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

SEAN GOLDBERG and DAISY ZHE WANG, University of Florida
CHRISTAN GRANT, University of Oklahoma

The amount of text data has been growing exponentially in recent years, giving rise to automatic information extraction methods that store text annotations in a database. The current state-of-the-art structured prediction methods, however, are likely to contain errors and it is important to be able to manage the overall uncertainty of the database. On the other hand, the advent of crowdsourcing has enabled humans to aid machine algorithms at scale. In this article, we introduce pi-CASTLE, a system that optimizes and integrates human and machine computing as applied to a complex structured prediction problem involving Conditional Random Fields (CRFs). We propose strategies grounded in information theory to select a token subset, formulate questions for the crowd to label, and integrate these labelings back into the database using a method of constrained inference. On both a text segmentation task over academic citations and a named entity recognition task over tweets we show an order of magnitude improvement in accuracy gain over baseline methods.

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

Additional Key Words and Phrases: Crowdsourcing, information extraction, probabilistic database systems, probabilistic models

ACM Reference Format:
Sean Goldberg, Daisy Zhe Wang, and Christan Grant. 2016. A probabilistically integrated system for crowd-assisted text labeling and extraction. J. Data and Information Quality 8, 2, Article 10 (February 2017), 23 pages.
DOI: http://dx.doi.org/10.1145/3012003

1. INTRODUCTION

In recent years, there has been an explosion of unstructured text data from social networks like Twitter and Facebook, within enterprises via emails and digitized documents, and across the Web. 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 content annotations from bibliographic citations and entity relationships from news articles.

Automatic information extraction can be viewed as a *structured classification* problem using statistical machine learning techniques. Given an input sentence x, the

This work was partially supported by DARPA under FA8750-12-2-0348-2 (DEFT/CUBISM), a Campus Executive Fellowship from Sandia National Laboratories, and a National Science Foundation Graduate Research Fellowship under Grant No. DGE-0802270.
Authors' addresses: S. Goldberg and D. Z. Wang, Computer & Information Science & Engineering Department, University of Florida, E457 CSE Building, PO Box 116120, Gainesville, Florida 32611-6120; emails: sean@cise.ufl.edu, daisyw@cise.ufl.edu; C. Grant, School of Computer Science, University of Oklahoma, 110 W. Boyd, DEH 234, Norman, Oklahoma 73019; email: cgrant@ou.edu.

DOI: http://dx.doi.org/10.1145/3012003

output label y has a rich internal structure. An example is a probabilistic sequence of annotations for each word in the sentence. This approach of *sequence learning* has been the focus of much research into automatic IE for tasks such as Text Segmentation (TS) or Named Entity Recognition (NER). The most common and state-of-the-art sequence model for these tasks is the linear-chain Conditional Random Field (CRF) [Lafferty et al. 2001].

Because of the inherent uncertainty and fallibility of many machine learning algorithms, recent work has turned to the incorporation of a human element for correcting errors or validating output of machine results. Crowdsourcing platforms like Amazon Mechanical Turk (AMT) have made it possible to utilize human computation efficiently and cheaply. Nevertheless, human annotations are still much more expensive and time-consuming compared to algorithmic labeling [Mozafari et al. 2014] and care must be taken to optimize the work administered to the crowd.

Previous work in utilizing a hybrid of traditional and crowd computation for structured classification includes entity resolution [Mozafari et al. 2014], web table [Fan et al. 2014] and ontology alignment [Sarasua et al. 2012], and probabilistic query processing [Ciceri et al. 2016]. The main research challenges are optimizing *data selection* and *question construction*. Data selection involves targeting the most significant human contributions given a fixed budget of questions. Question construction is concerned with extracting the maximum possible information out of each question and is intimately connected to the data selection problem.

There has been little work in combining human and machine computation for text classification due to the complexity of the structured prediction models. Data selection and question construction are more difficult because they have to reason with the internal structure of the probabilistic graphical models.

In this article, we build on existing work to develop an end-to-end system that not only tackles both of these selection problems, but fully integrates the crowdsourced response back into the machine model. Our system, pi-CASTLE, is a crowd-assisted Statistical Machine Learning (SML)-based IE system that uses a probabilistic database to execute, optimize, and integrate human and machine computation for improving text extraction and processing. pi-CASTLE initially employs a linear-chain 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 through the deployment of Amazon Mechanical Turk Human Intelligence Tasks (HITs) to correct the most uncertain and influential tokens and integrate their responses back into the data model.

By allowing trained algorithms 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 manage uncertainty associated with classification results and do data cleaning from within the database, a probabilistic data model and system is needed. We use the model described in Wang et al. [2010b], storing both uncertain relations and probabilistic models as first class objects. We also implement 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 evaluating tokens in terms of their information value to the rest of the database and generating questions based on the highest scoring tokens to be pushed to AMT. Information value of each token is determined by a set of *information functions* that optimize different metrics over the database. pi-CASTLE uses concepts from information theory to select either the most uncertain tokens or the ones likely to have the most influence on other tokens. 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. Optimal selection is NP^{PP}-hard in general [Krause and Guestrin 2009] and we develop a set of approximate scoring functions. We choose to perform data cleaning at the token level instead of the sentence or document level in order to exploit a phenomenon known as *correction propagation* [Culotta et al. 2006], wherein influence can spread throughout the graphical model from a single observation. This allows pi-CASTLE to maximize the efficiency and value of the data cleaning process.

pi-CASTLE is also able to construct questions in such a way that they maximize the impact on selected tokens. By exploiting redundancies in token usage on a global scale across documents, pi-CASTLE is able to map many tokens to a single question to achieve the greatest "bang for our buck." As an example, if the crowd is able to correctly resolve the token *Obama* as a reference to a PERSON, then all entries containing *Obama* in a particular context can be updated to reflect this new information. This directly improves on many existing systems [Kondreddi et al. 2014] that correct at best a single entry at a time.

The final design challenge is how to adequately handle evidence that has been collected from the crowd. Because every question posed costs financial and temporal resources, a central theme of pi-CASTLE is getting the most impact out of every question. Because in a relational learning task the output of one example may influence that of "nearby" examples, we develop a *constrained inference* framework that can use answers provided from the crowd to improve the results of other entries in the database. Understanding *Obama* is a person may improve the machine's decision making on other related tokens such as *Barack*.

One key application of this work is in knowledge transfer of models from one domain to another. Many pretrained off-the-shelf models give poor results when applied to a new domain or data that follow a different distribution and are expensive or impossible to retrain. We demonstrate in our experiments pi-CASTLE's ability to optimize the cost of this knowledge transfer process through appropriate balancing of the human and machine components.

The following are the key technical contributions of this article:

—We design and implement a crowd-assisted IE system, pi-CASTLE, based on a CRF that uses a probabilistic database as a foundation to execute, optimize, and tightly integrate machine computation over CRF models and human computation over crowdsourcing services.

—We develop novel approximate information theoretic techniques that can automatically generate AMT questions to maximally reduce uncertainty and errors for IE transfer tasks compared to a set of baselines. Using data from multiple citation datasets for text segmentation and newswire and Twitter data for named entity recognition, we show that these techniques lead to orders-of-magnitude fewer questions, reducing cost in improving the overall accuracy of extractions.

—We demonstrate the performance benefit of performing constrained inference and selecting questions in such a way as to maximize this benefit. Probabilistic integration of crowd answers can achieve up to a 33% increase in F1 compared to not running constrained inference.

The article is summarized as follows. Section 2 compares and contrasts similar systems to pi-CASTLE. We give an overview of the pi-CASTLE system in Section 3. Section 4 details the CRF model at the heart of the system and how it performs learning and inference. Sections 5 and 6 cover our techniques for performing question selection and integration, respectively. Our experiments can be found in Section 7, while Section 8 contains our Conclusions.

2. RELATED WORK

Crowdsourcing has been employed in many different systems as a tool for improving database construction and data processing. Humans can be used for common query operations such as sorts and joins [Marcus et al. 2011] or ranking and grouping data [Davidson et al. 2013]. A crowd can also be used to fill in queried fields that may be missing [Franklin et al. 2011; Kondreddi et al. 2014]. These systems are deterministic and act over the entirety of missing data without regard to importance or budgetary concerns.

pi-CASTLE, on the other hand, is primarily concerned with managing an existing database containing uncertain data and prioritizing the most informative corrections. There are a number of systems that attempt a similar problem in completely different domains. Ciceri et al. [2016] optimizes a Top-k ordering of videos and images by focusing users to prune a tree of possible orderings. Fan et al. [2014] probabilistically extracts web tables and tasks users with aligning both the “most difficult” and “most influential” columns in the table. Zhang et al. [2015] is also concerned with crowdsourcing uncertain database fields, but under a simpler set of assumptions that do not incorporate the relational nature of the underlying probabilistic model.

Crowdsourcing has also been applied in similar domains of citation extraction and named entity recognition. Culotta et al. [2006] applies mutual information to selecting tokens from citations, but does not discuss it in a system perspective that scales to the crowd or apply any batching of human edits to multiple citations. They also use a different approximation of mutual information. The relationship among task features, crowd preferences, and crowdsourcing performance for hybrid NER systems particularly as it applies to tweets is explored in Feyisetan et al. [2015]. pi-CASTLE differs by focusing on the role machines play in the probabilistic selection and integration process. Improving the crowdsourcing specifics are outside the scope of this article. A human-machine hybrid NER system is discussed in Braunschweig et al. [2013], but their goal is to decompose individual examples into either automatic-only or crowd-only and use a rule-based automatic extraction method. By contrast, pi-CASTLE employs statistical machine learning extraction techniques with information-theoretic selection mechanisms and example-batching among questions to create an optimal human-machine hybrid extraction system.

3. SYSTEM OVERVIEW

Figure 1 outlines the basic architecture of the pi-CASTLE system, which can be conceptualized into four main components: (1) CRF Extraction and Inference, (2) Question Selection, (3) HIT Management, and (4) Human/Machine Integration. The arrows chart the flow of data within and between different components. Overall, data flows through the four components in order. First, the CRF model performs automatic text extraction and labeling. Both the extractions themselves and their associated uncertainties are stored in a probabilistic database. Next, a set of questions is generated as some function of the data uncertainty and given budget. The HIT manager formulates and pushes these questions to the crowd and retrieves the answers. Finally, the Turker answers are integrated back into the database as a set of constraints on the inference that improve the initial results.

In this section, we briefly outline each of the system’s main components and how they are related, as well as how data is specifically stored in the data model through the use of the running example in Figure 2. While we use existing techniques for (1) CRF

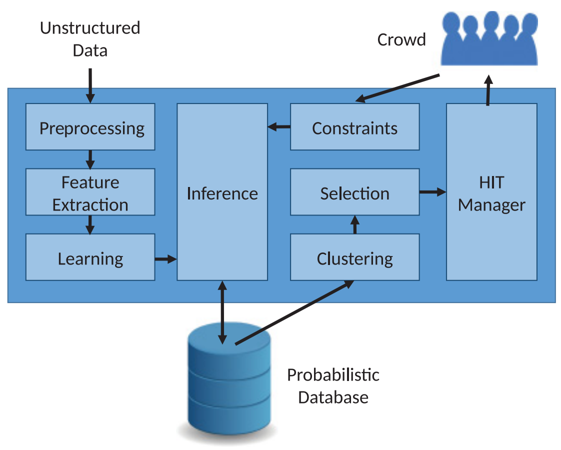
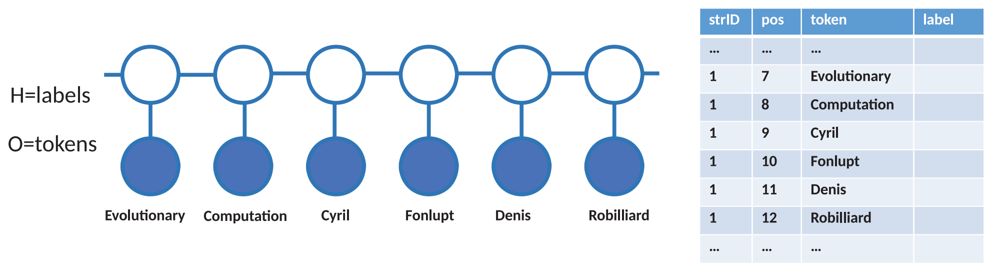


Fig. 1. Architecture of the pi-CASTLE system.



strID	pos	token	label
...	...	...	
1	7	Evolutionary	
1	8	Computation	
1	9	Cyril	
1	10	Fonlupt	
1	11	Denis	
1	12	Robilliard	
...	...	...	

Fig. 2. An example CRF model and the associated storage format in the database. Observed nodes correspond to token values. Hidden nodes are probabilistic labels that contain a distribution over the label space.

Extraction/Inference and (3) HIT Management, we develop novel techniques for (2) Question Selection and (4) Human/Machine Integration in Sections 5 and 6.

3.1. CRF Extraction and Inference

The initial machine approach to the structured prediction problem is handled by the components associated with the CRF, including preprocessing, feature extraction from the text, and model learning from a training set. The CRF model infers the hidden labels associated with each text token and stores the results in the database.

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). For efficient and persistent data storage and retrieval, we store the tokens in a probabilistic database, adopting and expanding upon the data model outlined in Wang et al. [2010a]. Each unique occurrence of a token, identified by a text-string ID (strID) and position (pos), is stored as a record in the relational table TokenTbl. A TokenTbl has the following schema:

$$\text{TokenTbl}(\text{strID}, \text{pos}, \text{token}, \text{label}^p).$$

An example is shown in Figure 2. The final attribute label^p is probabilistic and comes from the CRF inference process. It represents a distribution over the possible labels and their associated marginal probabilities. These probabilities will be updated after incorporating crowdsourced data acquired from Amazon Mechanical Turk. The CRF model is described in more detail in Section 4.

3.2. Question Selection

The main contribution of pi-CASTLE is the ability to take an uncertain database as constructed using the CRF components and automatically improve it by pinpointing likely errors. In order to work within the constraints of a fixed budget, we need to select the most *information-dense* fields to correct. As we describe in more detail later in the article, the relational structure of our model enables token inferences to affect other tokens through their dependencies. Additionally, highly redundant tokens have common contextual patterns across documents that allow fields to be mapped using the same human inferences.

Using these two notions, we develop a way to score each token in terms of its information density and select those most likely to have the strongest improvement on the database. Figure 4 shows a table view that includes the maximum likelihood label for each token and a score based on some scoring function. The goal of the Selection module is to evaluate tokens in such a way that their correction has a maximum influence on the database after integration.

Section 5 examines two different optimization functions on the database and the information functions that derive from them. One optimizes the reduction in uncertainty and suggests picking those tokens with the highest marginal entropy. The other maximizes the influence between the crowdsourced fields and the remaining fields and results in a mutual information calculation between tokens and their neighbors. In either case, we incorporate clustering into the scoring function so the most common errors have a higher weight in their selection and individual questions can be applied to many tokens simultaneously.

3.3. HIT Management

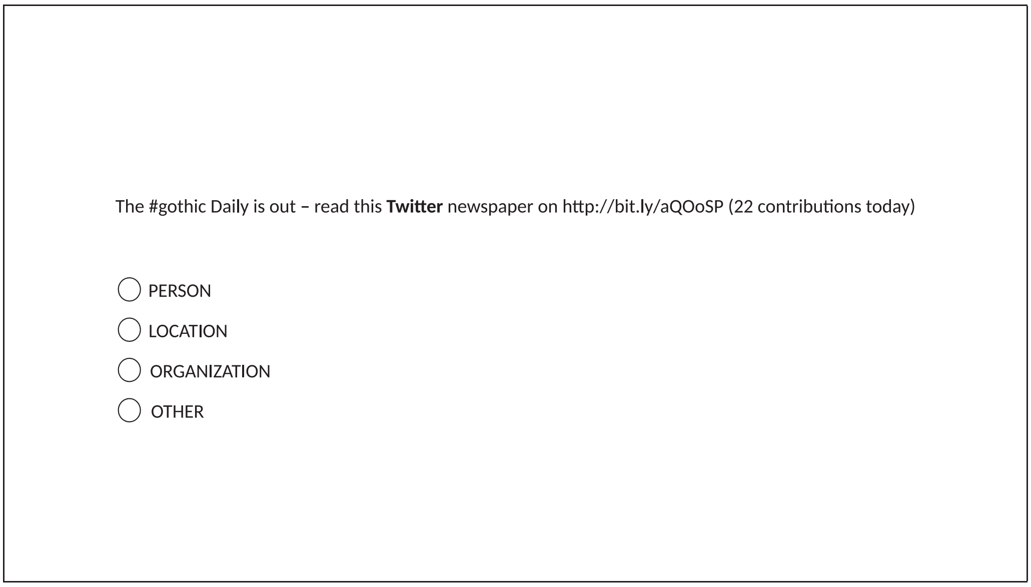
The HIT Management component has the responsibility of taking selected tokens, converting those tokens into questions in the form of HITs, and posting them onto Amazon Mechanical Turk. Part of the focus of pi-CASTLE is on reducing the problem of annotating an entire text string to annotating only specific tokens at a time. The simplicity of this task avoids unneeded redundancy and translates into a simple question interface less prone to human error.

An example interface is shown in Figure 3. The entire text document (in this case a citation) is shown with the query token bolded. Users select from the set of all labels the one they believe belongs to the bolded token. The brevity of each question allows bundling of multiple token annotations into a single HIT. For the time and cost of labeling a single unstructured text document from scratch, pi-CASTLE is able to acquire labels to the same number of super information-dense tokens which will have a much larger impact on improving the quality of the database.

Specific details of the AMT marketplace such as the price, length of posting, and number of Turkers assigned to each HIT are outside the focus of this article. In practice, they would be set according to the constraints of the user.

3.4. Human/Machine Integration

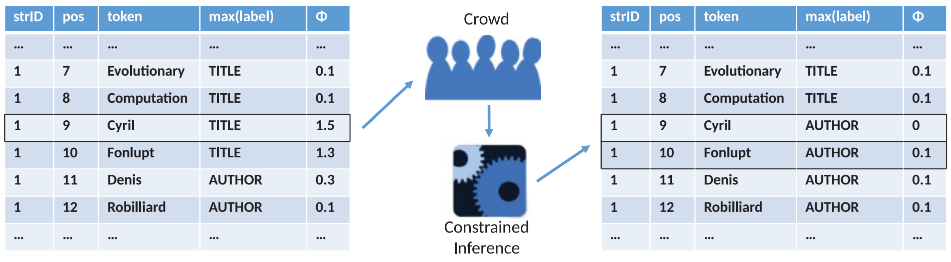
The final component takes the human response to selected questions, aggregates their results, and integrates them into the final database. pi-CASTLE is agnostic to the method by which crowd results may be combined and any method that derives a single answer from a pool of possibilities may be employed. We found majority voting worked well enough to be useful due to high worker ability on text annotation tasks. For more



The #gothic Daily is out – read this **Twitter** newspaper on http://bit.ly/aQOoSP (22 contributions today)

- ◯ PERSON
- ◯ LOCATION
- ◯ ORGANIZATION
- ◯ OTHER

Fig. 3. Sample Mechanical Turk HIT Interface.



strID	pos	token	max(label)	Φ
...	...	...	...	...
1	7	Evolutionary	TITLE	0.1
1	8	Computation	TITLE	0.1
1	9	Cyril	TITLE	1.5
1	10	Fonlupt	TITLE	1.3
1	11	Denis	AUTHOR	0.3
1	12	Robilliard	AUTHOR	0.1
...	...	...	...	...

strID	pos	token	max(label)	Φ
...	...	...	...	...
1	7	Evolutionary	TITLE	0.1
1	8	Computation	TITLE	0.1
1	9	Cyril	AUTHOR	0
1	10	Fonlupt	AUTHOR	0.1
1	11	Denis	AUTHOR	0.1
1	12	Robilliard	AUTHOR	0.1
...	...	...	...	...

Fig. 4. Database view of the process of data selection and integration. An information function ϕ maps each token to a selection score. Here, because "Cyril" is an uncommon name, the machine confuses it as a `Title`. It has the highest scoring information function so it is sent to the crowd for labeling where it is confirmed to be an `Author`. Constrained inference propagates this information, changing "Fonlupt" to an `Author` as well.

difficult tasks with variable worker ability, pi-CASTLE is modular enough to utilize any number of quality control mechanisms as found in Sheshadri and Lease [2013].

The key insight that distinguishes pi-CASTLE from other crowdsourced data cleaning systems is the treatment of new evidence as observed variables in the conditional random field inference process. CRF Inference finds a global best-fit path through the label space for each document. Fixing certain fields to the crowdsourced label constrains the total label space and has a direct influence on tokens in the neighborhood of the observed one. The Constraints module stores the crowdsourced evidence and is used in subsequent inference passes over the data. The specific details of constrained CRF inference are detailed in Section 6.

Figure 4 shows a scenario in which identifying and correcting one node from a `Title` to an `Author` allows the machine to infer that the following token is also likely to be an `Author`. pi-CASTLE chooses selections wisely so as to maximize this degree of influence.

4. MODEL

The particular model that pi-CASTLE uses for learning and inference is the linear-chain conditional random field, an undirected discriminative graphical model commonly used in natural language and speech processing tasks. Here we utilize a CRF to perform text annotation and discuss how the internal structure of the model is modified to integrate human responses coming from the crowd.

4.1. Conditional Random Fields

Conditional random fields (CRFs) [Sutton and Mccallum 2004] are a popular framework in natural language processing that use undirected graphical models for structured classification. Given a set of observed input random variables $\mathbf{O}$, a CRF encodes a probability distribution over a collection of latent random variables $\mathbf{H}$ using features of the input space. The benefit of CRFs compared to traditional classifiers is the ability to utilize multiple overlapping features of the input space. The most common structure is to link the input variables into a single chain. The first-order Markov assumption makes it so each output variable is dependent only on the current observation and the previous label, akin to a Finite State Machine (FSM).

Let $\mathbf{O} = \langle O_1, O_2, \ldots, O_T \rangle$ be a set of input variables, such as word tokens appearing in sequence in a document. There is a set of corresponding output variables $\mathbf{H} = \langle H_1, H_2, \ldots, H_T \rangle$ that represent a sequence of states for each input. For a text segmentation task, each H_t will denote the class of the word (such as `Title` or `Author`). The full CRF output defines a joint probability distribution over all possible state sequences as

$$p(\mathbf{H}|\mathbf{O}) = \frac{1}{Z}\exp\left\{\sum_{t=1}^{T}\sum_{j=1}^{J}\lambda_j f_j(H_i, H_{i-1}, O_i)\right\}, \tag{1}$$

where each $f_j(H_t, H_{t-1}, O_t)$ is a feature function over its arguments, λ_j is a weight for each feature, and Z is a normalization factor over all states. The features encode the Markov assumption in that each feature concerning H_t is only dependent on the prior state H_{t-1}, given the observation. Common features include syntactic and lexical properties as well as colocation (e.g., two `Title` states are more likely to appear next to each other).

Figure 2 shows an example CRF model over a subset of a citation string. The shaded nodes are the input variables and unshaded nodes the output states. The possible states for each token are H^i = {`Title`, `Author`, `Conference`, `ISBN`, `Publisher`, `Series`, `Proceedings`, `Year`}.

Maximum a posteriori training of the model parameters is usually done through hill-climbing methods such as gradient ascent or limited-memory BFGS [Liu and Nocedal 1989] using a labeled training sample. Inference can refer to either calculating the max probability assignment $\mathbf{H}* = \text{argmax}_{\mathbf{H}}\, p(\mathbf{H}|\mathbf{O})$ using the Viterbi algorithm [Forney 1973] or the marginal probabilities $p(H_i|\mathbf{O})$ of each output node using the Forward-Backward algorithm [Rabiner and Juang 1986]. The computational complexity of the Forward-Backward algorithm used for training is $O(TL^2NG)$, where T is the length of the document, L is the number of labels, N is the number of training documents, and G is the number of gradient computations. For inference, both Viterbi and Forward-Backward have a complexity of $O(TL^2)$ for each document.

Understanding the Viterbi algorithm is crucial to understanding how pi-CASTLE incorporates crowdsourced evidence back into the system. Viterbi uses dynamic programming to avoid searching over the exponentially numerous global sequences $\mathbf{H}$. Let H_t^i be the i^{th} label of token t and $\delta_t(H_t^i)$ be the probability after having observed nodes

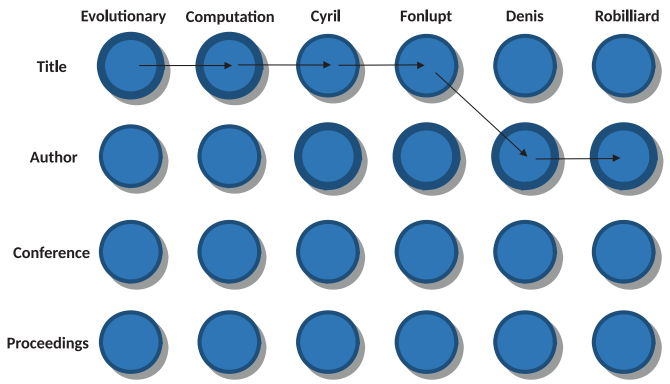


Fig. 5. Example matrix of labels and tokens and a maximum likelihood path through the label space. The machine initially gets the citation wrong and incorrectly identifies the tokens "Cyril" and "Fonlupt" to be part of the `Title` when they should be labeled `Author`. Dark outer circles represent the true token labels.

$O_1, \ldots, O_t$ that H_t has label i. The Viterbi algorithm operates by the induction step:

$$\delta_{t+1}(H_t^i) = \max_{H_t^{i'}} \left[\delta_t(H_t^{i'}) \exp\left(\sum_{j=1}^{J} \lambda_j f_j(H_{i'}, H_i, O_i) \right) \right]. \tag{2}$$

The induction step fills out a dynamic programming table that can be backtracked to recover the maximum likelihood state $\mathbf{H}*$. The algorithm can be understood as traversing through a $T \times L$ matrix, where L is the number of labels. This is visualized in a truncated form in Figure 5. Here, the arrows represent the maximum likelihood "path" through the grid. The next section gives insight into how crowdsourcing can be used to modify this path.

4.2. Constrained Conditional Random Fields

Constrained Conditional Random Fields (CCRFs) have been previously studied in the context of *interactive information extraction* [Kristjansson et al. 2004; Culotta et al. 2006] where the goal is to solicit feedback from a human to improve a CRF-based information extraction model. After incorporating the newly observed evidence back into the model, the inference is performed a second time using a constrained version of Viterbi.

Constrained Viterbi modifies Equation (2) such that $\mathbf{H}*$ is constrained to pass through a particular node H_t^l in the $\mathbf{H}$ grid, given some label l supplied for the token t. This is implemented by setting all $\delta_t(H_t^i)$ to 0 if $i \neq l$. Because Viterbi defines the optimal path recursively and the CRF features are dependent on adjacent labels, the constraint propagates to other nodes in the grid and results in different predictions for those nodes. Culotta et al. [2006] calls this phenomenon *correction propagation*.

5. QUESTION SELECTION

In this section, we formalize the question of optimally posing questions to the crowd by analyzing two key selection strategies in the context of which global optimization functions they minimize. This leads us to strategies for either maximizing the uncertainty reduction of the system or the influence of the selected nodes. To maximize the information density of each question, we describe a contextual clustering technique that folds multiple uncertain examples into a single question.

5.1. Question Selection Problem

A general probabilistic graph can be described simply as tuple $G = (\mathbf{V}, \mathbf{E}, P)$ that consists of *nodes* $\mathbf{V}$, *edges* $\mathbf{E}$, and a *probability distribution* P over those nodes and edges. In general, P may denote either a directed Bayesian network or an undirected Markov random field. Assume the $\mathbf{V}$ nodes are partitioned into a set of observed nodes $\mathbf{O}$ and hidden nodes $\mathbf{H}$ such that $\mathbf{O} \cup \mathbf{H} = \mathbf{V}$. By virtue of having learned a model, there exists some P that can be used to make predictions on the distribution of values of $\mathbf{H}$. Assume also that we are given a budget of K observations to make on the hidden nodes, after which we have a new partition between the observed hidden nodes $\mathbf{Y}$, where $|\mathbf{Y}| = K$, and the still unobserved hidden nodes $\mathbf{X}$, such that $\mathbf{X} \cup \mathbf{Y} = \mathbf{H}$. While this article focuses on a specific probabilistic graph known as a CRF, the problem we are posing is a more general one applied to any probabilistic graph.

Question Selection Problem: *Given a budget of $K = |\mathbf{Y}|$ questions, how do we pose questions to the observer in such a way that the resulting observations $\mathbf{Y}$ minimize the prediction error on the remaining hidden nodes $\mathbf{X}$?*

The mapping of questions to observable nodes need not be one-to-one and the problem decomposes into using questions to trade off both the largest mapping from a question to many nodes and the individual information density of those nodes. The former is related to the content and redundancy of connected observations and the latter deeply related to the structure of the graph.

The Question Selection Problem 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. Traditionally these examples are Independently and Identically Distributed (IID) and take no account of the causal influence of one example's label on another. While active learning has been applied to the structure prediction domain [Settles and Craven 2008; Cheng et al. 2008], the examples primarily consist of individual independent graphs and in each case the entire graph is labeled. This reduces the problem to labeling a set of IID examples. The Question Selection Problem is concerned with labeling individual hidden nodes in a larger graph structure. Given that most examples contain sparse labeling errors, we hope to achieve maximum efficiency in terms of financial and temporal cost while also being more amenable to AMT's microtask framework.

Our approach to solving the Question Selection Problem is to phrase it as an optimization function $g(\mathbf{X}, \mathbf{Y}, \mathbf{O})$ over the graph. From the collection of hidden nodes $\mathbf{H}$, we seek to find the set of new observations $\mathbf{Y}$ that maximizes $g(\mathbf{X}, \mathbf{Y}, \mathbf{O})$.

$$\mathbf{Y}* = \underset{\mathbf{Y}}{\operatorname{argmax}}\, g(\mathbf{X}, \mathbf{Y}, \mathbf{O}). \tag{3}$$

We discuss two possible optimization functions in the succeeding sections and approximate solutions to solving them. A key requirement in scaling to large graphs is the ability to evaluate and rank nodes independently by some information function ϕ such that the optimal $\mathbf{Y}*$ are just the Top-k scoring nodes $\phi(Y_k)$. These information functions have been previously published as heuristics in Goldberg et al. [2013] and in this article we put them on a more sound theoretical footing.

Our primary purpose is to establish and make progress on the Question Selection Problem as it pertains to text-based linear-chain CRFs, which have wide applicability in natural language processing applications.

5.2. Uncertainty Reduction

Most classifiers give as output both a class prediction and confidence score. The confidence score measures how difficult the machine finds the problem to be and subsequently how confident it is in its output. It is reasonable to assume then that for properly trained classifiers there is some correlation between prediction accuracy and confidence. For structured prediction problems, the confidence is typically renormalized as a probability distribution over all possible joint distributions of the hidden nodes. One strategy for improving prediction performance is to try to reduce the uncertainty associated with the output distribution.

Uncertainty can be modeled using entropy. The entropy of a distribution of possible outputs T is defined as

$$\mathcal{H}(\mathbf{T}) = -\sum_{t \in \mathbf{T}} P(t) log P(t). \tag{4}$$

When the distribution is peaked toward some particular output value t we say the system has low uncertainty and is confident in that t is the correct output. If the distribution is spread over a wider range of possible values, the system is characterized as having high uncertainty.

After the initial machine prediction phase, the system of hidden nodes $\mathbf{H} = \mathbf{X} \cup \mathbf{Y}$ has some associated uncertainty $\mathcal{H}(\mathbf{X}, \mathbf{Y}|\mathbf{O})$. After selecting $\mathbf{Y}$ nodes for observation, the remaining uncertainty is $\mathcal{H}(\mathbf{X}|\mathbf{Y}, \mathbf{O})$. One way of minimizing final prediction error is to observe those nodes that give the largest reduction between $\mathcal{H}(\mathbf{X}, \mathbf{Y}|\mathbf{O})$ and $\mathcal{H}(\mathbf{X}|\mathbf{Y}, \mathbf{O})$. Thus, the optimization function is

$$g(\mathbf{X}, \mathbf{Y}, \mathbf{O}) = \mathcal{H}(\mathbf{X}, \mathbf{Y}|\mathbf{O}) - \mathcal{H}(\mathbf{X}|\mathbf{Y}, \mathbf{O}). \tag{5}$$

Using the identity

$$\mathcal{H}(\mathbf{X}, \mathbf{Y}|\mathbf{O}) = \mathcal{H}(\mathbf{Y}|\mathbf{O}) + \mathcal{H}(\mathbf{X}|\mathbf{Y}, \mathbf{O}), \tag{6}$$

this uncertainty reduction is equivalent to maximizing the marginal entropy of the selected nodes $\mathcal{H}(\mathbf{Y}|\mathbf{O})$. Solving this problem exactly requires calculating the joint distribution of all possible subsets of the budget size K. As a simplifying assumption, we relax the notion of connectivity between nodes.

For a set of independent random variables $Y_1, \ldots, Y_K$, the entropy can be written as the sum of the individual marginal entropies

$$\mathcal{H}(Y_1, \ldots, Y_K|\mathbf{O}) = \mathcal{H}(Y_1|\mathbf{O}) + \cdots + \mathcal{H}(Y_K|\mathbf{O}). \tag{7}$$

Under this independence assumption, maximization of $\mathcal{H}(\mathbf{Y}|\mathbf{O})$ is equivalent to selecting the individual $\mathcal{H}(Y_k|\mathbf{O})$ that have the largest individual marginal entropies given the observed nodes. This results in an information function

$$\phi_{ENT}(Y_k) = \mathcal{H}(Y_k|\mathbf{O}). \tag{8}$$

We refer to this information function as the *token entropy*.

Token entropy has appeared elsewhere in the literature where it is referred to as *uncertainty sampling* [Lewis and Gale 1994]. Token entropy is ideal for rooting out specific individual tokens that the machine has difficulty classifying. The fundamental shortcoming is that for a structured prediction problem, it is unable to take the data's connectivity into account. Even without relaxing the dependency assumptions, maximization of $\mathcal{H}(\mathbf{Y}|\mathbf{O})$ in no way takes into account how the newly observed nodes $\mathbf{Y}$ are related to the still-unobserved nodes $\mathbf{X}$. In the next section, we introduce a novel way of incorporating token dependencies into the Question Selection Problem using the concept of mutual information.

5.3. Influence Maximization

Due to the dependence properties of certain nodes in the graph on other nodes, an ideal selection strategy would take into account the influence an observation has on its surrounding neighborhood. *Mutual information* (MI) is a pairwise metric between random variables that quantifies how much the uncertainty of one is reduced when the other is observed.

Specifically, for two sets of random variables $\mathbf{X}$ and $\mathbf{Y}$, we can define the mutual information between them in terms of their entropies as[1]

$$\mathcal{I}(\mathbf{X};\mathbf{Y}|\mathbf{O}) = \mathcal{H}(\mathbf{X}|\mathbf{O}) + \mathcal{H}(\mathbf{Y}|\mathbf{O}) - \mathcal{H}(\mathbf{X},\mathbf{Y}|\mathbf{O}). \tag{9}$$

It represents the difference between the joint entropy $\mathcal{H}(\mathbf{X},\mathbf{Y}|\mathbf{O})$ and the individual entropies $\mathcal{H}(\mathbf{X}|\mathbf{O})$ and $\mathcal{H}(\mathbf{Y}|\mathbf{O})$. Random variables that are highly uncorrelated will have a joint entropy equivalent to the sum of their entropies and thus zero information. On the other hand, highly correlated variables will have large degrees of mutual information.

In terms of the Question Selection Problem, we would like to select those variables $\mathbf{Y}$ which, once observed, have the largest "effect" on the remaining variables $\mathbf{X}$. The impact of observation on surrounding random variables was discussed in Section 4 in reference to the Viterbi algorithm since each label depends on the labels of its neighbors. Thus, we are concerned with optimizing the difference in uncertainty between the variables $\mathbf{X}$ initially and those same variables conditioned on the selected variables $\mathbf{Y}$, given the original observed variables $\mathbf{O}$. The optimization function for this strategy becomes

$$g(\mathbf{X},\mathbf{Y},\mathbf{O}) = \mathcal{H}(\mathbf{X}|\mathbf{O}) - \mathcal{H}(\mathbf{X}|\mathbf{Y},\mathbf{O}). \tag{10}$$

Using the identity for conditional entropy

$$\mathcal{H}(\mathbf{X}|\mathbf{Y},\mathbf{O}) = \mathcal{H}(\mathbf{X},\mathbf{Y}|\mathbf{O}) - \mathcal{H}(\mathbf{Y}|\mathbf{O}) \tag{11}$$

as well as the definition of MI in Equation (9), it is clear that this is precisely equivalent to optimizing the mutual information between the newly observed variables $\mathbf{Y}$ and remaining variables $\mathbf{X}$.

This problem is in some sense "harder" than the uncertainty reduction problem of the previous section. As before, we have to calculate all possible partitions of variables into $\mathbf{X}$ and $\mathbf{Y}$ and calculate marginal entropies. Here, we also have mutual entropies to calculate and the problem cannot be reduced by relaxing dependency properties. In fact, if we do assume independence we lose the entire ability to reason using mutual information.

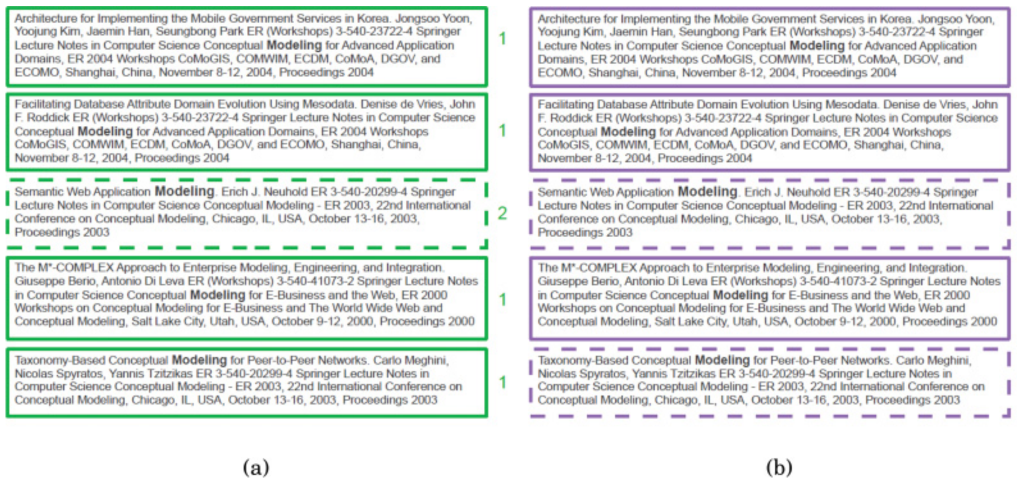
Instead, we rely on a different approximation strategy, exploiting the structural properties of the graph. Given that all hidden nodes are composed as a linear chain, the bulk of influence a node has is purely to its two neighbors. This is due to the Data Processing Inequality [Kinney and Atwal 2014], which says states that information along a Markov chain can only decrease. As a first-order approximation we consider only this influence, for each node calculating the mutual information between it and its neighbors:

$$g(\mathbf{X},\mathbf{Y},\mathbf{O}) = \sum_{Y_k\in\mathbf{Y}} \mathcal{I}(Y_k;\text{neighbors}(Y_k)|\mathbf{O}) \tag{12}$$

$$= \sum_{Y_k\in Y} \mathcal{I}(Y_k,\text{left}(Y_k)|\mathbf{O}) + \mathcal{I}(Y_k,\text{right}(Y_k|\mathbf{O}), \tag{13}$$

where left(Y_k) refers to the neighbor preceding Y_k in the chain and right(Y_k) refers to the neighbor succeeding it. This results in a simple information function defined by

[1]Note that although mutual information is not strictly designed to function over sets of random variables, treating the joint probability of a set as a single random variable allows the previous definition to hold. We continue to use the set notation for clarity with previous notation.


(a)

Architecture for Implementing the Mobile Government Services in Korea. Jongsoo Yoon, Yoojung Kim, Jaemin Han, Seungbong Park ER (Workshops) 3-540-23722-4 Springer Lecture Notes in Computer Science Conceptual **Modeling** for Advanced Application Domains, ER 2004 Workshops CoMoGIS, COMWIM, ECDM, CoMoA, DGOV, and ECOMO, Shanghai, China, November 8-12, 2004, Proceedings 2004 — 1

Facilitating Database Attribute Domain Evolution Using Mesodata. Denise de Vries, John F. Roddick ER (Workshops) 3-540-23722-4 Springer Lecture Notes in Computer Science Conceptual **Modeling** for Advanced Application Domains, ER 2004 Workshops CoMoGIS, COMWIM, ECDM, CoMoA, DGOV, and ECOMO, Shanghai, China, November 8-12, 2004, Proceedings 2004 — 1

Semantic Web Application **Modeling**. Erich J. Neuhold ER 3-540-20299-4 Springer Lecture Notes in Computer Science Conceptual Modeling - ER 2003, 22nd International Conference on Conceptual Modeling, Chicago, IL, USA, October 13-16, 2003, Proceedings 2003 — 2

The M*-COMPLEX Approach to Enterprise Modeling, Engineering, and Integration. Giuseppe Berio, Antonio Di Leva ER (Workshops) 3-540-41073-2 Springer Lecture Notes in Computer Science Conceptual **Modeling** for E-Business and the Web, ER 2000 Workshops on Conceptual Modeling for E-Business and The World Wide Web and Conceptual Modeling, Salt Lake City, Utah, USA, October 9-12, 2000, Proceedings 2000 — 1

Taxonomy-Based Conceptual **Modeling** for Peer-to-Peer Networks. Carlo Meghini, Nicolas Spyratos, Yannis Tzitzikas ER 3-540-20299-4 Springer Lecture Notes in Computer Science Conceptual Modeling - ER 2003, 22nd International Conference on Conceptual Modeling, Chicago, IL, USA, October 13-16, 2003, Proceedings 2003 — 1

(b)

Architecture for Implementing the Mobile Government Services in Korea. Jongsoo Yoon, Yoojung Kim, Jaemin Han, Seungbong Park ER (Workshops) 3-540-23722-4 Springer Lecture Notes in Computer Science Conceptual **Modeling** for Advanced Application Domains, ER 2004 Workshops CoMoGIS, COMWIM, ECDM, CoMoA, DGOV, and ECOMO, Shanghai, China, November 8-12, 2004, Proceedings 2004 — 1

Facilitating Database Attribute Domain Evolution Using Mesodata. Denise de Vries, John F. Roddick ER (Workshops) 3-540-23722-4 Springer Lecture Notes in Computer Science Conceptual **Modeling** for Advanced Application Domains, ER 2004 Workshops CoMoGIS, COMWIM, ECDM, CoMoA, DGOV, and ECOMO, Shanghai, China, November 8-12, 2004, Proceedings 2004 — 1

Semantic Web Application **Modeling**. Erich J. Neuhold ER 3-540-20299-4 Springer Lecture Notes in Computer Science Conceptual Modeling - ER 2003, 22nd International Conference on Conceptual Modeling, Chicago, IL, USA, October 13-16, 2003, Proceedings 2003 — 2

The M*-COMPLEX Approach to Enterprise Modeling, Engineering, and Integration. Giuseppe Berio, Antonio Di Leva ER (Workshops) 3-540-41073-2 Springer Lecture Notes in Computer Science Conceptual **Modeling** for E-Business and the Web, ER 2000 Workshops on Conceptual Modeling for E-Business and The World Wide Web and Conceptual Modeling, Salt Lake City, Utah, USA, October 9-12, 2000, Proceedings 2000 — 1

Taxonomy-Based Conceptual **Modeling** for Peer-to-Peer Networks. Carlo Meghini, Nicolas Spyratos, Yannis Tzitzikas ER 3-540-20299-4 Springer Lecture Notes in Computer Science Conceptual Modeling - ER 2003, 22nd International Conference on Conceptual Modeling, Chicago, IL, USA, October 13-16, 2003, Proceedings 2003 — 2

Fig. 6. Clustering for the token "Modeling" shown over five example citations using either (a) token trigrams or (b) label trigrams. Label trigrams are used for better precision in applying labels.

mutual information:

$$\phi_{MI}(Y_k) = \mathcal{I}(Y_k, \text{left}(Y_k)|\mathbf{O}) + \mathcal{I}(Y_k, \text{right}(Y_k|\mathbf{O}). \tag{14}$$

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 token's frequency and its influence/uncertainty.

5.4. Clustering by Information Density

In addition to selecting tokens that exhibit either the highest uncertainty or largest influence, we would like to construct questions in such a way that a single question can have maximum impact on the whole of the database. Equivalently, we would like the questions posed to be varied enough that we are not wasting financial resources asking the same question twice. A data-driven solution is to utilize simple clustering to group tokens together and map individual questions onto entire clusters instead of single tokens. As we show in our experiments, in tasks such as text segmentation and named entity recognition there are many redundant tokens that produce clusters of large size.

Our strategy for clustering is to collect tokens that *should* be labeled the same based on label criteria for our model. Since the CRF model contains features $f_j(H_{i-1}, H_i, O_i)$ that depend on observations at each token and their labeling neighborhood, we utilize the same aspects as a means for clustering. The *label trigram* method operates after the initial machine prediction and clusters together tokens H_i that that have the same previous and succeeding prediction labels (H_{i-1}, H_{i+1}) as well as the same observation token O_i. Though the features are sequential and do not consider succeeding tokens H_{i+1}), we include it in the clustering to differentiate tokens further and reduce possible errors. An example is shown in Figure 6, which clusters five documents into two clusters based on their appearing in either the TITLE or the SERIES as compared to a typical token trigram approach. Clustering algorithms are furthered compared with experimental results in Goldberg et al. [2013] with the conclusion that label trigrams produce superior results.

When tokens are selected for crowdsourcing, the entire context of the token is displayed to the user. Since each item in a cluster has its own independent context, we select a specific token at random to be the *representative token* for that cluster. When mapping clusters to questions, the representative token is the one specifically used to provide context and to formulate the question. When retrieving the answer to a question, the observed label is applied to all tokens in the representative token's cluster. The effect of clustering and constrained inference influences many tokens with only a single question.

The final concern is how to map the selection strategies introduced earlier in this section onto entire clusters. We looked at a number of strategies for aggregating ϕ_{ENT} and ϕ_{MI} including taking the max information function in the cluster (`MAX`), taking the average information function (`AVG`), and taking the sum of all information functions (`SUM`). Since clusters are unevenly distributed in size, we found a metric that takes into account cluster size to be preferable to one that does not. Therefore, taking the sum of all information functions in a cluster reflects both the high information redundancy (size of the cluster) and high uncertainty/influence (value of information function) in each question.

ALGORITHM 1: Selection Algorithm.

```
  input: Set of all tokens $\mathcal{T}$
  output: Ranked set $C$ of maximum information clusters
1 Initialize selected token set $S$;
2 Initialize cluster set $C$;
3 foreach $t \in \mathcal{T}$ do
      //Apply information function;
4     $t$.info $\leftarrow \phi(t)$;
      //Clustering;
5     Add $t$ to cluster $c(t, t$.label$, t$.prev_label$, t$.post_label$)$;
6     if $size(c) == 1$ then
7         $c$.rep_token $\leftarrow t$;
8     $c$.totalInfo $\leftarrow c$.totalInfo $+ t$.info;
9 SORT clusters $c \in C$ by $c$.totalInfoGain;
```

Algorithm 1 reviews the basic selection strategy for applying information functions to clusters and ranking them to determine the Top-k questions. We first iterate through all tokens in an initial pass applying the requisite information function. Since both ϕ_{ENT} and ϕ_{MI} employ simple approximate entropy calculations, they can be computed in constant time with respect to the token space. During the same iteration pass, tokens are hashed to a cluster according to the four-tuple $(H_{i-1}, H_i, H_{i+1}, O_i)$. These correspond to the token and its label as well as its neighbors' labels. The first token put into a cluster is made the representative token and all subsequent tokens contribute to a running sum of the information function associated with each cluster. Finally, the clusters are ordered by their total information functions and selected based on the budget to be mapped into questions.

6. HUMAN/MACHINE INTEGRATION

After data collection has occurred from Amazon Mechanical Turk, the system must aggregate their answers and integrate them back into the database. The integration component assumes the result coming back from the crowd is equivalent to a ground truth label. This necessitates a high quality method of deriving consensus answers.

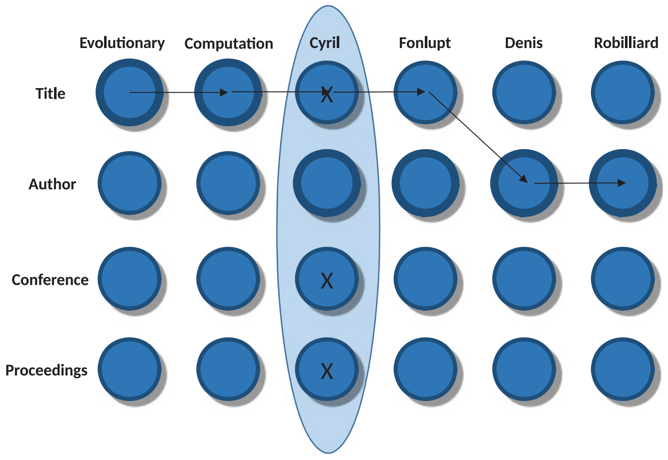


Fig. 7. Example maximum likelihood path through the label lattice. The token "Cyril" is chosen for querying and the result comes back that it is an `Author`. As a result, all paths not passing through `Author` are removed from the inference space.

Probabilistically aggregating various workers is an expanding area of research and there are many complex ways to combine answers into a single result. These include complex schemes to weight both workers [Demartini et al. 2012] and question difficulty [Whitehill et al. 2009]. See Sheshadri and Lease [2013] for a good survey of crowd aggregation techniques.

For our text extraction tasks, we find that Turkers perform particularly well and are able to achieve consistently high accuracy. The high capability enables simple aggregation methods like majority voting to be as powerful as some of the more complex methods. To be sure, we implemented a Bayesian-based [Raykar et al. 2010] aggregation method and found the results to be nearly identical to majority voting.

The majority vote aggregation produces a crowd-consensus label that must be integrated back into the database. The advent of Constrained Viterbi, as described in Section 4, allows us to use this newly acquired evidence to improve the machine's prediction on other labels through correction propagation. The crowd effectively becomes part of the inference step.

An example of how integration is performed is highlighted in Figures 7 and 8. First, in Figure 5 we have the original inference that associates a maximum likelihood path through the hidden label space. The selection process results in a node being sent to the crowd for observation, which "clamps" that node to its true label (Figure 7). The inference process is run again as a constrained inference. Because label values have dependencies on the previous label, a change in token t results in potential changes to $t - 1$ and $t + 1$, which can potentially propagate further down the chain (Figure 8).

Human feedback is stored in the database in a table with schema

$$\text{FeedbackTbl}(\text{strID}, \text{pos}, \text{token}, \text{label}).$$

This is similar to the TOKENTBL from Section 3, but the final label column is not probabilistic and corresponds to the crowd-consensus label. Using joins with a table containing all the cluster assignments, we can propagate the evidence to all tokens in the same cluster as the representative token.

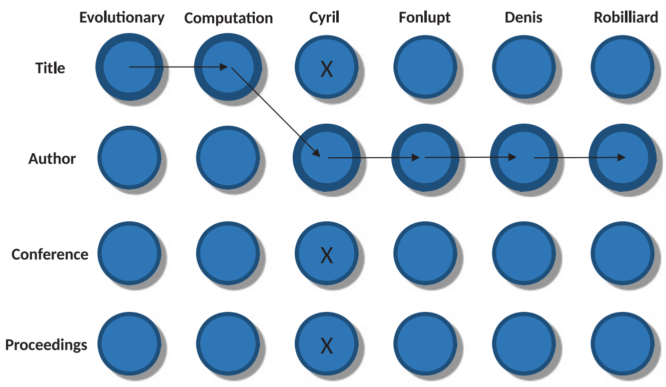


Fig. 8. New maximum likelihood path when the inference is constrained to pass through the `Author` label for the token "Cyril." The new constraints coupled with the existing features cause the token "Fonlupt" to be influenced and changed to an `Author` as well.

7. EXPERIMENTS

In the following section, we demonstrate the effectiveness of our approach applied to a text segmentation task and NER task. Given a semistructured document like an academic citation string, text segmentation is a problem concerned with partitioning the string into different classes. This information can be used to build a database of records that allow for efficient search and analysis. We also test a NER task where the goal is to extract entities such as people, places, and organizations from tweets. NER can be distilled to an annotation task where every token is labeled as a specific entity type or else `Other`. NER is a common preprocessing step to higher-order semantic algorithms.

In order to fully show how our methods perform when the machine learning task is only partially accurate, we apply our training and testing sets to different datasets. This is a common practice in knowledge and learning transfer as well as in practical applications where vendors are required to use off-the-shelf models that are difficult or expensive to retrain. Our results do not apply only in this domain and could just as easily be applied to standard structured prediction tasks.

Our pipeline consists of two key phases for which we provide experimental evidence of their utility. The first phase involves selection of information-dense tokens that would be sent to the crowd to answer. We compare our entropy and mutual information approximation along with other selection baselines. In the integration phase the true label for selected tokens is applied and constrained inference is performed. By comparing directly to the unconstrained selection phase experiments, we demonstrate the performance gain coming directly from the influence of selected tokens.

In the following sections, we describe the experimental setup and datasets in more detail before discussing the individual experiments.

7.1. Setup and Datasets

7.1.1. Text Segmentation. For training in the text segmentation task, we used the currently popular UMass citation dataset [Anzaroot and McCallum 2013]. State-of-the-art citation extraction has typically used the CORA dataset, which is much smaller and is being replaced by this current dataset. It contains 1800 bibliographic citations from physics, mathematics, computer science, and computational biology domains pulled from *Arxiv.org* and is fully labeled with 32 fine-grained (*FIRST NAME*, *LAST NAME, TITLE*, etc.) fields.

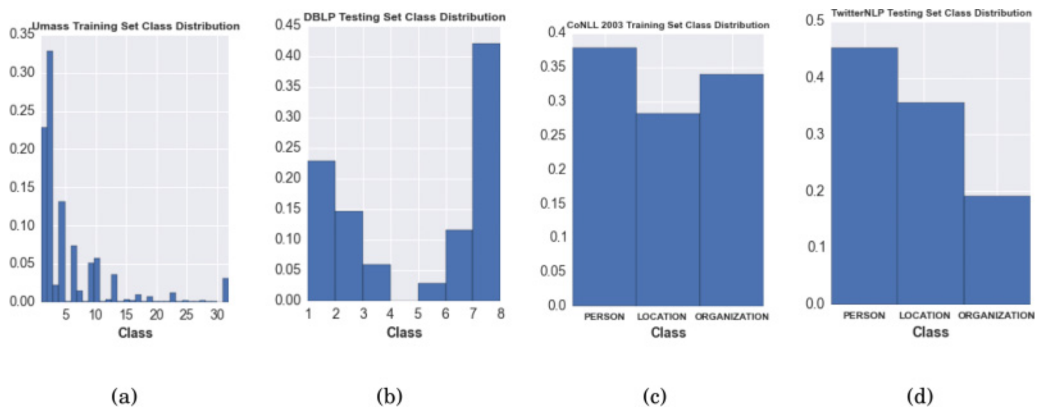


Fig. 9. Class distributions for Umass, DBLP, CoNLL 2003 Newswire, and TwitterNLP datasets. The top three UMass fields are `AUTHOR`(2), `TITLE`(1), and `JOURNAL`(4). The top three DBLP fields are `PROCEEDINGS`(7), `TITLE`(1), and `AUTHOR`(2).

The test set consists of 7K citations comprising 242K tokens extracted from the DBLP[2] database, which contains primarily computer science papers. Each citation is represented as a row in the database with columns for the seven classes: `TITLE`, `AUTHOR`, `CONFERENCE`, `SERIES`, `PROCEEDINGS`, `ISBN`, and `YEAR`. We reproduced the original citation by concatenating all of the string text together as a single semistructured document. From this we generated a training set and testing set based on a 50% split.

We used the IITB CRF[3] model for training and testing, which is heavily engineered to represent the state-of-the-art in the text segmentation task. The tokenizer it comes with includes commas and all numbers mapped to a single "DIGIT" token in the tokenization. We filtered these out for scoring.

The class distributions for both datasets are shown in Figure 9. Both UMass and DBLP sets contain citations from different domains and are structured in many different ways with differing pieces of information. While DBLP is more course than UMass, each of its labels maps onto either a direct label or set of labels in UMass. We computed this mapping in advance and translated the field sets prior to computing F1 scores.

7.1.2. Named Entity Recognition. For the named entity recognition task, we used the off-the-shelf Stanford NER[4] parser. This is the current state-of-the-art in NER and used widely throughout the literature. The model contains four classes (`PERSON`, `LOCATION`, `ORGANIZATION`, or `OTHER`) and is trained on the CoNLL 2003 shared task, which itself is a set of newswire articles from the Reuters corpus. This model performs best in its native newswire domain, but it is commonly applied to other domains as well.

We wanted to see how the Stanford NER parser would apply to a Twitter domain, using the TwitterNLP[5] dataset, which consists of 2,400 unstructured tweets comprising 34K total tokens. TwitterNLP contains many novel entities not found in the Stanford training set. In addition, the variability in punctuation and capitalization makes Twitter a very difficult domain for NER using purely machine learning. The original TwitterNLP included extraneous classes like `Product` or `TV-Show`, but we converted these to `Other`.

[2]http://dblp.uni-trier.de/.
[3]http://crf.sourceforge.net/.
[4]http://nlp.stanford.edu/software/CRF-NER.shtml.
[5]https://github.com/aritter/twitter_nlp.

7.1.3. Experimental Setup. As stated in Section 5, the goal of the selection problem is to select those tokens that are (1) the most likely to resolve errors by their own correction and (2) the most likely to improve incorrect neighbors through inference propagation. We evaluate by computing F1 scores on a token-by-token basis, where F1 is the harmonic mean of precision and recall. Precision is the ratio of correct tokens for a class to all predicted tokens for that class. Recall is the ratio of correct tokens to all true tokens for that class. Because both applications are instances of a multiclass classification problem, we use the microaveraged F1. This uses the sums of all true positives, false positives, and true negatives to compute a final F1. Given a budget of K questions, we seek to maximize the F1 gain from an initial machine-learning only baseline we can achieve with each question. Experiments show F1 increases incrementally as questions are posed.

The experiments that follow contain both synthetic and real crowdsourced data. The synthetic data is designed to test the selection and integration mechanisms which are the focus of this article independent of crowdsourcing uncertainty. We do this by substituting the true field in place of any selected tokens. It would also be untenable to attempt as many permutations of the experiments as we require using only real data. The use of synthetic oracle data allows us to significantly scale up the experiments. We include a small set of end-to-end experiments where we crowdsourced the answers using Amazon Mechanical Turk to verify the ability of the crowd to perform this task.

7.2. Selection Experiments

For all experiments, we perform a filtering step prior to clustering or selection that reduces the pool of available tokens. For each citation in the DBLP dataset or tweet in the Twitter dataset, we select either the highest Mutual Information or Entropy token depending on the experiment being considered. For our random baseline, we randomly select a token from each document. This reduces the DBLP pool to 7,000 tokens and the Twitter pool to 2,400 tokens. Then we either select or cluster and select depending on the experiment.

Figures 10 and 11 show how the F1 scores are improved using different selection metrics. The plots are differentiated by permutations of dataset, clustering, and constraining. Figure 10 shows experimental results for Umass to DBLP, while Figure 11 shows results for newswire to Twitter. The initial F1 without any human correction is 33.8% for training on Umass and testing on DBLP, while for training on CoNLL 2003 and testing on TwitterNLP the initial F1 is 56.6%. While these scores may seem low, this is standard for a highly difficult knowledge transfer task and is the perfect application for pi-CASTLE to exploit machines performing one-half or one-third the task while humans fill in the gaps. The x-axis shows how the total system F1 increases for every question we ask. For unclustered experiments, this corresponds to only a single token correction in the entire dataset. For clustered experiments, each question maps on a cluster of tokens, all of whom are given the crowdsourced label.

The baseline for comparison without clustering is randomly selecting tokens (`Rand`) for improvement, bypassing the filtering and ranking steps. If we perform clustering, we choose a different baseline (`Size` that randomly selects a token from each citation (i.e., filtering) and then clusters them and ranks them according to size. This generates a much stronger baseline and directly compares a baseline data-centric algorithm with our information-theoretic algorithms. `Ent` selects tokens whose predicted marginal label distribution has the highest entropy in the filtering step and ranks them either by token entropy or total cluster entropy. Similarly, `MI` selects those tokens with the highest mutual information approximation as discussed in Section 5 and clusters and ranks in the same fashion.

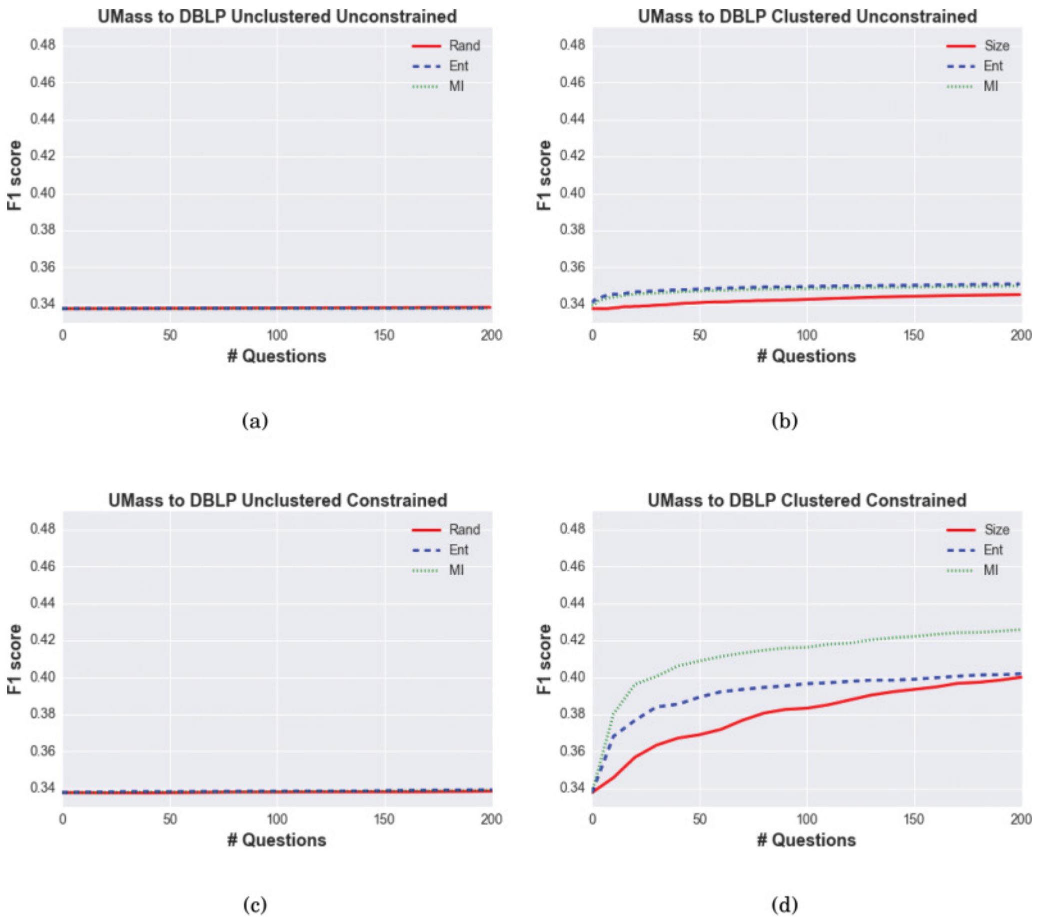
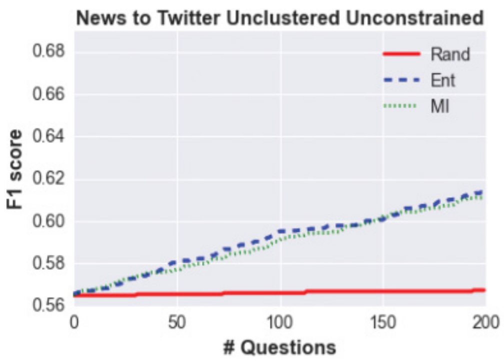


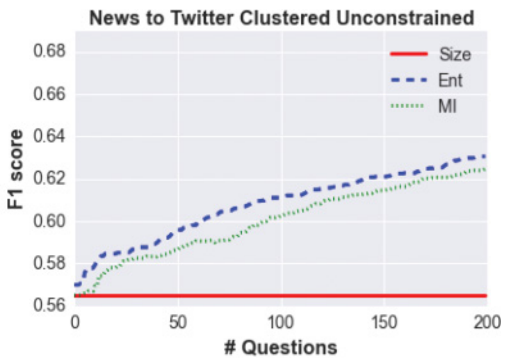
Fig. 10. F1 score increases for Umass training and DBLP testing.

The size of the DBLP test set is 242k tokens and without clustering, we are only at liberty to correct one token at a time per question asked. This results in mostly flat plots that do not at all differ by constraining. Clustering without constraining provides a marginal increase for both information-theoretic methods. Constraining the selected clusters, however, provides the largest increase across the board for all selection mechanisms. The performance of random is still quite good because when the F1 score is as poor as it is, even random corrections will result in some common incorrect tokens being clustered and corrected. Entropy outperforms it by selecting tokens more likely to be incorrect and result in score increases upon correction. Mutual Information performs strongest here because it selects those tokens most likely to improve other tokens.

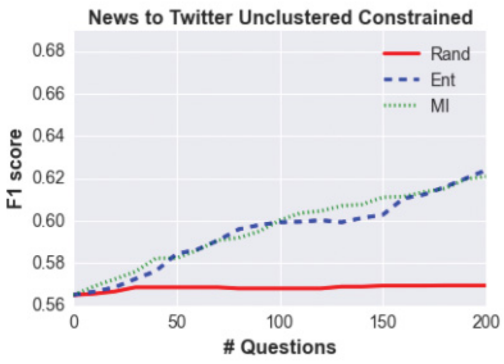
The performance on the Twitter test set is a bit closer for all plots compared to DBLP. There are two reasons for this. First, named entities appear more often as singleton mentions over a single token. This reduces the effect of constraining. Second, the increased variety among Twitter compared to semisupervised citations leads to a reduction in the impact of clustering. Nevertheless, not performing any type of clustering or constraining is still the weakest across the board, while performing both clustering and constraining leads to the overall strongest performance. Selecting by entropy is



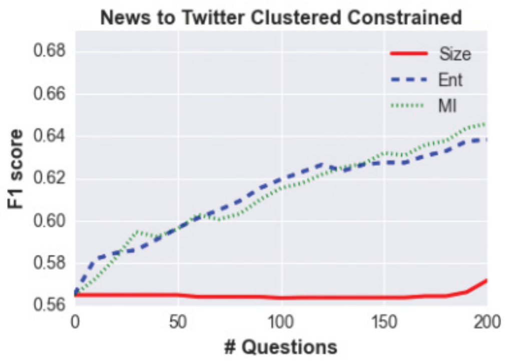
(a)



(b)



(c)



(d)

Fig. 11. F1 score increases for newswire training and Twitter testing.

best performer without constraining, but Mutual Information edges out slightly when constrained inference is performed. In all cases, the information-theoretic methods significantly outperform the random baselines.

There are a couple of additional observations on both sets of plots. In the Twitter testing set, some of the graphs appear to dip despite getting a ground truth correction. There are two additional sources of error our methods introduce into the problem. One comes from errors in clustering, where incorrectly clustered tokens receive the same label in error. Another comes from the constrained inference process, where neighboring tokens correctly labeled by chance are incorrectly labeled after applying corrections. Our experiments show both of these scenarios are rare and the benefits of clustering and constraining far outweigh the possible negative effects.

Of particular note is the small number of questions. For the DBLP set at 242k tokens, 200 questions represents only 0.1% of the whole dataset and would cost only about \$50 on Amazon Mechanical Turk. And yet using a Mutual Information framework with some question clustering we are able to improve the F1 score from 34% to 43%, a gain of nearly 33% of the original. This is much more efficient than selecting at the citation level for labeling, which requires more redundancy and is unable to take advantage of clustering. By allowing the machine to do most of the work, we can make the most

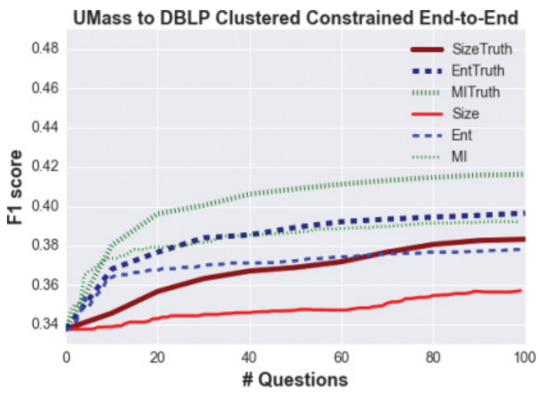


Fig. 12. End-to-end experiment using clustering and constrained inference.

necessary improvements with a small number of questions. This shows that iterating on this framework with batches of K questions would lead to higher performance with lowered costs than a higher granularity labeling process.

7.3. End-to-End Experiment

While we employed an oracle in order to show a number of experimental baselines and dissect where the advantages of our methods are coming from and how they apply to multiple datasets, we are compelled to confirm the utility of pi-CASTLE in practice. To this end, we set up an experiment that uses real crowdsourced labels in the correction process. We chose not to do this for all experiments in order to decouple extraction uncertainty from crowd uncertainty, the former being the focus of this article.

Using the DBLP test set, we applied filtering, and clustering using random, entropy, and Mutual Information metrics and ranked clusters according to the same metrics. We then selected a representative token from each cluster to send to the crowd. Whereas in the previous experiments the true label was supplied for each representative token, here we used the crowd consensus for the token, if there was one.

Workers were presented with an unstructured citation containing a bolded token and a multiple choice response for the class. They were provided two examples showing the proper classification of such tokens. Each HIT batched 10 citations together and cost a total of \$0.10, or \$0.01 for each citation. For redundancy, we five workers labeled each citation and we used the majority vote label in our experiments. The simplicity of our problem domain and question interface exempted the need for many of the more advanced truthing mechanisms that exist in literature that seek to model question difficulty and worker domain accuracy. Since we already had machine predictions, we valued precision of responses over recall. As such, we only replaced machine predictions if there was a crowd consensus of at least three out of five.

Figure 12 shows the results of our end-to-end experiment along with the upper-bound from the previous experiments that supplied the truth for every cluster. The crowd answered all HITs with an accuracy of about 80%, accounting for the small drop in F1 compared to the bounds. The relative performance among mutual information and entropy compared to random remains roughly the same. These experiments could reach closer to the upper bound by applying more advanced crowd training, interface design, and discrimination algorithms that are beyond the scope of this article. We do show, however, that even a naive crowdsourcing implementation shows a strong performance benefit when using superior selection algorithms for the hybridization process.

8. CONCLUSION

In this article, we introduced pi-CASTLE, a crowd-assisted SML-based IE system that can improve the accuracy of its automated results through a crowdsourced workforce. We developed two information functions and a clustering heuristic to formulate the most information-dense questions to the crowd given a fixed budget. Our experiments showed order-of-magnitude performance increases for a given set of questions compared to baselines.

While we focus on text extraction in the article, 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 human/machine integration are applicable to other SML-based tasks in the CAMeL framework.

REFERENCES

Sam Anzaroot and Andrew McCallum. 2013. A new dataset for fine-grained citation field extraction. In *Proceedings of the ICML Workshop on Peer Reviewing and Publishing Models*.

Katrin Braunschweig, Maik Thiele, Julian Eberius, and Wolfgang Lehner. 2013. Enhancing named entity extraction by effectively incorporating the crowd. In *BTW Workshops*, Vol. 13. 181–195.

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

Eleonora Ciceri, Piero Fraternali, Davide Martinenghi, and Marco Tagliasacchi. 2016. Crowdsourcing for top-k query processing over uncertain data. *IEEE Transactions on Knowledge and Data Engineering* 28, 1 (2016), 41–53.

Aron Culotta, Trausti Kristjansson, Andrew McCallum, and Paul Viola. 2006. Corrective feedback and persistent learning for information extraction. *Artificial Intelligence* 170, 14 (2006), 1101–1122.

Susan B. Davidson, Sanjeev Khanna, Tova Milo, and Sudeepa Roy. 2013. Using the crowd for top-k and group-by queries. In *Proceedings of the 16th International Conference on Database Theory*. ACM, 225–236.

Gianluca Demartini, Djellel Eddine Difallah, and Philippe Cudré-Mauroux. 2012. ZenCrowd: Leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking. In *Proceedings of the 21st International Conference on World Wide Web*. ACM, 469–478.

Ju Fan, Meiyu Lu, Beng Chin Ooi, Wang-Chiew Tan, and Meihui Zhang. 2014. A hybrid machine-crowdsourcing system for matching web tables. In *Proceedings of the 2014 IEEE 30th International Conference on Data Engineering (ICDE)*. IEEE, 976–987.

Oluwaseyi Feyisetan, Markus Luczak-Roesch, Elena Simperl, Ramine Tinati, and Nigel Shadbolt. 2015. Towards hybrid ner: A study of content and crowdsourcing-related performance factors. In *European Semantic Web Conference*. Springer, 525–540.

G. D. Forney. 1973. The Viterbi algorithm. *Procceedings of the 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. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*. ACM, 61–72.

Sean Goldberg, Daisy Zhe Wang, Jeff Depree, and Tim Kraska. 2013. CASTLE: A crowd-assisted system for textual labeling and extraction. In *Proceedings of the 2013 Conference on Human Computation (HCOMP'13)*.

Justin B. Kinney and Gurinder S. Atwal. 2014. Equitability, mutual information, and the maximal information coefficient. *Proceedings of the National Academy of Sciences* 111, 9 (2014), 3354–3359.

Sarath Kumar Kondreddi, Peter Triantafillou, and Gerhard Weikum. 2014. Combining information extraction and human computing for crowdsourced knowledge acquisition. In *Proceedings of the 2014 IEEE 30th International Conference on Data Engineering (ICDE)*. IEEE, 988–999.

Andreas Krause and Carlos Guestrin. 2009. Optimal value of information in graphical models. *Journal of Artificial Intelligence Research* (2009), 557–591.

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.

David D. Lewis and William A. Gale. 1994. A sequential algorithm for training text classifiers. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'94)*. Springer-Verlag, New York, NY, 3–12. http://dl.acm.org/citation.cfm?id=188490.188495

Dong C. Liu and Jorge Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical Programming* 45, 1–3 (1989), 503–528.

Adam Marcus, Eugene Wu, David Karger, Samuel Madden, and Robert Miller. 2011. Human-powered sorts and joins. *Proceedings of the VLDB Endowment* 5, 1 (2011), 13–24.

Barzan Mozafari, Purna Sarkar, Michael Franklin, Michael Jordan, and Samuel Madden. 2014. Scaling up crowd-sourcing to very large datasets: A case for active learning. *Proceedings of the VLDB Endowment* 8, 2 (2014), 125–136.

Lawrence R. Rabiner and Biing-Hwang Juang. 1986. An introduction to hidden Markov models. *IEEE ASSP Magazine* 3, 1 (1986), 4–16.

Vikas C. Raykar, Shipeng Yu, Linda H. Zhao, Gerardo Hermosillo Valadez, Charles Florin, Luca Bogoni, and Linda Moy. 2010. Learning from crowds. *The Journal of Machine Learning Research* 11 (2010), 1297–1322.

Cristina Sarasua, Elena Simperl, and Natalya F. Noy. 2012. Crowdmap: Crowdsourcing ontology alignment with microtasks. In *The Semantic Web–ISWC 2012*. Springer, 525–541.

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, 1070–1079.

Aashish Sheshadri and Matthew Lease. 2013. SQUARE: A benchmark for research on computing crowd consensus. In *First AAAI Conference on Human Computation and Crowdsourcing*.

Charles Sutton and Andrew McCallum. 2004. *Collective Segmentation and Labeling of Distant Entities in Information Extraction*. Technical Report TR # 04-49. University of Massachusetts.

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

Jacob Whitehill, Ting-fan Wu, Jacob Bergsma, Javier R. Movellan, and Paul L. Ruvolo. 2009. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *Advances in Neural Information Processing Systems*. 2035–2043.

Chen Jason Zhang, Lei Chen, Yongxin Tong, and Zheng Liu. 2015. Cleaning uncertain data with a noisy crowd. In *Proceedings of the 2015 IEEE 31st International Conference on Data Engineering (ICDE)*. IEEE, 6–17.

Received July 2015; revised October 2016; accepted October 2016