A Probabilistically Integrated System for Crowd-Assisted Text Labeling and Extraction

SEAN GOLDBERG, University of Florida
DAISY ZHE WANG, University of Florida
CHRISTAN GRANT, Brown University

The amount of text data has been growing exponentially in recent years. State-of-the-art statistical text extraction methods over this data are likely to contain errors. Recent work has shown probabilistic databases can store and query uncertainty over extraction results, however, these systems do not nately result in a reduction of error. In this paper we propose pi-CASTLE, a system that uses a probabilistic database as an anchor to execute, optimize and integrate machine and human computing. Uncertain fields are crowdsourced with the goal of reducing uncertainty and improving accuracy. We use information theory to optimize the set of questions and a novel Bayesian probabilistic model to integrate uncertain crowd answers back into the database. Experiments show promising results in significantly reducing machine error using very small amounts of data. Additionally, probabilistic integration is shown to more effectively resolve conflicting crowd answers and provide users with the flexibility to tune the desired trade-off between accuracy and recall according to the need of applications. Using crowds to assist machine-learned models proves to be a cost-effective way to close the "last mile" in terms of accuracy for text labeling and extraction tasks.

CCS Concepts: •**Information systems** → **Crowdsourcing; Information extraction;**

Additional Key Words and Phrases: Crowdsourcing, Information Extraction, Probabilistic Models

ACM Reference Format:
Sean Goldberg, Daisy Zhe Wang, and Tim Kraska, 2015. A Probabilistically Integrated System for Crowd-Assisted Text Labeling and Extraction. *ACM J. Data Inform. Quality* 0, 0, Article 0 (2015), 21 pages.
DOI: 0000001.0000001

1. INTRODUCTION

In recent years, there has been an explosion of unstructured text data in social networks like Twitter and Facebook, in enterprises via emails and digitized documents, and on the Web. Automatic information extraction (IE) over large amounts of text is important for applications that depend on efficient search and analysis, such as question answering, trend analysis, and opinion mining. Various types of structured information that can be extracted include part-of-speech labels from tweets, named entities from emails, and relational attributes from bibliography citations.

The state-of-the-art approach for IE uses statistical machine learning (SML) models and algorithms such as hidden Markov models (HMM) and conditional random fields (CRF) [Lafferty et al. 2001]. Most current IE systems store the maximum likelihood extraction into a database for querying, but such extractions using even the best mod-

This work was sponsored by many different people FIXME.
Author's addresses: S. Goldberg and D. Z. Wang and C. Grant, Computer & Information Science & Engineering Department, University of Florida; E457 CSE Building, PO Box 116120, Gainesville, Florida 32611-6120.
Emails: seanlgoldberg@gmail.com; daisyw@cise.ufl.edu, cgrant@cise.ufl.edu

DOI: 0000001.0000001

els are still prone to errors even for a task as simple as text segmentation. Consider the following example citations.

Example 1.1.

`Building New Tools for Synthetic Image Animation by Using Evolutionary Techniques Xavier Provot, David Crochemore, Michael Boccara, Jean Louchet Artificial Evolution 3-540-61108-8 Springer`

With no obvious distinction between fields, the model may mislabel `Xavier` *as part of the* `title` *rather than* `author`*. The model may also confuse treat* `Artificial Evolution` *as a* journal *attribute or as the last* author *attribute.* □

A possible means of correcting errors and improving the accuracy of SML-based extraction results uses a human-in-the-loop, manually validating extractions that the machine performs poorly on or is highly uncertain of. When the corrected extractions are re-introduced into the training set, this is known as *active learning*. Human annotation is expensive and time-consuming. Platforms like Amazon Mechanical Turk (AMT) deploy Human Intelligence Tasks (HITs) that make it possible to include humans as a resource during the text extraction process in a *convenient*, *timely*, and *scalable* fashion. An ideal IE system would be one that efficiently leverages the advantages of both human and machine computation [Wang et al. 2012; Quinn et al. 2010] into a single cohesive unit.

For this purpose, we introduce pi-CASTLE: a crowd-assisted SML-based IE system. pi-CASTLE uses a probabilistic database to execute, optimize, and integrate human and machine computation for improving text extraction. The human computation aspect is powered by (AMT) and the machine computation using linear-chain CRF models. pi-CASTLE initially employs a CRF to annotate all input text data. In contrast to other IE systems, however, pi-CASTLE uses a probabilistic data model to store IE results and manage data cleaning. It has the ability to automatically query humans to correct the most uncertain tokens and integrates their responses back into the data model.

By allowing machines to do most of the work and focusing on humans only in the "last mile", pi-CASTLE achieves an optimal balance between cost, speed, and accuracy for IE problems. We address three challenges in the design and implementation of pi-CASTLE: the probabilistic data model, selection of uncertain entries, and integration of human corrections.

First, in order to utilize human computing to reduce errors and uncertainty in SML-based IE results, a probabilistic data model and system is needed to store the uncertain results as well as evidence obtained from the crowd. We use the model described in [Wang et al. 2010], storing both uncertain relations and probabilistic models as first class objects. We also implement a number of user defined functions (UDFs) for statistical inference, question selection, and uncertain data integration over this probabilistic data model to connect the SML and crowd components in pi-CASTLE.

The data cleaning process entails automatically selecting entries likely to be incorrect and generating questions to be pushed to AMT. pi-CASTLE utilizes concepts from information theory such as mutual information to select the most informative entries. This idea of informativeness is defined as those entries likely to reduce the maximum amount of error and uncertainty over the entire database. This is a technically challenging task as tokens, represented as nodes in a graphical model, are not independent, but adhere to the dependence properties modeled by the CRF. Correcting individual tokens can influence other surrounding tokens in the same neighborhood

and pi-CASTLE takes this information into account. Additionally, token redundancy across the database is used to make the most impactful corrections.

Lastly, answers from crowdsourcing services, while generally more accurate, can still contain erroneous and conflicting information if worker qualities are low or if questions are difficult or ambiguous. One of the standard techniques is to take a majority vote among a collection of workers, but such a deterministic approach does not capture the controversy and uncertainty in crowdsourced answers. We introduce a Bayesian probabilistic integration model for combining uncertain answers. This model uses prior knowledge to estimate the confidence of Turker answers and integrate uncertain and conflicting answers in a principled manner.

The following are the key technical contributions of this paper:

- We design and implement a crowd-assisted IE system pi-CASTLE based on a CRF that uses a probabilistic database as an foundation to execute, optimize, and tightly integrate machine computation over CRF models and human computation over crowdsourcing services;
- We develop novel information theoretic techniques that can automatically generate AMT questions to maximally reduce uncertainty and errors in IE result by as much as 60%. Using data from DBLP and PubMed repositories, we show that these techniques lead to orders-of-magnitude fewer questions, reducing cost in improving the overall accuracy of extractions;
- We develop, implement and evaluate a Bayesian probabilistic combination model to integrate the uncertain data from multiple workers. Probabilistic integration of crowd answers can achieve up to 50% over majority voting for relatively difficult questions. Additionally, such integration provides confidence values that can be used for accepting or rejecting a consensus answer. The integrated probabilistic result gives the user the flexibility to tune the trade-off between accuracy and recall (as shown in Section 6) according to the need of the application.

The paper is summarized as follows. Section 2 covers the necessary background in probabilistic databases, conditional random fields, and Amazon Mechanical Turk. We give an overview of the pi-CASTLE system in Section 3. Sections 4 and 5 cover our techniques for performing question selection and integration respectively. Our experiments can be found in Section 6, while Sections 7 and 8 contain the Related Work and Conclusions.

2. BACKGROUND

In this section we briefly describe the necessary background to understanding key components of the pi-CASTLE system.

2.1. Probabilistic Databases

A *probabilistic database* $\mathcal{DB}^p$ consists of two key components: (1) a collection of incomplete relations $\mathcal{R}$ with missing or uncertain data, and (2) a probability distribution F on all possible database instances. These *possible worlds* denoted by pwd(DB^p) represent multiple viable instances of the database. The attributes of an incomplete relation $R \in \mathcal{R}$ may contain deterministic attributes, but also include a subset that are *probabilistic attributes* $\mathcal{A}^p$. The values of $\mathcal{A}^p$ may be present, missing, or uncertain. Each possible database instance is a completion of the missing or uncertain data in $\mathcal{R}$.

2.2. Conditional Random Fields (CRF)

The linear-chain CRF [Lafferty et al. 2001; Sutton and Mccallum 2006], an extension of the hidden Markov model, is a state-of-the-art probabilistic graphical model for solving IE tasks. In the context of IE, a CRF model encodes the probability distribution

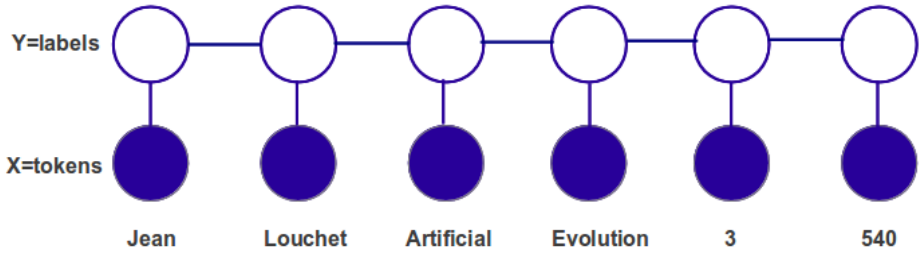


Fig. 1. Example CRF model.

over a set of *label* random variables $y \in \mathbf{Y}$, given the value of a set of *token* random variable $x \in \mathbf{X}$. Assignments to $\mathbf{X}$ are given by $\mathbf{x}$ and to $\mathbf{Y}$ by $\mathbf{y}$. In a linear-chain CRF model, label y_i is correlated only with the previous label y_{i-1} and the corresponding token x_i. The set of these correlations $k \in K$ are represented by the feature functions $\{f_k(y_i, y_{i-1}, x_i)\}_{k=1}^{K}$.

Example 2.1. Figure 1 shows an example CRF model over a subset of the citation string from Example 1.1. Observed (known token) variables are shaded nodes in the graph. Hidden (unknown label) variables are unshaded. Edges in the graph denote statistical correlations. For citations, the possible labels are $Y = \{$title, author, conference, isbn, publisher, series, proceedings, year$\}$. Two possible feature functions of this CRF are:

$$f_1(y_i, y_{i-1}, x_i) = [x_i \text{ appears in a conf list}] \cdot [y_i = \text{ conf}]$$
$$f_1(y_i, y_{i-1}, x_i) = [y_i = \text{ author}] \cdot [y_{i-1} = \text{ title}]$$

Let $\{f_k(y_i, y_{i-1}, x_i)\}_{k=1}^{K}$ be a set of real-valued feature functions, and $\Lambda = \{\lambda_k\} \in R^K$ be a vector of real-valued parameters, a CRF model defines the probabilistic distribution of segmentations $\mathbf{y}$ given a specific token sequence $\mathbf{x}$:

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z}\exp\{\sum_{i=1}^{T}\sum_{k=1}^{K}\lambda_k f_k(y_i, y_{i-1}, x_i)\}, \tag{1}$$

where Z is a standard partition function that guarantees probability values between 0 and 1.

2.3. Inference Queries over a CRF Model

There are three types of inference queries used in pi-CASTLE.

Top-k Inference: The top-*k* inference computes the segmentations with the top-*k* highest probabilities given a token sequence $\mathbf{x}$ from a text-string d. The Viterbi dynamic programming algorithm [Forney 1973] is the key algorithmic technique for CRF top-*k* inference.

Constrained Top-k Inference: Constrained top-*k* inference [Kristjansson et al. 2004] is a special case of traditional top-*k* inference. It is used when a subset of the token labels has been provided (e.g., via a user interface such as Amazon Mechanical Turk). Let $\mathbf{s}$ be the evidence vector $\{s_1, \ldots, s_T\}$, where s_i is either NULL (i.e., no evidence) or the evidence label for y_i. Constrained top-*k* inference can be computed for a variant of the Viterbi algorithm which restricts the chosen labels $\mathbf{y}$ to conform to the evidence $\mathbf{s}$.

Marginal Inference: Marginal inference computes a marginal probability $p(y_t, y_{t+1}, \ldots, y_{t+k}|\mathbf{x})$ over a single node's label or a sub-sequence of nodes [Sutton and

Mccallum 2006]. The Forward-Backward algorithm, a variation of the Viterbi algorithm is used for such marginal inference tasks. In pi-CASTLE, marginal inference is primarily employed over a single node corresponding to the marginal label distribution for individual tokens.

2.4. Crowdsourcing

Platforms such as Amazon Mechanical Turk (AMT) and Crowdflower have made the leveraging of human computation and intelligence at large scale both cost effective and time efficient. Workers, or"Turkers", complete various jobs and earn money on a per-job basis. The range of "microtasks" available for completion may be anything from simple image or text annotation tasks, completing surveys, or ranking search results, usually within a few minutes and at a cost of a few cents per task. These jobs generally don't require any special training or domain expertise, allowing a large workforce from all over the world to be utilized. Amazon does not publish current statistics about the marketplace, but it contained over 200,000 [AWS 2006] Turkers in 2006, and by all estimates, has grown dramatically since then [Ross et al. 2010].

3. SYSTEM OVERVIEW

Figure 2 outlines the basic architecture of the pi-CASTLE system, which is comprised of four main components: 1. CRF Extraction & Inference, 2. Question Selection, 3. HIT Management, and 4. Uncertain Data Integration. The arrows chart the flow of data within and between different components. Overall, data flows through the four components in order. First, SML models are applied to perform automatic text extraction and labeling. The uncertain extractions are stored in a probabilistic database. Second, a set of ranked questions are selected from uncertain IE results. Third, the HIT manager formulates and pushes these questions to the crowd and retrieves the answers. Fourth, the Turker answers are combined probabilistically and integrated back into the database, improving the initial IE results from the CRF.

In this section we briefly outline each of the system's components and how they are related, as well as how data is specifically stored in the data model. While we use existing techniques for 1. CRF Extraction and 3. HIT Management, we develop novel techniques for 2. Question Selection and 4. Uncertain Data Integration in Section 4 and Section 5.

3.1. Feature Extraction & Inference

The initial extraction and labeling of unstructured text is handled by the Extractor & Inference module. Parameter estimation (learning) of the model is done in advance using a labeled data set. Maximum likelihood labels and marginal probabilities can be inferred from the CRF model and the uncertain label or extraction tables in a probabilistic database. We adopt the probabilistic data model outlined in [Wang et al. 2010].

Unstructured text is treated as a set of documents or text-strings $\mathcal{D}$. Each document $d \in \mathcal{D}$ has a substructure comprising a set of tokens t_i^d, where $i \in \{1, \ldots, N\}$ and N is the length of the string (document). These tokens are stored in the relational format with attributes (*docID, pos, token*). In addition, a probabilistic attribute label^p is also included, whose value can be inferred by the model. The schema of the uncertain IE results generated from the CRF is:

$$\text{CRFLBLTBL(docID, pos, token, label}^p\text{)}$$

The probabilistic label attribute label^p maps to a random variable within a grounded CRF model. Queries such as MARGINAL and TOP-K extraction can be run over the collection of probabilistic labels.

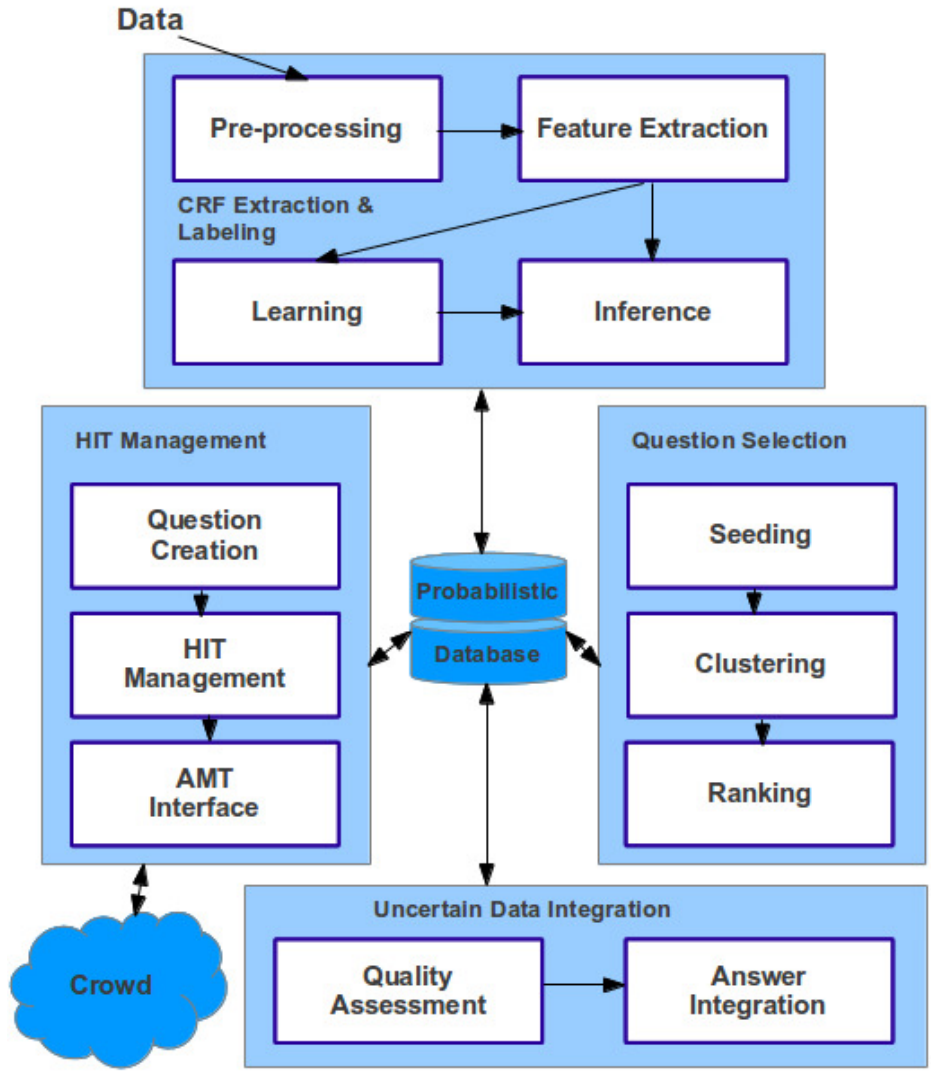


Fig. 2. Architecture of the pi-CASTLE system.

3.2. Question Selection

The novelty of pi-CASTLE is that it improves the results of automatic text labeling and extraction with additional evidence collected from the crowd. Utilizing data uncertainty information, it automatically selects questions over tokens that are more uncertain and likely to contain errors. This process of *Question Selection* is illustrated by the right-hand module in Figure 2, which applies three consecutive techniques over the uncertain CRFLBLTBL table: Filtering, Clustering, and Ranking. Filtering selects an initial set of tokens to be corrected based on mutual information; Clustering, groups those filtered tokens with similar attributes together using a trigram contextual model to reduce redundancy; Ranking, performs a total ordering over token clusters. Cluster assignments and rankings for all tokens passing the initial Filtering operation are stored in a selection table.

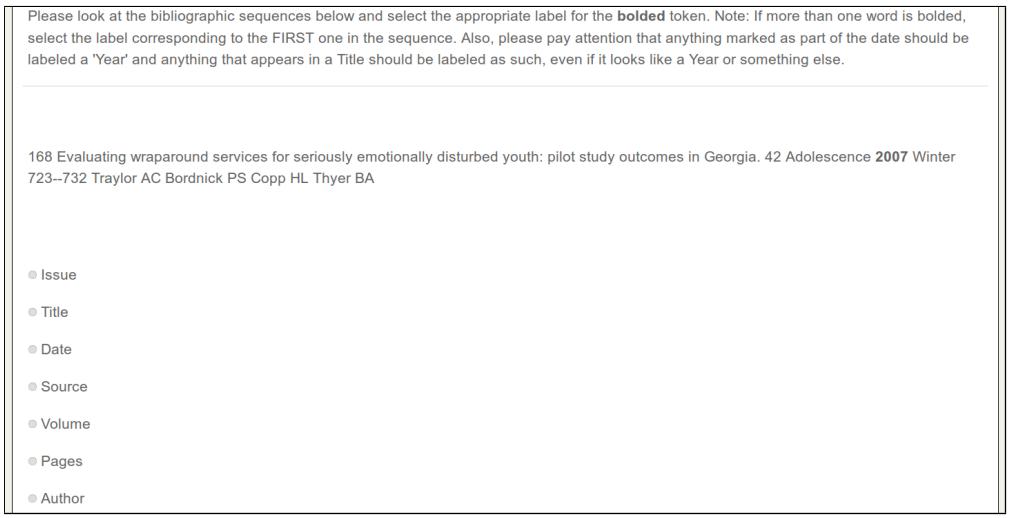


Fig. 3. Sample Mechanical Turk HIT Interface

SELECTIONTBL (docID, pos, clusterID, ranking)

The selection of questions is performed with a fixed budget and optimized for the cost of each question. We focus on uniform, fixed-cost questions, though our methods may be easily extended to multiple question types incurring different costs.

3.3. HIT Management

The HIT manager has the responsibility of taking the rows of SELECTIONTBL, converting them into AMT questions (i.e., HITs), and posting those questions onto Amazon Mechanical Turk. The Question Creation subcomponent generates an XML template for a multiple choice question for each selected token. An example interface is shown in Figure 3. The entire text document (in this case a citation) is shown and the queried token is bolded. Users select from the set of all labels the one they believe belongs to the bolded token. The Question Creator has the ability to assign multiple questions to a single HIT, this ensures Turkers answer the same subsets of questions.

The HIT manager also allows an administrator to set specifics such as the price of each HIT, length of posting, and the number of Turkers assigned to each HIT. There is an AMT interface that uses an API to handle both the posting of questions and retrieval of answers. Both questions and answers are recorded and stored in their corresponding base tables, the primary key being a HITID supplied by Amazon when posting.

POSTTBL(docID, pos, HITID)

TURKERLBLTBL(HITID, workerID, label)

3.4. Uncertain Data Integration

The goal of integration in pi-CASTLE is to use both the crowd supplied labels and the machine generated labels to infer the annotation distribution. The first step is to assess the quality of each Turker using the well-documented method introduced by Dawid & Skene [Dawid and Skene 1979]. This tells us how to appropriately weight Turkers of various quality in the integration process.

TURKERCONFTBL(workerID, quality)

pi-CASTLE employs a novel probabilistic model for uncertain data integration over crowdsourced answers using the machinery of Bayesian conditional probability. The CRF annotation (Section 2.2) is a prior and human edits from TURKERLBLTBL weighted by values in TURKERCONFTBL are treated as additional evidence to yield a posterior annotation distribution.

CRWDLBLTBL(docID, pos, label^p)

As a last step, the responses in CRWDLBLTBL are integrated into the probabilistic database by constraining the node's marginal probability to the combined posterior. Performing constrained inference has the ability to correct additional token labels based on correlations in the CRF model, further improving overall accuracy. In the next two sections, we go into details of the new techniques developed for question selection and uncertain data integration.

4. QUESTION SELECTION

The problem of question selection is similar to that found in active learning where select examples are chosen from a pool of unlabeled data to be annotated based on some querying strategy. While active learning has been applied to the sequential learning domain [Settles and Craven 2008; Cheng et al. 2008] the financial and temporal cost of labeling an entire sequence (document) is not amenable to AMTs microtask framework. Additionally, we found documents to contain sparse labeling errors and annotation of an entire document represents unneeded redundancy.

This necessitates tasks where examples can be partially labeled over specific tokens. Since these examples cannot be used to re-train the supervised learning algorithm without a complete annotation, feedback is no longer used to improve the model, but to reduce the posterior uncertainty in the results. By re-running inference with selected tokens constrained to their crowd-annotated values, we can drastically improve accuracy in a cost effective way that does not require the labeling of every token.

To properly select tokens, we need a way of properly assessing their *information value*. For a token $\mathbf{x_i}$ with labels $\mathbf{y_i}$, let $\phi(\mathbf{x_i})$ be a function that maps each token to its information value according to some strategy. A standard technique in active learning is to choose examples with the highest entropy, or for a sequence, the highest average marginal entropy of the individual nodes [Settles and Craven 2008]. This **token entropy (TE)** is defined as

$$\phi^{TE}(\mathbf{x_i}) = -\sum_{l=1}^{L} P(y_i = l) log P(y_i = l), \tag{2}$$

where the sum is over the range of labels L and $P(y_i = l)$ is the marginal probability of token $\mathbf{x_i}$ given label l.

Token entropy quantifies the uncertainty in the prediction result for each individual token. While this method works well in practice for sequence models, the dependence properties shared between tokens increase the complexity of the selection process. Indeed, labels are not chosen greedily by their highest marginal probabilities, but using the dynamic programming Viterbi algorithm where suboptimal local choices can lead to a correct global solution.

In short, marginal probabilities and their corresponding entropy are not telling us the whole story. We develop two new techniques for maximizing the information value of tokens sent to the crowd for labeling. First, we exploit *mutual information* (MI) to select those tokens whose result will have the greatest significance on its neighbors

ALGORITHM 1: QuestionSelect

input : Set of all tokens $\mathcal{T}$
output: Ranked set C of maximum information clusters

```
Initialize selected token set S;
Initialize cluster set C;
//Filtering;
foreach t ∈ T do
    i ← t.docID;
    if S(i) = NULL then
        S(i) = t;
    else if S(i).MI < t.MI then
        S(i) = t;
//Clustering;
Load all tokens in S into queue Q;
foreach t ∈ Q do
    foreach cluster c ∈ C do
        if c.text = t.text &
        c.label = t.label &
        c.prevLabel = t.prevLabel &
        c.postLabel = t.postLabel then
            Add t to cluster c;
            c.totalInfoGain ← c.totalInfoGain +t.totalInfoGain;
    if t not added to a cluster then
        Initialize new cluster c;
        c.text ← t.text;
        c.label ← t.label;
        c.prevLabel ← t.prevLabel;
        c.postLabel ← t.postLabel;
        Add c to cluster set C;
        c.totalInfoGain ← t.totalInfoGain
//Ranking;
SORT clusters c ∈ C by c.totalInfoGain;
```

within a document. Additionally, we use *density estimation* to select tokens with the greatest redundancy across multiple documents. These techniques have been previously studied in the active learning domain [Xu et al. 2007; Zhao and Ji 2010] particularly for document retrieval, but to our knowledge have not been applied to a partial labeling scheme over a probabilistic sequence model.

Algorithm 1 shows the psuedo-code for our entire selection method. The filtering step assumes each token already has a mutual information score associated with it. We iterate through all tokens, keeping only the maximum MI tokens for each document. The clustering step iterates through filtered tokens, adding those with similar properties to the same cluster and creating new clusters as necessary. The final cluster set is then sorted where the top-k may be drawn. We now delve into each of these steps in more detail, based on a review of previous work found in [Goldberg et al. 2013].

4.1. Filtering by Mutual Information

Mutual information (MI) is an information theoretic measure of the mutual dependence shared by two random variables (RVs). Specifically, for two RVs $\mathcal{X}$ and $\mathcal{Y}$, the

mutual information is defined in terms of entropy as

$$I(\mathcal{X};\mathcal{Y}) = H(\mathcal{X}) + H(\mathcal{Y}) - H(\mathcal{X},\mathcal{Y}). \tag{3}$$

It represents the difference between the joint entropy $H(\mathcal{X};\mathcal{Y})$ and the individual entropies $H(\mathcal{X})$ and $H(\mathcal{Y})$. Intuitively, MI describes the reduction of uncertainty of one RV given knowledge of another. Random variables that are highly correlated will have small joint entropies whereas they are equivalent to the sum of individual entropies if the variables are independent.

If we plan to run the inference algorithm over a partially labeled set, we need to determine precisely which variables will give the most information for the remaining ones in the sequence. This entails calculating the mutual information of every node against all others. The query strategy then becomes

$$\begin{aligned}\phi^{MI}(\mathbf{x_i}) = H(\mathbf{x_i}) + H(\mathbf{x_1},\ldots,\mathbf{x_n}\backslash\mathbf{x_i}) \\ - H(\mathbf{x_1},\ldots,\mathbf{x_n})\end{aligned} \tag{4}$$

where $H(\mathbf{x_1},\ldots,\mathbf{x_n}\backslash\mathbf{x_i})$ is the entropy of all nodes except for $\mathbf{x_i}$. This strategy is computationally expensive to perform on every node in every sequence. Instead we invoke a correlation of the data processing inequality, which states that information processed along a Markov chain cannot increase, i.e., for a chain $X \to Y \to Z$, where X, Y, Z are states in a chain.

$$I(X;Y) \geqslant I(X;Z) \tag{5}$$

We approximate equation 4 by utilizing its most informative neighbors, those just to the left and right in the chain. The choice becomes selecting those tokens $\mathbf{x_i}$ likely to have the greatest impact on its most immediate neighbors $\mathbf{x_{i-1}}$ and $\mathbf{x_{i+1}}$

$$\begin{aligned}\phi^{MIapprx}(\mathbf{x_i}) &= I(\mathbf{x_{i-1}};\mathbf{x_i}) + I(\mathbf{x_i};\mathbf{x_{i+1}}) \\ &= H(\mathbf{x_{i-1}}) + 2H(\mathbf{x_i}) + H(\mathbf{x_{i+1}}) \\ &\quad - H(\mathbf{x_{i-1}},\mathbf{x_i}) - H(\mathbf{x_i},\mathbf{x_{i+1}})\end{aligned} \tag{6}$$

The entropies in equation 6 can be efficiently computed using the *forward-backward algorithm* [Rabiner 1989] to compute the marginal and joint probabilities, then the entropy calculated in the standard fashion.

Mutual information can be useful in determining the impact a node's observation has on other nodes within an individual sequence, but tells us nothing about the distribution of tokens across all documents. If we want to optimize our selection strategy, especially for a batched selection process, we should additionally incorporate a tokens ferquency and its uncertainty.

4.2. Clustering by Information Density

A major efficiency drawback to many active learning schemes is that they are myopic. An instance is selected for labeling, the model is re-trained, and the process is iteratively repeated. The iterative method fails to harness the parallelizability of the crowd ecosystem. Alternativley, the faculty select tokens in batch and query the crowd at once rather than in a sequential manner. One factor that can compromise effectiveness is if there are similar token instances in the batch, as querying the label of two similar instances is equivalent to querying either of them and applying the label to both.

We propose a scheme to cluster those tokens that should be labeled similarly and address two key issues. First, final batch must be *diverse* and contain only one token from each cluster. The batch must also be *dense* and contain the largest clusters whose labeling will have the greatest effect.

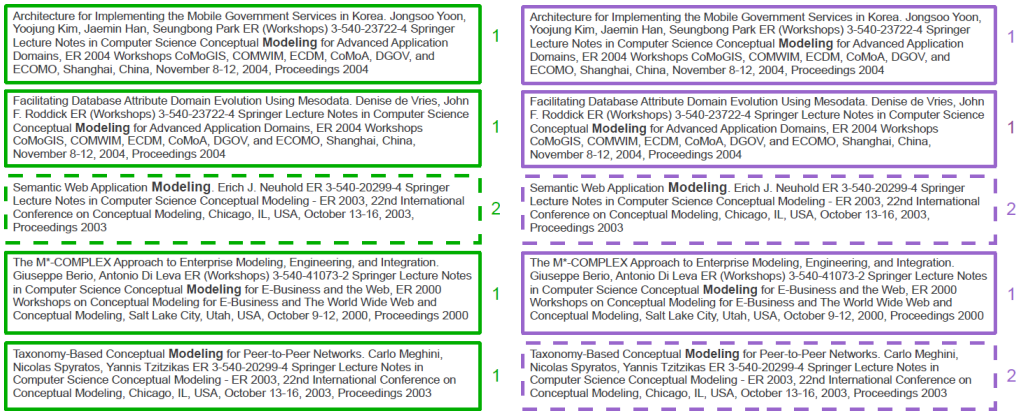


Fig. 4. Clustering for the token "Modeling" shown over five example citations with each line type denoting a different cluster. The level of clustering and error rate differ among whether (a) token trigrams or (b) label trigrams are used for clustering.

In order to cluster tokens appropriately, we must define a meaningful similarity measure between them. A naive approach would cluster strictly those tokens which are equivalently out of context. This is less than desirable in our text segmentation scenario where location of the token in the document matters. Context is also important in other IE problems such as named entity recognition (NER) where homonyms with different meanings and subsequent labelings would be incorrectly grouped together.

Thus we are led to consider a **token trigram** clustering model, where tokens with similar neighbors are clustered together. Let $\mathbf{x_i}$ be a token at position i in some document. Together with its left and right neighbors we form the trigram $(\mathbf{x_{i-1}}, \mathbf{x_i}, \mathbf{x_{i+1}})$. Despite being clustered as a trigram, the selection process selects the single middle token to query the crowd. We take the intuitive assumption that middle tokens $\mathbf{x_i}$ belonging to the same trigram are highly likely to share the same context and ought to be labeled the same. For each trigram cluster in the corpus, a single "representative token" with the highest mutual information (or token entropy) is selected and crowd-annotation applied to all the middle tokens in the cluster. Figure 4(a) shows an example token trigram clustering.

In domains such as bibliographic citation IE, many-token phrases such as common proceedings names and conference locations appear throughout multiple documents. Common trigrams when compared across all tokens, however, are relatively infrequent. The token trigram model produces very few classification errors, but non-singleton clusters are very sparse.

There is more to the notion of context than just duplicate words appearing together. Words used in a similar "sense" and likely to share the same label may use many different words which contextually mean the same thing. There are many ways to label how a word is used that form the fundamental backbone of NLP annotation tags, such as part-of-speech (POS), entity tags, segmentation tags, dependency parses, etc.

A token $\mathbf{x_i}$ has a set of associated labels $\mathbf{l_{i,j}}$, where i again denotes label position and j some numerical representation of the classifier type. For example, $\mathbf{l_{i,0}}$ might be the POS tag associated with $\mathbf{x_i}$ while $l_{i,1}$ might be a segmentation tag. A **label trigram**

clustering model consists of tokens that share some specified set of label trigrams. One possible cluster would be $(x_i, (l_{i-1,1}, l_{i,1}, l_{i+1,1}))$, which groups individual tokens labeled with the same segmentation tag and sharing left and right neighbors labeled the same. One requirement for all label trigram clusters is that that the individual tokensx_i should still be the same. Figure 4(b) illustrates an example of label trigram clustering.

While these labels are themselves the uncertain output of machine learning classifers, our experiments show contextually similar tokens are also similarly mislabeled and still cluster appropriately. Overall, the label trigram model increases the recall and amount of clustering, but at the expense of a greater rate of classification error compared to the token trigram model.

Both trigram models correspond to a mapping of tokens to a lower dimensional space where tokens sharing the same trigram properties are mapped to the same point. Selecting the largest token clusters is equivalent to selecting the "highest density" instances according to the data distribution, a technique that has shown positive yield in traditional active learning [Guo and Greiner 2007].

4.3. Ranking by Total Information Gain

Given a limited budget of questions, clusters should be ranked to facilitate selection of the top-*k*. We experimented with three different ranking schemes: ranking by mutual information score of a cluster's representative token, ranking by cluster size, and ranking by total information gain. We define the total information gain of a cluster to be the sum of all mutual information scores of all tokens that belong to a cluster. A comparison of the three ranking approaches can be found with the experiments in Section 6.

5. UNCERTAIN DATA INTEGRATION

Many systems that utilize crowdsourced annotation tasks treat the results as ground truth due to the overconfidence of having a human workforce. In practice, Turker responses have a tendency to be noisy or conflicted, thus increasing the difficulty of establishing veracity. This means that results from the machine learning model are uncertain as well as the crowdsourced answers. pi-CASTLE is equipped with a means of managing both types of uncertainties and integrating them into a consistent result for maximum accuracy. In this section we motivate the need for a probabilistic approach to quality control and truth discovery and describe the generative Bayesian model implemented in pi-CASTLE.

5.1. Probabilistic Evidence

The de facto technique for performing quality control in a crowdsourced environment is to redundantly post the question to a number of Turkers and take a majority vote. This naive technique is ineffective approach and there are a number reasons for wanting to pursue a probabilistic means of data integration.

First, not all Turkers have the same capabilities and not all votes should be treated equally. pi-CASTLE uses the expectation maximization model of Dawid & Skene [Dawid and Skene 1979] that identifies the individual reliabilities of Turkers and the confidence of their answers [Ipeirotis et al. 2010]. This approach has proven effective for reducing the influence of malicious workers and spammers, however, it is less suited to ambigious or difficult questions that justifiably split the workforce.

Results are generally truncated to the maximum vote-getter and treated as truth. Information pertaining to the ambiguity of the response is thrown out. pi-CASTLE is one of the first systems to integrate crowdsourced answers into the system in a principled manner not by replacement, but by probabilistic combination. pi-CASTLE

believes that truth should be inferred and maximum votes not taken as fact. The system treats Turker responses as individual pieces of evidence and arrives at a label distribution combining all crowdsourced information with additional information from the machine model.

We now describe pi-CASTLE's approach in the context of the Bayesian theory of evidence.

5.2. Bayesian Integration Model

Bayesian models of evidence are initiated with some prior and updated to a posterior accounting for additional evidence. pi-CASTLE utilizes this approach by treating the machine output as the prior and human responses as the additional evidence.

Let $A^q = \{A_1^q, \ldots, A_K^q\}$ be a set of categorical random variables corresponding to the answers received from K Turkers for question $q \in Q$. The random variable L follows a categorical distribution over the label space, L_i representing the i^{th} label, and $P(L)$ is the prior estimate of the label distribution for a specific token. The choice of prior comes from the token's original CRF marginal distribution, which means we start from the machine labeling and update using new crowdsourced information. The integration problem is to find the posterior distribution $P(L^q|A_1^q,...,A_K^q)$ conditioned on the answers provided by the Turkers. This can be calculated using Bayes's Rule:

$$P(L^q|A_1^q, ..., A_K^q) = \frac{P(A_1^q, ..., A_K^q|L^q)P(L^q)}{\mathcal{Z}}, \tag{7}$$

where $\mathcal{Z} = \Sigma_i(P(A_1^q, ..., A_K^q|L_i^q)P(L_i^q))$. The term representing the evidence, $P(A_1^q, ..., A_K^q|L^q)$, is the probability the Turker answers were generated from a specific true label. pi-CASTLE's Bayesian model assumes Turker quality is an adequate measure of their agreement with the true label,

$$P(A_1^q, ..., A_K^q|L^q) = \prod_k P(A_k^q|L^q) \tag{8}$$

$$P(A_k^q = a|L^q = l) = \mathbb{1}_{a=l} * Q_k + (1 - Q_k) * \frac{1}{|L|} \tag{9}$$

where a and l are values drawn from the label space and Q_k is the quality of the kth worker. Equation 8 follows from all Turker answers being independent of each other and equation 9 simply restates our assumption about the use of Turkery quality Q_k. If the answer matches the label l, the first term on the right hand side is the probability the Turker is reliable and answers the question truthfully. The second term incorporates the probability they are unreliable or a spammer and through *random guessing* finds the correct answer with probability $1/|L|$, $|L|$ being the number of possible labels. If they don't match, we have the probability the Turker is unreliable, $1 - Q$, and the probability a random guess produces an incorrect answer, $(L - 1)/L$.

The full model to obtain the probability of a label L^q given turker answers $A_1^q \ldots A_K^q$ is

$$\begin{aligned} &P(L^q = l|A_1^q = a_1, ..., A_K^q = a_k) = \\ &\quad \frac{1}{Z} P(L^q = l) \prod_k \left(\mathbb{1}_{a_k=l} * Q_k + (1 - Q_k) * \frac{1}{|L|}\right) \end{aligned} \tag{10}$$

Using equation 10 for all possible labels l and renormalizing produces a new posterior distribution accounting for both the initial ML extracted result and evidence gath-

ered from the crowd. The product can even be extended and updated as new evidence comes in over time allowing for a persistently updateable system of truth management. This approach has been considered using a different technique in [Sheng et al. 2008]. While currently evidence is designed to come from the crowd in pi-CASTLE, there is no explicit restriction preventing future updates from incorporating evidence from a number of different extractions as well as the crowd. We close this section with an example to better illustrate the integration algorithm.

Example 5.1. Assume a binary question is answered by three Turkers. Turker A has quality 0.9 and answers with label 0, Turker B has quality 0.4 and answers with label 1, and Turker C also has quality 0.5 and also responds with label 1. The prior CRF marginal probability over {0,1} is {0.5,0.5}. A majority vote among the three Turkers would give label 1, even though both are of relatively low quality. The Bayesian combination can be found from equation 10,

$$\begin{aligned} &P(L=0|A,B,C) \\ &\quad = P(L=0)P(L=0|A)P(L=0|B)P(L=0|C) \\ &\quad = (0.5)\left(0.9+(0.1)(\frac{1}{2})\right)(0.6)(\frac{1}{2})(0.5)(\frac{1}{2}) \\ &\quad \approx .036 \end{aligned} \tag{11}$$

$$\begin{aligned} &P(L=1|A,B,C) \\ &\quad = P(L=1)P(L=1|A)P(L=1|B)P(L=1|C) \\ &\quad = (0.5)(0.1)(\frac{1}{2})\left(0.4+(0.6)(\frac{1}{2})\right)\left(0.5+(0.5)(\frac{1}{2}\right) \\ &\quad \approx .013 \end{aligned} \tag{12}$$

After combining and normalizing, the final distribution over {0,1} is {0.73,0.27}. Instead of truncating and taking label 0 as truth, this binary distribution would be retained in the system. If task the budget were to increase, this question might be chosen again and the distribution refined even further.

5.3. Data Integration Pipeline

Here we briefly summarize for clarity the entire data integration pipeline in Algorithm 2. After questions have been selected based on the techniques described in Section 4, the HIT Management module produces questions and posts them to Amazon Mechanical Turk, the results of which are automatically retrieved and stored in the database.

Questions are posted as HITs in batches of size N. The fixed batch size ensures that enough Turkers answer the same questions for the EM process. For a set of M Turkers answering an HIT of N questions, we use Dawid & Skene to estimate accuracies for each Turker. These accuracies are saved in a table in the database so if a worker performs more than one HIT their accuracies for each one are averaged together. Recording the history of turker accuracy is an additional benefit of having a persistent system.

For each question, the Turker accuracies and answers are integrated with model output using the Bayesian framework described in the previous section. This produces a distribution over the label space. Since queried tokens were pulled from clusters as per Section 4, the answer corresponds to the selected "representative token". The entire reason for clustering, however, was to map this distribution back to all the tokens belonging to the same cluster and satisfying some similarity threshold. Therefore, all token accuracies belonging to a selected cluster get updated.

ALGORITHM 2: Probabilistic Integration

input : Set of clusters C,
CRF prior for each cluster
output: Labeled document d_labeled

```
//Query AMT for responses;
ans ← queryAMT();
//Calculate Turker qualities;
qual ← Dawid_Skene(ans);
//Compute posterior distribution of answers;
combo_ans ← Bayesian_integrate(pr, ans, qual);
//constrain CRF for all tokens in clusters;
foreach cluster c ∈ C do
    foreach tok ∈ cluster c do
        (docID, pos) ← getToken(tok);
        CRF(docID,pos) ← combo_ans(c);
        RunViterbi(CRF(docID,pos));
```

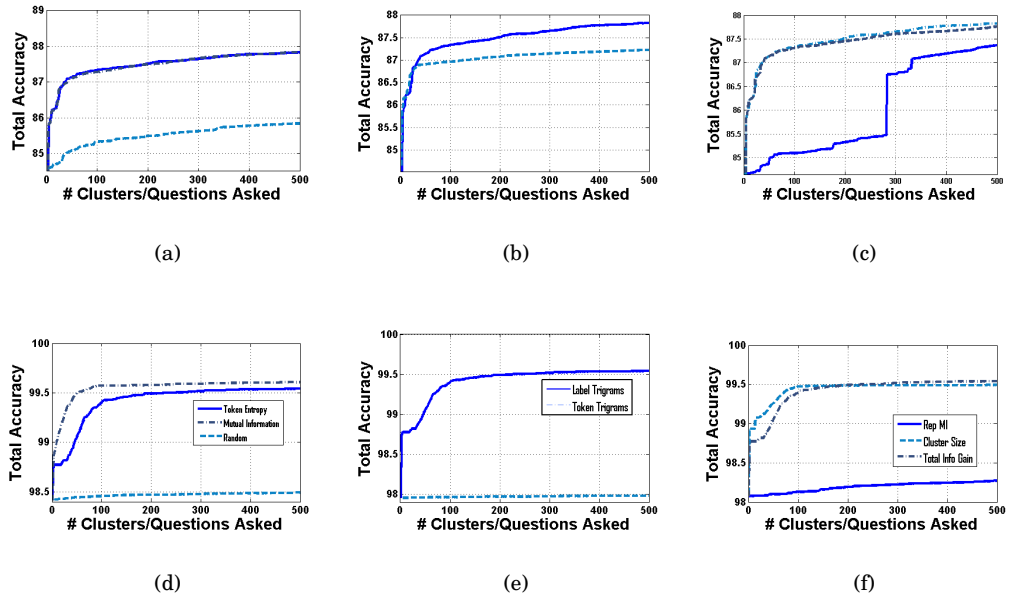


Fig. 5. Accuracy improvements for various selection parameters. The first row is for the DBLP data set and the second is for PubMed. From the left to right the columns compare the filtering, clustering, and ranking methods mentioned in this paper. The parameter legend applies to both plots in each column. Where not explicily being compared, the defaults are Mutual Information, Same Label, and Total Information Gain, respectively.

For all documents containing tokens that were updated, constrained Viterbi inference is performed, the constraint being that the incoming transition probability is reflective of the new updated marginal distribution. The final labeling represents an efficient integration of both human and machine computation.

6. EXPERIMENTS

In this section we demonstrate the effectiveness of our selection and integration approaches on sets of synthetic and real data.

6.1. Setup and Data Sets

We extracted 14,000 labeled citations from the DBLP[1] and 500,000 from the PubMed[2] databases. For unlabeled testing data, we removed the labels and concatenated text from each of the available fields. Order of fields was randomly mixed in keeping with real-life inconsistency of citation structure.

To test our selection methods without the effects of integration accuracy, we supplemented the ground truth values in place of Turker answers when running constrained inference, equivalent to a workforce of perfect Turkers. To evaluate the integration techniques, we experimented with simulated answers as well as real answers from AMT. For answer simulation we sampled the accuracy Q_i of Turker i from a normal distribution $\mathcal{N}(\mu,\ \sigma)$ whose parameters we allowed to vary. Where not listed explicitly, we used the values $\mu = 0.5$ and $\sigma = 0.3$. We sampled the ground truth value with probability Q_i and with probability $1 - Q_i$ a random guess was used instead.

We also performed a set of experiments on Amazon Mechanical Turk using both datasets while varying question difficulty. Each HIT consisted of 10 bibliographic token annotation tasks at \$0.10 per HIT, or \$0.01 per token. From the DBLP set we produced a set of 500 HITs and required users to pass a qualification test before being allowed to answer questions. We produced an additional "hard" set where the same questions were stripped of any punctuation and no qualification test was used. The motivation was to develop questions of a more ambiguous nature likely to cause more conflicts. We coined this set DBLP-hard. The PubMed set consisted of 500 questions and also required no such qualification test. Some of the more ambiguous questions were selected to further test confusion among Turkers and study conflict. In all sets 5 Turkers were assigned to each HIT and quality was assessed on a per HIT basis using the Dawid & Skene method.

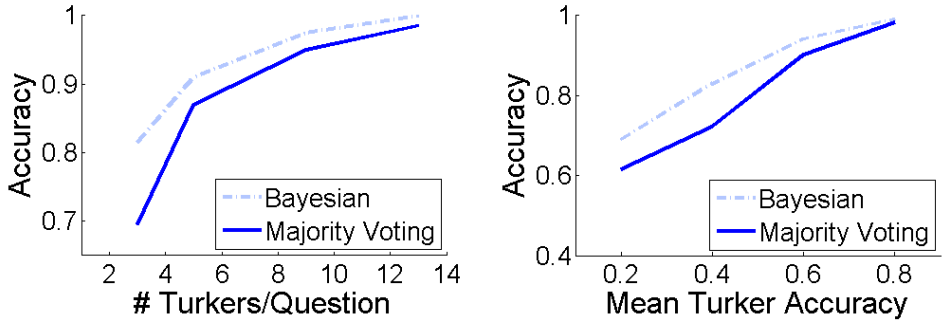
6.2. Selection Experiments

Figures 5(a)-5(f) contain experiments comparing our various selection algorithms by detailing the accuracy improvements after a specific number of questions have been asked. Tokens were selected based on the pipeline of Algorithm 1 using a specific combination of filtering, clustering, and ranking approaches. The choices for filtering were random selection, highest token entropy, and highest mutual information. Clustering compares the token trigram and label trigram methods. Finally, ranking looks at ordering by highest representative token mutual information, cluster size, and total information gain. Where not explicitly listed, the default values were highest mutual information, label trigram, and total information gain, respectively.

To eliminate the possible variability of crowd answers and study purely the ability of the selection algorithms to identify highly inaccurate and highly impactful tokens, we supplied the ground truth as the answer after selection. The cluster representative's answer (label) to each question (token) is applied to all subsequent tokens in its cluster. A constrained Viterbi inference algorithm runs over all documents that are supersets of tokens belonging to question clusters. The accuracy value in each figure represents the final token accuracy after running constrained inference.

The results of Figures 5(a) and 5(d) show a marked increase in accuracy using information theoretic techniques after just a few questions compared to selecting at ran-

[1]http://kdl.cs.umass.edu/data/dblp/dblp-info.html
[2]http://www.nlm.nih.gov/bsd/licensee/medpmmenu.html



(a) (b)

Fig. 6. Comparison of integration methods versus (a) number of Turkers per question and (b) average Turker quality for a synthetic set of 500 questions. Where not stated the default number of Turkers was 5 and mean accuracy was 0.5.

dom. After about 25 questions in the DBLP set, both information theoretic methods produce a 5-fold increase in accuracy compared to random and reduce the overall error by about 15%. The initial PubMed accuracy was much better, but still saw a 66% reduction in error using our selection methods after about 50 questions. This is significant when considering DBLP contained 36k individual tokens and PubMed 3.3m tokens. Each question only queries for the label to a single token, so we are seeing large reductions in error querying on significantly small fractions of the data.

The trigram methods are compared in Figures 5(b) and 5(e), where label trigrams outperform token trigrams in both data sets. The knee in the PubMed set is due to several large clusters of similar tokens being selected first. There are more likely to be misclassification errors using label trigrams, but experiments show the increased coverage outweighs this concern. Finally, ranking metrics are compared in Figures 5(c) and 5(f). The presence of large clustering gives added weight to both methods that take cluster size into account. Information gain, the sum of a mutual information scores of all tokens in a cluster, takes into account both cluster size and informativeness and excels in both data sets.

6.3. Integration Experiments

To measure the efficacy of our integration methods, we conducted experiments over both real and synthetically generated data. The parameters of the synthetic model were listed in Section 6.1 and allowed us to vary both the number of Turkers answering questions and their individual quality values.

Figure 6(a) shows the gain in accuracy that comes with using a probabilistic combination method as opposed to taking a simple majority vote as we increase the number of Turkers answering each question. The mean worker accuracy is 0.5 in these results and the prior used in the Bayesian method is uniform over a label space of 8 labels. While both methods increase monotonically as expected, the Bayesian method produces the best results for low redundancy and attains 100% accuracy before majority voting.

The availability of high or low quality workers is certain to affect the information gathering, so in Figure 6(b) we compare the accuracy of answers as we vary the quality. Variation is achieved by shifting the center of the Gaussian which produces worker

Integration Method	Accuracy
Majority Vote	97.5
Bayesian	97.5

(a) DBLP

Integration Method	Accuracy
Majority Vote	75.8
Bayesian	77.3

(b) DBLP-hard

Integration Method	Accuracy
Majority Vote	14.3
Bayesian	20.9

(c) PubMed

Fig. 7. Amazon Mechanical Turk accuracies for a set of 500 standard (DBLP) and two sets of 500 difficult (DBLP-hard & PubMed) questions.

quality values. We initially set it at 0.2 and shifted to a maximum of 0.8. As before, the Bayesian approach shows that combining probabilistically, even when source accuracies are so low as to be just slightly better than random, produces large gains in answer accuracy.

In addition to the synthetically generated Turker data, we performed a set of experiments using our Amazon Mechanical Turk interface. As described in the introduction to this section, 500 questions from the PubMed and DBLP data sets were provided to Turkers, with both sets containing selected questions to measure the extremes of Turker viability. The DBLP set contained questions ranked using highest mutual information filtering, label trigram clustering, and total information gain ranking. It was presented twice to Turkers, the second time being stripped of all punctuation to make the classification problem more difficult. After integration the maximum likelihood value was compared to the ground truth to determine accuracy. Figure 7 shows a comparison for relatively easy DBLP set compared to the harder sets when using either majority voting or Bayesian combination. The more difficult the question and the lower the quality of the incoming work the larger the increase in benefit with the Bayesian approach, even attaining a nearly 50% increase in accuracy for the difficult PubMed set. The main cause for the extreme difficulty in this set was due to a number of ambiguous numerical fields like journal numbers, issue numbers, etc.

In addition, one can improve accuracy even further by only accepting those answers for which there is high confidence. Our probabilistic approach lends itself to a natural confidence indicator in the entropy of the combined distribution. The trade-off between accuracy and recall is a new functionality compared to deterministic integration which adds flexibility for users to tune the system according to the needs of their application (ie. high accuracy vs. high recall). Figure 8 shows how the accuracy varies for the DBLP-hard set based on how many answers we accept as determined by a confidence threshold. The initial accuracy was 77%, but by retaining only the top two-thirds of the answers in this set, we can keep the total accuracy above 90%. Previous work [Sheng et al. 2008] has attempted selectivity by answer entropy.

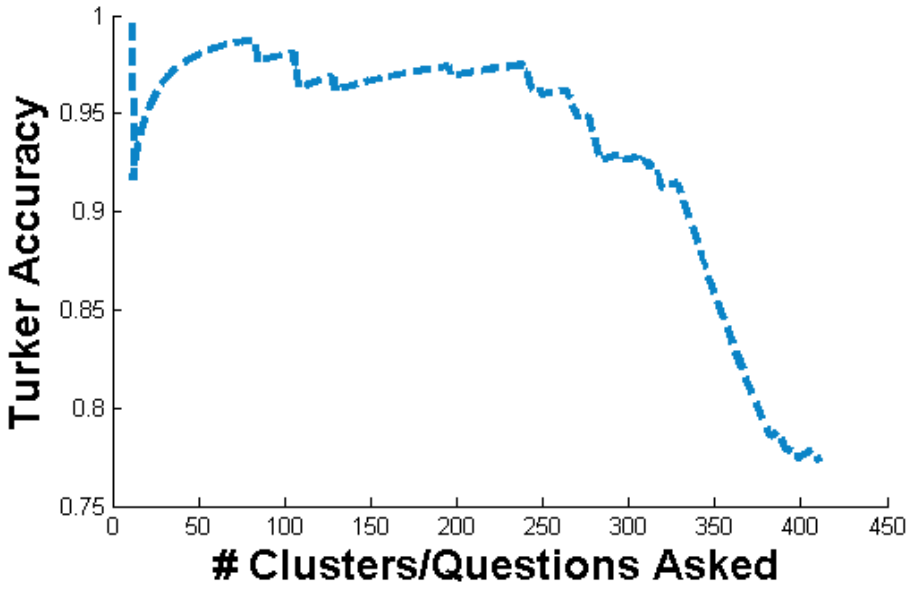


Fig. 8. Accuracy vs. recall plot for the DBLP-hard data set using probabilistic integration. If an entropy threshold is set in advance, a high accuracy can be obtained across all accepted answers.

Summary: Our experiments show a large savings in cost and gain in accuracy over previous methods. Selecting those tokens with maximum within-document mutual information and strongest clustering produce an orders-of-magnitude improvement in the number of questions needed compared to random. Integrating multiple answers probabilistically provides an increase in accuracy by reasoning over Turker uncertainty. The results in Figure 7 from real Turker experiments are consistent with those observed synthetically in Figure 6(b). Finally, probabilistic integration shows promise in providing users with a trade-off between accuracy and recall.

7. RELATED WORK

There have been a number of previous attempts at combining humans and machines into a unified data management system, but to our knowledge none that have done it probabilistically. CrowdDB [Franklin et al. 2011] utilizes humans to process queries that are either missing from the database or computationally prohibitive to calculate. The crowd is invoked only if an incoming query contains one of the incomplete values. In contrast, pi-CASTLE operates on batches over a complete, but uncertain database, improving accuracy well in advance of queries. Qurk [Marcus et al. 2011] crowdsources workflow operations of the database, such as filtering, aggregating, sorting, and joining, but makes no attempt to optimize the workload between both humans and machines. AskIt! [Boim et al. 2012] provides a similar goal of ”selecting questions” to minimize uncertainty given some budget, however their approach is purely for quality control. CrowdER [Wang et al. 2012] uses a hybrid human-machine system to perform entity resolution. The machine does an initial course pass over the data and the most likely matching pairs are verified through crowdsourcing. The primary approach of that work is HIT interface optimization as opposed to the specific question selection and answer integration of ours. DBLife [DeRose et al. 2007] invites mass collaboration to improve a database seeded by machine learning, but selection is done by humans

as needed without any means of automatically identifying likely errors. There is also no means of doing quality control or redundancy checking if humans introduce erroroneous edits.

8. CONCLUSION

In this paper we introduce pi-CASTLE, a crowd-assisted SML-based IE system that can improve the accuracy of its automated results through a crowdsourced workforce. Our mutual information-based question selection algorithm produces questions with the largest information gain with a given budget. Probabilistic answer integration can be achieved using a Bayesian formulation that combines multiple, possibly conflicting Turker answers into a single posterior distribution. We showed a large improvement in cost, speed, and accuracy over random question selection and standard majority voting combination.

While we focus on text extraction in the paper, we envision a more general Crowd-Assisted Machine Learning (CAMeL) system that uses a probabilistic database to efficiently connect and integrate crowdsourcing to improve the imperfect results from SML methods. Many of the core elements developed in pi-CASTLE such as uncertainty management, question selection, and uncertain data integration are applicable to other SML-based tasks in CAMeL.

REFERENCES

2006. Amazon. AWS Case Study: Smartsheet. (2006).

2011. *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2011, Athens, Greece, June 12-16, 2011*. ACM.

Rubi Boim, Ohad Greenshpan, Tova Milo, Slava Novgorodov, Neoklis Polyzotis, and Wang Chiew Tan. 2012. Asking the Right Questions in Crowd Data Sourcing. In *ICDE*. IEEE Computer Society, 1261–1264.

Haibin Cheng, Ruofei Zhang, Yefei Peng, Jianchang Mao, and Pang-Ning Tan. 2008. Maximum Margin Active Learning for Sequence Labeling with Different Length. In *Proceedings of the 8th industrial conference on Advances in Data Mining: Medical Applications, E-Commerce, Marketing, and Theoretical Aspects (ICDM '08)*. Springer-Verlag, 345–359. DOI:http://dx.doi.org/10.1007/978-3-540-70720-2_27

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

Pedro DeRose, Warren Shen, Fei Chen, Yoonkyong Lee, Douglas Burdick, AnHai Doan, and Raghu Ramakrishnan. 2007. DBLife: A Community Info. Mgmt Platform for the Database Research Community (Demo). In *CIDR*. 169–172.

G. D. Forney. 1973. The viterbi algorithm. *Proc. IEEE* 61, 3 (March 1973), 268–278. DOI:http://dx.doi.org/10.1109/PROC.1973.9030

Michael J. Franklin, Donald Kossmann, Tim Kraska, Sukriti Ramesh, and Reynold Xin. 2011. CrowdDB: answering queries with crowdsourcing, See DBL [2011], 61–72.

Sean Goldberg, Daisy Zhe Wang, Jeff Depree, and Tim Kraska. 2013. CASTLE: A Crowd-Assisted System for Textual Labeling & Extraction. In *Proceedings of the 2013 Conference on Human Computation (HCOMP '13)*.

Yuhong Guo and Russ Greiner. 2007. Optimistic active learning using mutual information. In *Proceedings of the 20th international joint conference on Artifical intelligence (IJCAI'07)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 823–829.

Panagiotis G. Ipeirotis, Foster Provost, and Jing Wang. 2010. Quality management on Amazon Mechanical Turk. In *Proceedings of the ACM SIGKDD Workshop on Human Computation (HCOMP '10)*. ACM, New York, NY, USA, 64–67. DOI:http://dx.doi.org/10.1145/1837885.1837906

Trausti Kristjansson, Aron Culotta, Paul Viola, and Andrew McCallum. 2004. Interactive information extraction with constrained conditional random fields. In *Proceedings of the 19th national conference on Artifical intelligence (AAAI'04)*. AAAI Press, 412–418.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *ICML*. 282–289.

Adam Marcus, Eugene Wu, David R. Karger, Samuel Madden, and Robert C. Miller. 2011. Demonstration of Qurk: a query processor for humanoperators, See DBL [2011], 1315–1318.

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

Lawrence R. Rabiner. 1989. A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of the IEEE*. 257–286.

Joel Ross, Lilly Irani, M. Six Silberman, Andrew Zaldivar, and Bill Tomlinson. 2010. Who are the crowdworkers?: shifting demographics in mechanical turk. In *CHI '10 Extended Abstracts on Human Factors in Computing Systems (CHI EA '10)*. ACM, New York, NY, USA, 2863–2872.

Burr Settles and Mark Craven. 2008. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the Conference on Empirical Methods in NLP (EMNLP '08)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 1070–1079.

Victor S. Sheng, Foster Provost, and Panagiotis G. Ipeirotis. 2008. 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)*. ACM, New York, NY, USA, 614–622.

Charles Sutton and Andrew Mccallum. 2006. *Introduction to Conditional Random Fields for Relational Learning*. MIT Press.

Daisy Zhe Wang, Michael J. Franklin, Minos N. Garofalakis, and Joseph M. Hellerstein. 2010. Querying Probabilistic Information Extraction. *PVLDB* 3, 1 (2010), 1057–1067.

Daisy Zhe Wang, Eirinaios Michelakis, Michael J. Franklin, Minos N. Garofalakis, and Joseph M. Hellerstein. 2010. Probabilistic declarative information extraction. In *ICDE*. IEEE, 173–176.

Jiannan Wang, Tim Kraska, Michael J. Franklin, and Jianhua Feng. 2012. CrowdER: Crowdsourcing Entity Resolution. *PVLDB* 5, 11 (2012), 1483–1494.

Zuobing Xu, Ram Akella, and Yi Zhang. 2007. Incorporating diversity and density in active learning for relevance feedback. In *Proceedings of the 29th European conference on IR research (ECIR'07)*. Springer-Verlag, Berlin, Heidelberg, 246–257.

Y. Zhao and Q. Ji. 2010. Non-myopic active learning with mutual information. In *2010 IEEE International Conference on Automation and Logistics (ICAL*. 511–514.

Received May 2015; revised June 2015; accepted July 2015