordable, so does the ability to employ a wide variety of data analyses. A growing trend is in the study of probabilistic data analyses through the application of statistical and machine learning models. Sample applications include businesses modeling their customers' activity through clickstreams and providing recommendations, or the information extraction and entity resolution from raw unstructured text.

One approach for processing, managing, and storing these massive amounts of probabilistic data is to utilize a Probabilistic Database (PDB) [4]. While traditional relational databases have difficulty storing the inherent uncertainty that results from machine learning techniques, PDBs are able to model and provide declarative queries over uncertain data.

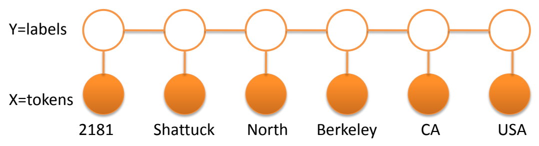
The first Probabilistic Databases that were built relied on simple models of uncertainty that could be easily mapped onto existing relational architectures [2]. This introduced a mismatch between the models used to process the data and those used to store the data. A number of more recent systems have attempted to solve this ”model-mismatch” problem by establishing the Statistical Machine Learning (SML) models and inference algorithms as first class citizens in the PDB. Common Probabilistic Graphical Models (PGMs) such as Bayesian Networks and Conditional Random Fields have already been implemented effectively within the database [18, 25, 24].

SML is not perfect and for some tasks SML can achieve an accuracy as high as 97% while for others (such as image recognition) it is only able to achieve 60%. There are cases where the model is unable to adequately reason about a difficult piece of data and as a result introduces large uncertainties into the data. The need for a new type of data cleaning process has emerged. Previous uses of data cleaning have been to reconcile incorrect or corrupt pieces of data in a deterministic database. For a PDB, a necessary procedure is to reconcile those data which are most uncertain by providing human input.

In addition to utilize machine learning techniques for large scale analysis, an increasing trend has been to harness human computation in a distributed manner using crowdsourcing. Benefits can be found in problems that are too difficult or expensive for computers. Services like Amazon Mechanical Turk (AMT) have led the way by setting up an infrastructure that allows payment for the combined resources of up to hundreds of thousands of people.

CrowdPillar is a system designed to harness the power of crowdsourcing applications like AMT to reduce the uncertainty and improve the reliability of PDB Systems with an inherent graphical model structure. Certain SML applications like information extraction can be performed easily by most humans, but require machine processing to scale with large amounts of data. Compared to traditional single user data cleaning (like that used in active learning), CrowdPillar employs the relative speed and accesibility of the crowd for large scale probablistic data cleaning. Given a graphical model operating on large scale data, CrowdPillar uses methods associated with entropy to pinpoint the weakest and most uncertain nodes in the graph and automatically queries the crowd for correction. Since the crowd may not agree on the results, we combine the response using the Dempster-Shafer theory of evidence before integration into the existing PDB.

We make three main contributions with this paper. First, we present a new algorithm for formulating a list of questions for the crowd ordered by the importance of their answers. These questions are derived from PDBs with a graphical model as their underlying structure. The second contribution is in the application of Dempster-Shafer's Theory of Evidence for combining data from multiple sources, including both the crowd and the SML model. Finally, we present a series of experiments involving real and synthetic data to demonstrate the effectiveness of the previous two contributions.



(a)

x	2181	Shattuck	North	Berkeley	CA	USA	
y1	**street num**	**street name**	**city**	**city**	**state**	**country**	(0.6)
y2	**street num**	**street name**	**street name**	**city**	**state**	**country**	(0.1)

(b)

Figure 1: (a). Conditional Random Field applied to labeling an address sequence. (b). Possible label sequences along with their associated probabilities.

This paper is organized as follows. Section 2 provides a background on PDBs in general as well as some of the theoretical topics necessary to understand how CrowdPillar works. In Section 3, we give an overview of the CrowdPillar system and discuss the process of data flow through the system. The uncertainty in PDBs needs to be reformulated into questions posed to the crowd. Different methods for assigning which questions should be asked are discussed in Section 4. Section 5 overviews Dempster-Shafer theory as an alternative to the common majority voting procedure. Our experimental setup and results can be found in Section 6 while Section 7 contains conclusions drawn and showcases future work to be done.

2 Background

2.1 Probabilistic Databases

Probabilistic Databases arose out of the need to model large amounts of imprecise data. In the Possible Worlds Data Model [4], let $\mathbf{I} = \{I_1, I_2, ..., I_N\}$ be the set of all possible instances of a typical relational database. A PDB is the set $(I_i, p(I_i))$ of all instance-probability pairs, where $p(I_i) \rightarrow [0, 1]$ such that $\sum_{i=1}^{N} p(I_i) = 1$. Queries may be modified to return probabilistic results, though the number of possible worlds may grow exponentially with the size of the database. It is for this reason that queries generally return only the *top k* most probable results. A number of implementations of PDBs currently exist, including TRIO [1], MystiQ [3], MayBMS [7], and Orion [22].

2.2 Probabilistic Graphical Models

Probabilistic Graphical Models (PGMs) [11, 9] are compact graph structures used to encode the conditional independence properties between random variables. Bayesian Networks and Markov Networks are structured as directed acyclic graphs and undirected graphs, respectively. The random variables are represented by nodes in the graph, while dependence relationships are modeled by the edges between them. PGMs provide structure which aids in the inference and learning associated with probabilistic models.

A popular graphical model used extensively in shallow parsing, named entity recognition, and gene finding among others is the Conditional Random Field (CRF) [13]. A CRF is a discriminative undirected PGM pictured in Figure 1(a) which is similar to a Hidden Markov Model (HMM), but able to take advantage of a greater set of overlapping features. Given a set of observed sequence tokens $\mathbf{x}$, such as the words constituting a sentence, a CRF attempts to model the underlying hidden label sequence $\mathbf{y}$ by modeling the conditional distribution

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z} exp\Big(\sum_{t=1}^{T} \sum_{k} \lambda_k f_k(\mathbf{y}_t, \mathbf{y}_{t-1}, \mathbf{x}, t)\Big), \qquad (1)$$

where Z is a global normalization factor, T is the length of the sequence, and the λ_k are learned parameters used to weight a set of fixed *features* f_k. The features are associated with relationships between hidden labels and the observation sequence $\mathbf{x}$. An example feature in an address labeling task is *"the previous label is a number and the observed sequence is capitalized"*, in which case the value associated with the current label being a street name would be much higher than that for any other label. The label sequence $\mathbf{y}$ which maximizes equation 1 can be found by a Viterbi algorithm similar that of a HMM.

2.3 PGMs in PDBs

Early probabilistic databases were forced to make simplistic assumptions about the data (such as complete independence among tuples) as well as being confined to a restricted subset of queries expressable in traditional query language. A number of new systems have attempted to combat this by modeling a PGM directly within the database. BayesStore [25] explicitly represents uncertainty through the use of Bayesian Networks as first class citizens and PrDB [18] casts query evaluation as inference in a graphical model.

2.4 Entropy

The use of entropy as a measure of the uncertainty of a random variable (RV) was first introduced by Shannon [20] in his seminal paper on information theory. The entropy, defined for a random variable X with discrete probability mass distribution p is

$$H(X) = \sum_{i=1}^{n} p(x_i) log(p(x_i)), \qquad (2)$$

where i runs over all possible instances of the random variable. The entropy measures the spread among the probability of each instance. A roughly uniform distribution has high entropy and corresponds to a high degree of uncertainty. The entropy becomes the key metric for question formulation as we discuss later.

2.5 Amazon Mechanical Turks

The Amazon Mechanical Turk (AMT) infrastructure is a marketplace for distributed human computation. Requesters post Human Intelligence Tasks (HITs) such as image annotation and text translation and labeling to be completed by workers drawn from a pool of hundreds of thousands. Most micro-tasks are trivial for humans, but difficult to do automatically through SML. The structure of AMT allows for workers to be compensated and produce results much quicker than by traditional means.

3 System Overview

The CrowdPillar system design is shown in 3. 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 a Total Utility Function (TUF). This is discussed in greater detail in Section 4. 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 5.

In this paper 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.

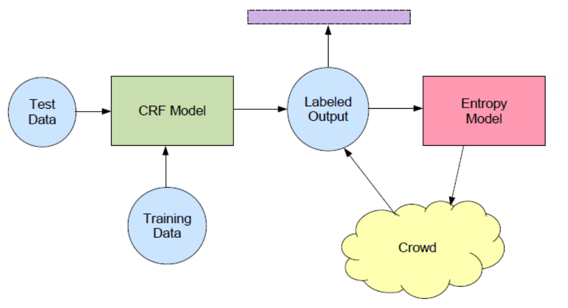


Figure 2: The CrowdPillar system design. The top uncertain labeled outputs as determined by the entropy model are sent to the crowd. The purple box denotes the specified application where the labeled data is ultimately sent.

Consider an extraction system for automatically reading published citations and storing them in the database. In order to fit an incoming citation into any nontrivial schema, the string has to be appropriately chunked to determine which tokens belong to which attributes. A string such as:

> *Perfect Model Checking via Unfold/Fold Transformations. Maurizio Proietti, Alberto Pettorossi CL 3-540-67797-6 Springer Lecture Notes in Computer Science Computational Logic - CL 2000, First International Conference, London, UK, 24-28 July, 2000, Proceedings 2000*

needs to be labeled with its appropriate title, author, conference, etc. This is in general a nontrivial problem, especially when one considers the numerous styles and orderings different citations come in. An efficient extraction system must be trained to handle these different occurences.

The state of the art model for this type of information extraction task is the conditional random field. We focus specifically on CRFs for the remainder of the paper and make it the flagship example of CrowdPillar, though there exist other models that can derive the appropriate labelings and confidences for which CrowdPillar would also apply. After the CRF has segmented and labeled the data, each label sequence is stored as a possible world as in Figure (1b)

The goal of CrowdPillar is to optimally select a number of sequences of high uncertainty from the CRF output and query the crowd for the label of each token in a sequence with the highest individually assigned score. We discuss specifically how scoring is done in Section 4 and how many questions can be asked depends on the user's cost and time budget. An on-line approach to selecting sequences is highly inefficient due to the relatively slow speed of returning a crowd response (on the order of hours) as well as the high parallelizability of largely independent sequences. For these reasons, sequences are selected in a batch for submission to the crowd.

After determining which sequences should be sent, 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 multiple answers to each question using Dempster-Shafer belief theory based on a quality metric attached to each answer to ultimately generate a probabilistic collection of responses to each question. The quality $Q \to [0, 1]$ may take a number of different forms, including difficulty of the question and approval rating and bias of the responder. It represents the confidence we have in the answer being correct as opposed to being a randomly generated guess.

The crowd submissions are sent back to the PDB upon completion, where they are combined with the original output CRF data using the Dempster-Shafer Theory of Evidence. The answers can be returned at once upon completion of the batch or queried in intervals. How often to specifically query for answers will be dependent on the user's specifications and needs.

4 Question Formulation

Given a graphical model designed to perform statistical analysis upon a PDB and the 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 label provided by the crowd.

This section is concerned with precisely how to select the optimal node to observe from a graphical model. The node whose resolution best reduces the overall model 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). 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. Node features such as entropy and dependency need to be taken into account and will be discussed in greater detail later in the section.

There are two paradigms that may be used here. The first is to assign each node t a score S_t 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, ie. for a model with T nodes:

$$TUF = \sum_{t=1}^{T} S_t \tag{3}$$

This makes node selection for uncertainty reduction simple. By observing the node with the highest score and reducing that 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 that we assert can be used for a wide range of probabilistic graphical models.

4.2 Simple Methods of Assigning Scores

Below we present two methods of attaching scores to nodes based on their marginal entropy and dependency among other nodes. The main application of these methods is to linear-chain conditional random fields, but additional model dependent methods may be used in the appropriate context.

4.2.1 Marginal Entropy

Linear-chain Hidden Markov Models or Conditional Random Fields are one the simplest type of PGM to assign a score to due to the uniformity of each node in the graph. With the exception of the first and last nodes in the chain, every node has the same type and number of pair-wise edge features. Thus any means of assigning S_t is consistent across all nodes.

The property we hope to achieve with score assignment is to give the largest scores to the most uncertain nodes. The uncertainty associated with a specific node is achieved by calculating the entropy over its marginalized distribution. Marginalization in a CRF is done through the forward-backward algorithm [13] utilizing forward (α) and backward (β) inference messages meeting at the marginalized node. The marginal probability of a node t in the sequence being assigned label i is given by

$$p(\mathbf{Y}_t = y_i|\mathbf{x}) = \frac{\alpha_t(y_i|\mathbf{x})\beta_t(y_i|\mathbf{x})}{Z(\mathbf{x})}. \quad (4)$$

The marginal entropy of node t is then

$$H_t = \sum_{i=1}^{L} p(\mathbf{Y}_t = y_i|\mathbf{x}) log[p(\mathbf{Y}_t = y_i|\mathbf{x})], \quad (5)$$

where L is the label space of the problem. This leads to a simple uncertainty score assignment of $S_t = H_t$.

In making a decision about which node in a sequence of length T to the submit to the crowd, we calculate the marginal entropy H_t for every node and take the highest one,

$$Selection = \underset{t}{\operatorname{argmax}}(H_t) \qquad t = 1, \ldots, T. \quad (6)$$

The crucial assumption here is that the most uncertain node generally has the highest probability of being incorrect compared with other nodes in the sequence. In our experiments in Section 6 we show that this is a safe assumption to make.

Although the response from the crowd only holds information pertaining to a single node's assignment, inference provides a wider-reaching chain reaction effect. As discussed earlier, the most likely path sequence is determined by a type of Viterbi dynamic programming algorithm. After clamping the label of node t to label i, we run a Constrained Viterbi [12] which fixes the Viterbi path to run through i at t. This may change the labels of other nodes in t's neighborhood compared to before clamping. Thus the total entropy of the entire sequence may be reduced well beyond the contribution of the selected node.

4.2.2 Most Dependency

Since inference provides a means of updating multiple nodes from a single answer, it would be useful to have a score that is proportional to this chain reaction effect. That is, those nodes with the greatest ability to impact its surrounding neighborhood should be awarded the highest score. This increases the impact of every question asked.

For generalized graphical models, it is clear the impact of resolving a node is proportional to the number of nodes which share edge-wise features with it. Nodes with a higher degree are more "important" to resolve. Thus another metric for scoring can be the degree of each node,

$$SELECTION = \underset{t}{\operatorname{argmax}}(deg(V_t)) \qquad t = 1, \ldots, T. \quad (7)$$

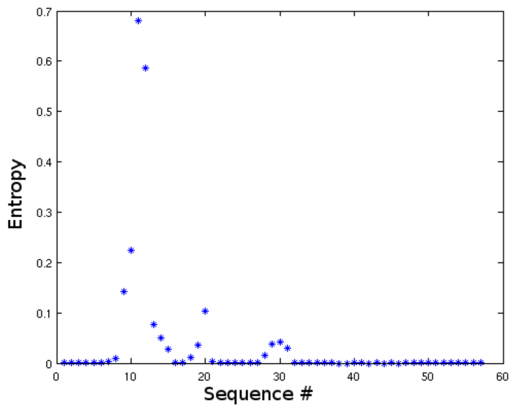


Figure 3: The entropy distribution for a sample bibliographic sequence.

To illustrate this idea further, consider an experiment involving a connected social network and movie recommendation system. We want to select a single user to ask about which movies they like and in turn make recommendations to their friends under the assumption that friends like similar movies. The user likely to give the maximum number of recommendations is simply the one with the most friends. Modeling the social network as a graph, this is the node with the most connections.

This score assignment based on the node with the highest dependency can also be used in the information extraction domain. One recent modification to the linear-chain model is to use "skip-edges" [23] connecting possibly distant nodes in the sequence. The goal is for multiple nodes representing the same entity to be resolved simultaneously. In this case we would seek to ask a question about the nodes connected by the greatest number of skip-edges, ie. resolving those entities that appear most in the sequence and whose resolution has the greatest impact.

4.3 Neighborhoods: Total Score Metric

It is desirable to be able to assign a score to a a node in a generalized probabilistic graphical model without explicitly depending on the type of model, eg. linear-chain vs. skip-chain. The two factors by which a score can be measured are the entropy of the node and its degree. To this end, we propose a method that combines both properties and can be applied to a numerous models more general than the linear-chain discussed in detail in this earlier in this section.

Figure 4.3 shows the entropy distribution for the labeling of a sample bibliographic sequence. Rather than being randomly distributed, high entropy values appear in pockets of small neigborhoods. This is standard among many sequences investigated. This notion of clustering of high entropy nodes around the highest ones in the sequence motivates an additional method of combining scores that naturally carries the properties needed from the previous two sections.

Instead of taking the marginal entropy over a single node as the only score metric, we combine the entropies of each node with those of all nodes of order K or less. By this we mean all nodes connected to the node in question by K or fewer edges. Formally:

$$\begin{aligned} S_t = &\sum_{i=1}^{L} p(\mathbf{Y}_t = y_i) log[p(\mathbf{Y}_t = y_i)] \\ &+ \sum_{k=1}^{K}\sum_{j=1}^{L} p(\mathbf{Y}_{t+k} = y_j) log[p(\mathbf{Y}_{t+k} = y_j)] \\ &+ \sum_{k=1}^{K}\sum_{m=1}^{L} p(\mathbf{Y}_{t-k} = y_m) log[p(\mathbf{Y}_{t-k} = y_m)] \end{aligned} \quad (8)$$

In our experiments we take $K = 1$ and analyze all nodes connected by a single edge. Again $SELECTION$ is the node with the max S_t.

As per Figure 4.3, this should have little effect on the single highest marginal method for linear-chain models. The highest entropy node in a sequence usually appears among other high entropy nodes. In addition, for models with greater variation in connectivity, nodes with a larger neighborhood will generally have a larger score by the Neighborhood Metric. This method automatically accounts for the tradeoff between node pockets of high entropy and others that are connected to a larger number neighboring nodes. Currently, we only cover linear-chains CRFs, but we plan to apply this method and conduct experiments on other models such as Bayesian Networks in our future work.

5 Probabilistic Data Integration

While the previous section focused on the number of ways to select a node in a PGM to submit to the crowd, it remains to be seen how to handle the retrieved result. The problem is handled as a task in probabilistic data integration involving data from both the machine model and the crowd. In this section we discuss a method of combining crowd response and integrating it back into the system in a probabilistic manner.

5.1 Uncertainty From the Crowd

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

Services like Amazon Mechanical Turk are subject to spammers that provide random responses in order to reap the benefit of a Human Intelligence Task (HIT) while doing little of the actual work [14]. A good deal of research has been done to reduce the effect of spammers including repeated labeling [21] and interspersing ground truth to gauge Turker effectiveness [15]. 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 some quality rating. This quality rating can be something simple like the Turker's approval rating or something more complex. For instance, Panagiotis et al. [8] build upon a model for recovering worker error rates by Dawid and Skene [5] and are able to separate worker bias to assign a true Turker accuracy.

We employ the machinary of the Dempster-Shafer model of belief functions to combine probabilistic evidence from multiple workers. The goal is to combine data to present a single unified crowd response for merging with the original CRF output in the PDB. 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.

5.2 Dempster-Shafer Belief Model

The Dempster-Shafer (DS) model [6, 19] 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 *mass function*. The fundamental properties of the mass function are:

$$m(\emptyset) = 0 \quad (9)$$

$$\sum_{A \in 2^X} m(A) = 1 \quad (10)$$

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 binary labelings for a token, $X = \{0, 1\}$. Of the four elements of the power set of X, only 3 have mass functions: $m(\{0\})$, $m(\{1\})$, and $m(\{0, 1\})$. They correspond to the mass that the proper label is 0, that the proper label is 1, or that we are uncertain and the proper label can still be either 0 or 1.

The *belief* in a set is the sum of all elements of the power set that contain that set. Therefore the belief in label 0 is defined as $m(\{0\}) + m(\{0, 1\})$, with a similar function for the belief in label 2. Renormalizing over the beliefs provides a probability distribution over the set of labels.

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_0 and m_1. The *joint mass* is found with:

$$\begin{aligned} m_{0,1}(\emptyset) &= 0 \\ m_{0,1}(A) &= (m_0 \oplus m_1)(A) \\ &= \frac{1}{1-K} \sum_{B \cap C = A \neq \emptyset} m_0(B) m_1(C) \end{aligned} \quad (11)$$

where

$$K = \sum_{B \bigcap C = \emptyset} m_0(B) m_1(C) \quad (12)$$

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

5.3 Combining Crowd Data

If multiple Turkers are treated as different sources and their responses as 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 generally effective, there arise situations where the crowd is split on a result. Possibilities include an image of poor quality or a text analytic question in which the majority of workers are not proficient in English [16].

Let's say that we limit responses to only those Turkers with a 75% quality rating (that is, they give the correct label 75% of the time and an incorrect label 25% of the time) or above and do a best of 5 majority vote. If three Turkers with an approval rating of 75% answer label and two turkers with an approval rating of 100% answer label 1, 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, such uncertainties can not be stored and processed in a traditional DB. Our use of a PDB framework allows the uncertainties to persist and the combination method provides a more reliable estimate of this uncertainty.

If the crowd gives conflicting responses to a question, this discrepancy can be reflected 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 their response is equivalent to a random answer.

Algorithm 1 displays the pseudocode for updating a question's current mass function with a new response provided by the Turker. 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 quality 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 and integrated back into the PDB.

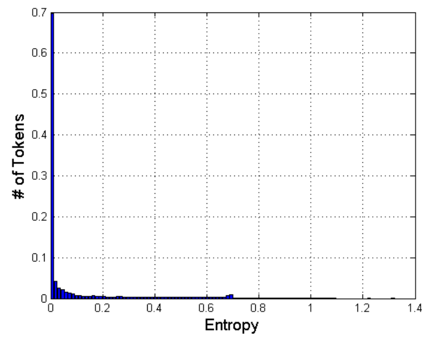
5.4 Combining Crowd and Machine

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. Given the increased reliability of human authors, this can lead to desirable results.

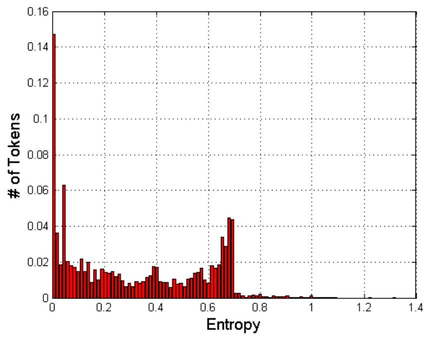
We take a differing view, however, like a detective collecting evidence from different sources. The PGM and crowd represent merely two different sources from which conclusions are being

Algorithm 1: Update Answer Belief

```
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
```



(a) Correctly Labeled



(b) Incorrectly Labeled

Figure 4: Normalized entropy histograms for correctly and incorrectly labeled tokens.

drawn. Rather than throwing away information, we seek to ML and crowd answers in the same principled way. On questions in which the crowd is divided amongst itself, we can use the original PGM to "break the tie".

The combination of crowd and machine is the same as Algorithm 1, except that that there exists no analog to the quality rating in the machine output. While a confidence value can be attributed to a measure of certainty, it is not necessarily true that the complement is a measure of randomness. The simple method is to forego the use of a quality rating and uncertainty metric and map the confidence values for each label directly to its associated mass function. The label confidence from a CRF can be attributed from the marginal probabilities [12].

The final result is a complete mass function which can easily be converted into a probability distribution.

6 Experiments

Our experiments were conducted using 9600 entries generated from the PROXIMITY DBLP database of Computer Science bibliographies [10]. Each bibliographic sequence contains 8 fields pertaining to the title, authors, conference, ISBN, publisher, volume, proceedings, and year. Training was performed on 4600 entries and testing on 5000 entries. We implemented most of the CrowdPillar algorithm in Java 1.8 using the feature set provided by Sarawagi [17].

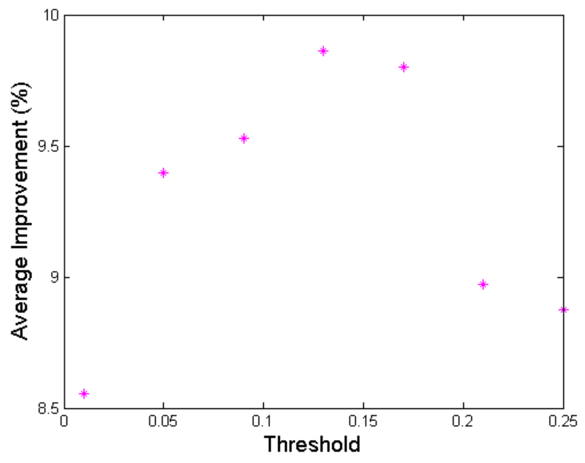


Figure 5: Average increase in accuracy for each clamped sequence selected by various thresholds. The threshold determines the level above which a node's entropy is considered "high".

6.1 Entropy vs. Inaccuracy

The fundamental assumption our question formulation methods hinge upon is the direct correlation between inaccurate tokens and their marginal entropy level. Figures 4(a) and 4(b) demonstrate the veracity of this assumption. The marginal entropy of every token in the 5000 sequence test set is measured and histogrammed by correct and incorrect sequences. The majority of accurate tokens tend to lower entropy values while the inaccurate ones appear in a much higher range.

6.2 Uncertainty Reduction

It is intractable to be able to ask a question about every node and send it to the crowd. The key to CrowdPillar's success is the ability to optimally select nodes from budget size of k sequences which reduce uncertainty in the entire system the most. In testing our node selection algorithm, the chosen nodes were replaced by their ground truth values. In practice, as discussed in Section 5, the real feedback from the crowd is complex and probabilistic. After being clamped to the nodes' ground truth values, each sequence was rerun with Constrained Viterbi to produce a new label sequence.

In practice, it is impractical to select a node from every sequence for sending to the crowd, so we sort the sequences according to some metric and select the top k depending on the budget. We tested two different sorting metrics. The first takes sorts by the entropy of the maximum entropy node in the sequence. The second is slightly more complex. We classified each sequence by the number of "high" entropy nodes it contained. The threshold for determining what constituted high entropy was found by a parameter search. Figure 5 shows the average increase in accuracy for the top 500 testing sequences sorted by the number of high entropy nodes determined by each threshold. Based on the results, we deduced $\tau = 0.13$ as an appropriate threshold.

After determining which sequences needed to be corrected by the two sorting metrics, we tested two different methods of assigning scores to each node as discussed in Section 4. We either assigned S_t to be the marginal entropy or the neighborhood marginal entropy with neighborhoods including tokens before and after node t.

The two sorting metrics and two scoring metrics allowed for four distinct ways of selecting the top k sequences and then selecting a

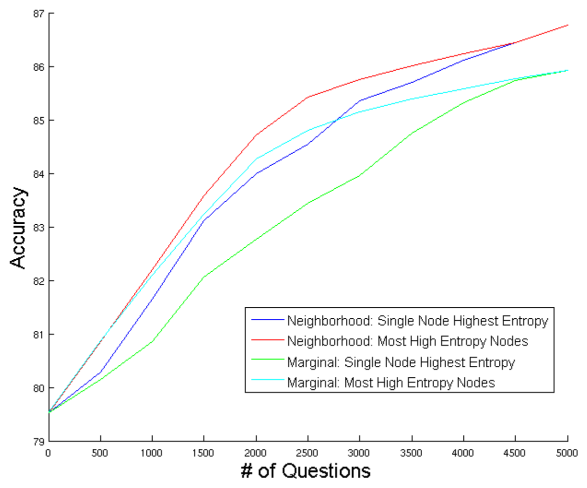


Figure 6: Comparison of the 4 different methods for sorting sequences and scoring nodes within each sequence.

node from those sequences. Figure 6.2 shows the accuracy increase for each of these methods as a function of the number of questions asked. Sorting by the number of high entropy nodes appears to be the best metric for determining inaccurate sequences. The neighborhood marginal entropy as a scoring metric also provides a small boost compared to simply taking the single node's marginal entropy. The combination of neighborhood marginal and number of high entropy nodes produced as much as a 7% total increase in accuracy when applied to all questions in the testing set.

6.3 Dempster-Shafer Combination

To test the use of DS combination vs. majority voting for probabilistic data, we generated a synthetic data set for Turker responses to 1,000 binary questions. M Turker responses were generated for each question for a total of $1000M$ responses, where M =3, 5, 7, 9, 11, and 15. For each response, a Turker was generated with quality rating drawn from a normal distribution centered at 0.9, 0.7, 0.5, and 0.3 with a deviation of 0.1 and thresholded beteen [0,1]. The quality rating determined the likelihood that Turker's answer reflected the ground truth. For example, a Turker with a quality rating of 0.7 was associated with an answer that was correct with 70% probability and a random choice with 30% probability. This rating could be derived from the individual Turker quality or could be seen as a function of a difficult question that produces low quality results.

Figure 7 shows an accuracy comparison between Dempster-Shafer combined responses and those combined with majority voting for differing levels of Turker quality. It's observed that the poorer the quality of the Turker or the question, the worse majority voting does in relation to DS. DS obtains it's full power when the quality of responses is very poor and there exists a large uncertainty between answers.

7 Conclusion

CrowdPillar is a graphical model-based PDB system designed to handle uncertain data and administer Amazon Mechanical Turk in the resolution of the most uncertain parts of the data. Experiments were tested on 5000 bibliographic sequences that required labeling. CrowdPillar was able to effectively select sequences most in

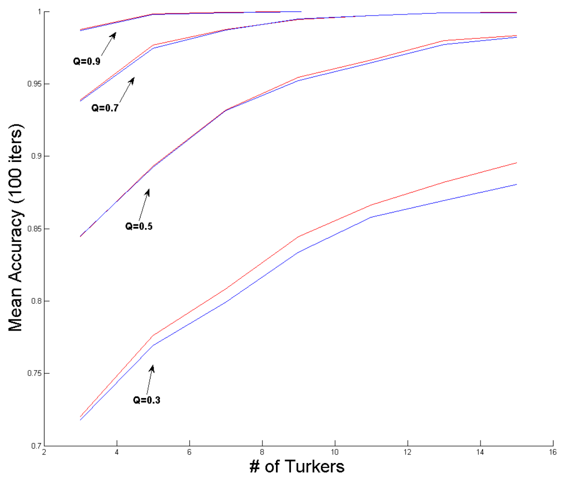


Figure 7: Comparison of Dempster-Shafer combination (red) vs. majority voting (blue) for possibly unreliable Turkers. Q denotes the mean quality level of the selected Turkers. The DS method increases in performance compared to majority voting as the Turkers become more unreliable.

need of correction and increase accuracy greatly by correcting just a single node in each one. While having the worker label the entire sequence is a highly laborous task, by asking a question at only one token we were able to increase accuracy by a total of 7.2%. The fact that accuracy increases most within the smallest number of questions shows CrowdPillar is putting priority on the most incorrect ones first even without access to ground truth.

Future directions of research include the testing on additional graphical models such as Bayesian Networks as well as additional scoring and selection metrics. In addition, we wish to explore new application domains including image annotation and entity resolution.

8 References

[1] P. Agrawal, O. Benjelloun, A. D. Sarma, C. Hayworth, S. U. Nabar, T. Sugihara, and J. Widom. Trio: A system for data, uncertainty, and lineage. In *VLDB*, pages 1151–1154, 2006.

[2] D. Barbará, H. Garcia-Molina, and D. Porter. The management of probabilistic data. *IEEE Trans. on Knowl. and Data Eng.*, 4:487–502, October 1992.

[3] J. Boulos, N. Dalvi, B. Mandhani, S. Mathur, C. Re, and D. Suciu. Mystiq: a system for finding more answers by using probabilities. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, SIGMOD '05, pages 891–893, New York, NY, USA, 2005. ACM.

[4] N. Dalvi and D. Suciu. Management of probabilistic data: foundations and challenges. In *PODS '07: Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 1–12, New York, NY, USA, 2007. ACM Press.

[5] A. P. Dawid and A. M. Skene. Maximum Likelihood Estimation of Observer Error-Rates Using the EM Algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):20–28, 1979.

[6] A. P. Dempster. Upper and lower probabilities induced by a multivalued mapping. *Annals of Mathematical Statistics*,

38:325–339, 1967.

[7] J. Huang, L. Antova, C. Koch, and D. Olteanu. Maybms: a probabilistic database management system. In *In SIGMOD Conference*, 2009.

[8] P. G. Ipeirotis, F. Provost, and J. Wang. Quality management on amazon mechanical turk. In *Proceedings of the ACM SIGKDD Workshop on Human Computation*, HCOMP '10, pages 64–67, New York, NY, USA, 2010. ACM.

[9] M. Jordan. *Learning in graphical models*. Adaptive computation and machine learning. MIT Press, 1998.

[10] U. o. M. A. Knowledge Discovery Laboratory. Proximity dblp database. http://kdl.cs.umass.edu/data/dblp/dblp-info.html, 2006.

[11] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[12] T. Kristjansson, A. Culotta, P. Viola, and A. McCallum. Interactive information extraction with constrained conditional random fields. In *Proceedings of the 19th national conference on Artifical intelligence*, AAAI'04, pages 412–418. AAAI Press, 2004.

[13] J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[14] I. Panos. Mechanical turk: Now with 40.92% spam. http://www.behind-the-enemy-lines.com/2010/12/mechanical-turk-now-with-4092-spam.html, Dec 2010.

[15] A. Quinn, B. Bederson, T. Yeh, and J. Lin. CrowdFlow: Integrating Machine Learning with Mechanical Turk for Speed-Cost-Quality Flexibility. Technical report, University of Maryland, May 2010.

[16] C. Rashtchian, P. Young, M. Hodosh, and J. Hockenmaier. Collecting image annotations using amazon's mechanical turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, CSLDAMT '10, pages 139–147, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[17] S. Sarawagi. Crf java package. http://crf.sourceforge.net/, 2004.

[18] P. Sen, A. Deshpande, and L. Getoor. Prdb: managing and exploiting rich correlations in probabilistic databases. *The VLDB Journal*, 18:1065–1090, October 2009.

[19] G. Shafer. *A mathematical theory of evidence*. Princeton university press, 1976.

[20] C. E. Shannon. A mathematical theory of communication. *Bell system technical journal*, 27, 1948.

[21] V. S. Sheng, F. Provost, and P. G. Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, pages 614–622, New York, NY, USA, 2008. ACM.

[22] S. Singh, C. Mayfield, S. Mittal, S. Prabhakar, S. E. Hambrusch, and R. Shah. The orion uncertain data management system. In *COMAD*, pages 273–276, 2008.

[23] C. Sutton and A. Mccallum. Collective Segmentation and Labeling of Distant Entities in Information Extraction. Technical Report TR # 04-49, University of Massachusetts, July 2004.

[24] D. Z. Wang, M. J. Franklin, M. Garofalakis, J. M. Hellerstein, and M. L. Wick. Hybrid in-database inference for declarative information extraction. In *Proceedings of the 2011 international conference on Management of data*, SIGMOD '11, pages 517–528, New York, NY, USA, 2011. ACM.

[25] D. Z. Wang, E. Michelakis, M. Garofalakis, and J. M. Hellerstein. Bayesstore: managing large, uncertain data repositories with probabilistic graphical models. *Proc. VLDB Endow.*, 1:340–351, August 2008.