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 27, 2012

1 Introduction

As the ability to acquire, maintain, and store large amounts of data becomes increasingly easier and more affordable, so too 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 and storing these massive amounts of probabilistic data is to store it in a Probabilistic Database (PDB). While traditional relational databases have difficulty storing the inherent uncertainty that results from machine learning techniques, PDBs are able to model this uncertainty naturally and provide declarative querying that supports it. This allows the power of statistical learning tasks to be applied in a manner that also scales efficiently with the data.

The first Probabilistic Databases that were built relied on simple models of uncertainty that could be easily mapped onto existing relational architectures. 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 models and inference algorithms as first class citizens in the PDB. Common Probabilistic Graphical Models (PGMs) such as Bayesian Networks have Conditional Random Fields have already been implemented effectively within the database (CITATION).

On occasion, 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.

CrowdPillar is a system designed to harness the power of crowdsourcing applications to reduce the uncertainty and improve the viability of Probabilistic Database Systems with an inherent graphical model structure. Certain statistical applications like information extraction can be performed easily by most humans, but require machine processing to scale with the large amounts of data. CrowdPillar employs the 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.

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

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

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 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 very difficult 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.

Despite the difficulty, conditional random fields are particularly well suited to this information extraction task. It is for this reason that we focus on CRFs for the remainder of the paper and make it the flagship example of CrowdPillar, though many other IE methods that can derive the appropriately labelings may be used. The parameters associated with the CRF allow us to designate in a simple way which questions should be provided to the crowd.

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 individual score. We discuss specifically how scoring is done in Section 4. 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 achieving 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 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 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 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). 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.

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

Linear-chain Hidden Markov Models or Conditional Random Fields are perhaps 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 pairwise edge features. Thus any means of assigning a score 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. For a linear-chain CRF, this leads to a simple method of ascertaining the score: (EQUATION) where p_i^j is the probability of the m^{th} node being assigned label i and L is the size of the label space.

In making a decision about which node in a sequence to the submit to the crowd, we calculate the marginal entropy for every node and take the highest one. 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 (SECTION) we prove this to be precisely the case.

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 that uses values obtained from the Forward-Backward algorithm. Changing the label of a single node m may change the labels of other nodes in m's neighborhood. The total entropy of the entire sequence may be reduced well beyond the contribution of the queried 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.

It is intractable to model this dependency exactly. First, it's impossible to know how a neighborhood's nodes will be updated in advance of the answer being received. One would have to consider the result of every answer and compute some average change in uncertainty over the entire graphical model. In addition, this process would have to be done over every node in the model. If takes an amount of time T to do inference, the total time complexity increases to (TIME). This is computationally infeasible for most models.

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 is the degree of each node.

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 edges.

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" 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. 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 paper.

Figure (FIGURE) shows the entropy distribution for a 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: (EQUATION) In our experiments we take $k = 1$ and analyze all nodes connected by a single edge.

As per Figure (FIGURE), 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. As we show in our experiments, in some cases where this is not true accuracy is actually improved by correcting a pocket of high nodes which individually do not have the highest entropy in the sequence.

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. While not covered in experiments in this paper, it is our belief that this metric could be applied universally to undirected models and undirected models. Currently, we only cover linear-chains CRFs, but application of this method to other models such as Skip-Chain and Directed Trees is a direction for future work.

While this section has 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 the next section we discuss a method of combining crowd response and integrating it back into the system in a probabilistic manner.

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

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
```

(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

6 Experiments

Our experiments were conducted using (NUM ENTRIES) entries from the PROXIMITY DBLP database of Computer Science bibliographies (CITATION). Each bibliographic sequence contains 8 fields pertaining to the title, authors, conference, ISBN, publisher, volume, proceedings, and year. While the entire CrowdPillar system is not yet complete, we seek to justify the ideas presented in this paper using a standard Java implementation of a linear-chain CRF (CITATION), namely the ability to highlight the most inaccurate tokens based on entropy methods and correct them in batch using the crowd.

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 (FIG) and (FIG) demonstrate the veracity of this assumption. The marginal entropy of every token in the 8,629 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. Table (TABLE) shows further statistics for two classes.

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 Crowd-Pillar's success is the ability to optimally select nodes from budget size 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 (SECTION), the real feedback from the crowd is complex and probabilistic. After being clamping to the nodes' ground truth values, each sequence was re-run with a "Clamped Viterbi" algorithm, which is the original CRF Viterbi modified so all label paths pass through the truthed node's value.

With a budget of $k = 100$, we explored two possible ways of selecting the optimal token sets. In the first, we merely sorted by the value of the highest entropy node in the sequence and took the k. As an alternative, we experimented with a metric embodying the number of "high" entropy nodes the sequence contained. The motivation for this additional method was the observation that some correct sequences contained a small number of high entropy nodes, while the more inaccurate ones contained entropies of a smaller magnitude but in greater number.

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 0.8 was associated with an answer that was correct with 80% probability and a random choice with 20% 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 1 shows an accuracy comparison between Dempster-Shafer combined responses and those com-

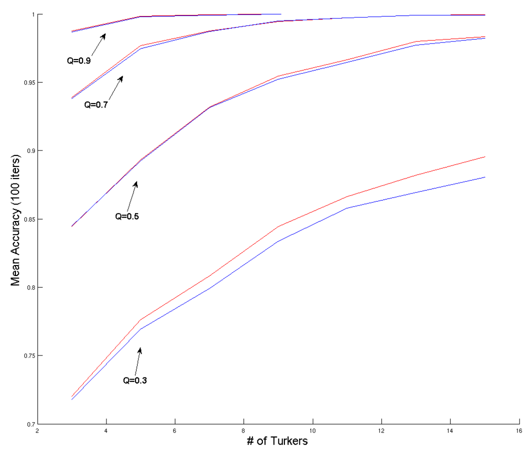

Figure 1: Comparison of Dempster-Shafer combination (red) vs. majority voting (blue) for possibly unreliable Turkers. Q denotes the mean quality level of the selected Turkers. The DS method increases in performance compared to majority voting as the Turkers become more unreliable.

bined 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 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.