Proceedings of the First AAAI Conference on Human Computation and Crowdsourcing

CASTLE: Crowd-Assisted System for Text Labeling and Extraction

Sean Goldberg
Comp Info Sci & Eng
Univ. of Florida
sean@cise.ufl.edu

Daisy Zhe Wang
Comp Info Sci & Eng
Univ. of Florida
daisyw@cise.ufl.edu

Tim Kraska
Comp Sci
Brown University
tim_kraska@brown.edu

Abstract

The amount of text data has been growing exponentially and with it the demand for improved information extraction (IE) efforts to analyze and query such data. While automatic IE systems have proven useful in controlled experiments, in practice the gap between machine learning extraction and human extraction is still quite large. In this paper, we propose a system that uses crowdsourcing techniques to help close this gap. One of the fundamental issues inherent in using a large-scale human workforce is deciding the optimal questions to pose to the crowd. We demonstrate novel solutions using *mutual information* and token clustering techniques in the domain of bibliographic citation extraction. Our experiments show promising results in using crowd assistance as a cost-effective way to close up the "last mile" between extraction systems and a human annotator.

1 Introduction

In recent years, the amount of unstructured text data has been growing exponentially 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 the key to a slew of applications that depend on efficient search and analysis. 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) techniques such as hidden Markov models (HMM) and conditional random fields (CRF) (Lafferty, McCallum, and Pereira 2001). Most current IE systems store the maximum likelihood extraction into a database for querying, but such extractions from even the best models are still prone to errors.

Example 1 *Consider the following example citation:*

```
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 delimiter bewtween fields, the model may mislabel Xavier *as part of the* title *rather than* author. *It may also be confused between two possible extractions:* 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 validated extractions are re-introduced into the training set, this is known as *active learning*. While generally able to produce better results, human annotation is expensive and time-consuming. Platforms like Amazon Mechanical Turk (AMT) deploy Human Intelligence Tasks (HITs) that make it possible for the first time 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 CASTLE: a crowd-assisted SML-based IE system. CASTLE uses a probabilistic database to execute, optimize, and integrate human and machine computation for text extraction. The human computation aspect is based on crowdsourcing services and the machine computation on linear-chain CRF models. CASTLE initially uses a CRF to annotate all input text data. In contrast to other IE systems, however, CASTLE uses a probabilistic data model to store IE results, automatically includes humans to correct the most uncertain tokens, and integrates their responses back to the probabilistic data model.

CASTLE Architecture: Figure 1 outlines the basic architecture of CASTLE. The outer boxes partition the system into four main components: 1) Extraction & Labeling, 2) Question Selection, 3) HIT Management, and 4) the Probabilistic Database used for storage and data management.

CRF Extraction & Labeling transforms a set of unlabeled data into a set of probabilistic extractions which are stored in the database. After some pre-processing and feature extraction on the input data, inference is performed using a previously trained Conditional Random Field model. If the database contains any evidence from the crowd, the inference is constrained so certain hidden states conform to their evidence values.

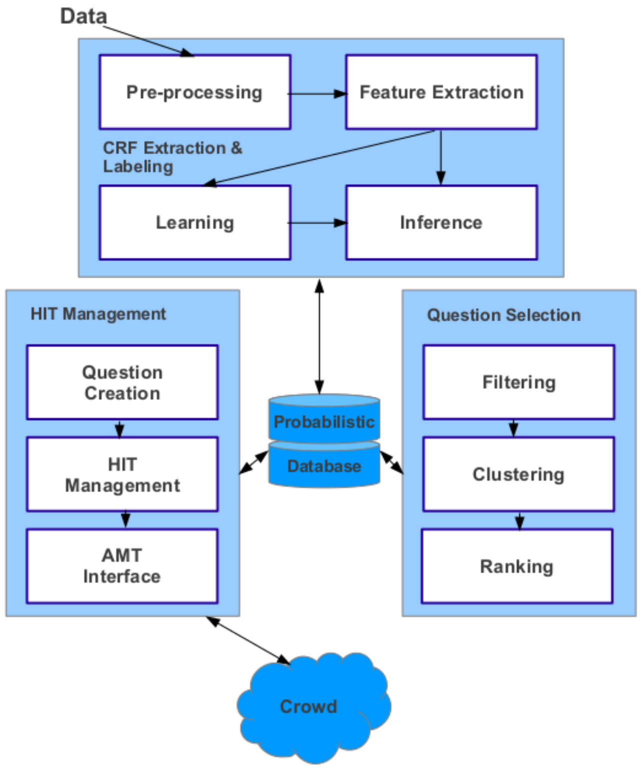


Figure 1: Architecture of the CASTLE system.

The Question Selection component acts on the database and determines which individual tokens to map into HITs for querying the crowd. Tokens are chosen by their within-document and cross-document importance using the concept of mutual information and a trigram-based clustering scheme. The three-step process of Filtering, Clustering, and Ranking will be described in Section 4.

We use Amazon Mechanical Turk as our crowdsourcing platform. All management settings for the HITs including interface, price, and qualification tests are handled by the HIT Management component.

The remainder of the paper is outlined as follows. Section 2 describes the initial extraction process using a CRF. Section 3 outlines our probabilistic data model. In Section 4 we cover the Question Selection problem in more detail and our proposed solutions. HIT Management is described in Section 5. Our experiments are found in Section 6, while Sections 7 & 8 cover the Related Work and our Conclusions.

2 CRF Extraction & Inference

The initial extraction and labeling of unstructured text is performed using a linear-chain Conditional Random Field (CRF) (Lafferty, McCallum, and Pereira 2001; Sutton and Mccallum 2006). In the context of IE, a CRF model encodes the probability distribution over a set of *label* random variables (RVs) $\mathbf{Y}$, given the value of a set of *token* RVs $\mathbf{X}$. 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 label y_{i-1} and token x_i. Such correlations are represented by

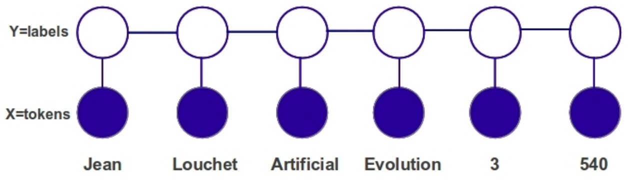


Figure 2: Example CRF model.

the feature functions $\{f_k(y_i, y_{i-1}, x_i)\}_{k=1}^{K}$.

Example 2 *Figure 2 shows an example CRF model over a subset of the citation string from Example 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:*

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

The conditional probability of a segmentation $\mathbf{y}$ given a specific token sequence $\mathbf{x}$ of T tokens is a weighted log-linear sum of feature functions:

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

where $Z(\mathbf{x})$ is a partition function and $\{\lambda_k) \in R$ are a collection of real-valued parameters.

There are three types of inference queries over the CRF model used in 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; Rabiner 1989) is the key algorithmic technique for CRF top-*k* inference.

The Viterbi algorithm computes a two-dimensional V matrix, where each cell $V(i, y)$ stores a ranked list of *entries* $e = \{score, prev(label, idx)\}$ ordered by a *score*. Each entry contains (1) the *score* of a top-*k* (partial) segmentation ending at position i with label y; and (2) a pointer to the previous entry *prev* on the path that led to the top-*k score's* in $V(i, y)$. The pointer e.prev consists of the label $label$ and the list index idx of the previous entry on the path to e. Based on equation 1, the recurrence to compute the ML (top-1) segmentation is as follows:

$$V(i,y) = \begin{cases} \max_{y'}(V(i-1,y') \\ +\sum_{k=1}^{K}\lambda_k f_k(y,y',x_i)), & \text{if } i \geqslant 0 \\ 0, & \text{if i} = -1 \end{cases} \qquad (2)$$

The complexity of the Viterbi algorithm is $O(T \cdot |L|^2)$, where $|L|$ is the number of possible labels.

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 from the crowd(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 label or a sub-sequence of labels (Sutton and Mccallum 2006). The Forward-Backward algorithm, a variation of the Viterbi algorithm is used for such marginal inference tasks.

CASTLE makes use of all three types. The initial top-1 extraction is used in the absence of any crowd input. For each token in each document in the corpus, marginal inference is performed as part of the Question Selection process (see next section). Finally, after the crowd has responded by labeling a selected set of tokens, constrained inference makes use of the available evidence by constraining the selected tokens to their crowd-appointed values. As more and more evidence is gathered, constrained inference converges to the same result had the crowd labeled the entire sequence.

3 Probabilistic Data Model

Probabilistic Databases arose out of the need to model large amounts of imprecise data. In the Possible Worlds Data Model (Dalvi and Suciu 2007), let $\mathbf{I} = \{I_1, I_2, ..., I_N\}$ be the set of all possible instances of a typical relational database. A PDB is the set $(I_i, p(I_i))$ of all instance-probability pairs, where $p(I_i) \rightarrow [0, 1]$ such that $\sum_{i=1}^{N} p(I_i) = 1$. Queries may be modified to return probabilistic results, though the number of possible worlds may grow exponentially with the size of the database. It is for this reason that queries generally return only the *top k* most probable results.

A *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(D^p) represent multiple viable instances of the database. The attributes of an incomplete relation $R \in \mathcal{R}$ may contain deterministic attributes, but include a subset that are *probabilistic attributes* $\mathcal{A}^p$. The values of $\mathcal{A}^p$ may be present, missing, or uncertain.

In CASTLE, the marginal probabilities of the token labels and the joint probabilities of pairs of labels which are updated over time as the crowd continues to reduce the uncertainty and constrained inference is performed. These marginal and joint probabilities are used to compute quantities like Mutual Information and Token Entropy in the selection process, which we describe in the next section. While we focus primarily on the question selection process in this paper, we plan to cover a more general probabilistic data model (Wang et al. 2008) in greater detail in future work.

4 Question Selection

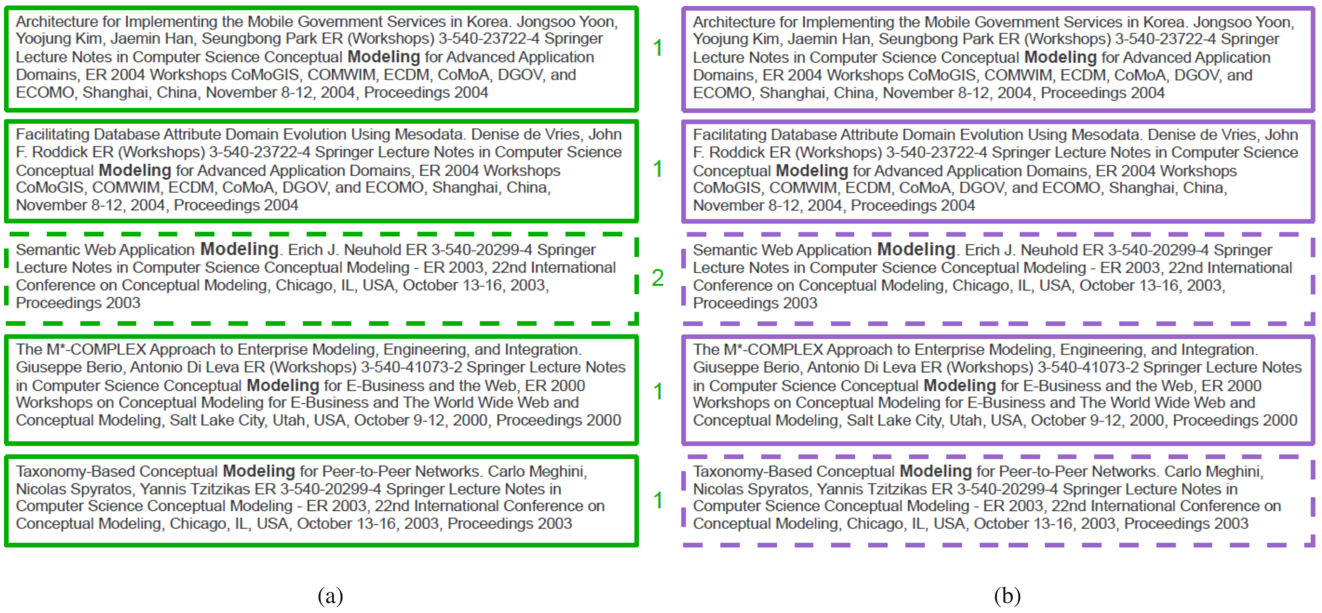
The problem of question selection is similar to that found in active learning where select examples are chosen from a pool of unlabeled data to be annotated based on some querying strategy. While active learning has been applied to the sequential learning domain (Cheng et al. 2008; Settles and Craven 2008), the financial and temporal cost of labeling an entire sequence (document) is not amenable to AMT's 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 reduce the posterior uncertainty in the results. By re-running inference with selected tokens constrained to their 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 infor-

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 $t \in \mathcal{T}$ do
    $i \leftarrow t$.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 $t \in Q$ do
    foreach cluster $c \in C$ 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 $\leftarrow$ $c$.totalInfoGain
            $+t$.totalInfoGain;
    if $t$ not added to a cluster then
        Initialize new cluster $c$;
        $c$.text $\leftarrow$ $t$.text;
        $c$.label $\leftarrow$ $t$.label;
        $c$.prevLabel $\leftarrow$ $t$.prevLabel;
        $c$.postLabel $\leftarrow$ $t$.postLabel;
        Add $c$ to cluster set $C$;
        $c$.totalInfoGain $\leftarrow$ $t$.totalInfoGain
//Ranking;
SORT clusters $c \in C$ by $c$.totalInfoGain;
```


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

(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 2

(b)

Figure 3: Clustering for the token ”Modeling” shown over five example citations with each line type denoting a different cluster. Example clusters are shown using (a) token trigrams and (b) label trigrams for clustering. Note that clustering by labels permits the 3rd and 5th citations to both be labeled as Title while clustering by tokens produces an incorrect clustering in this scenario.

mation content 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

$$\phi^{TE}(\mathbf{x_i}) = -\sum_{l=1}^{L} P(y_i = l)\log P(y_i = l), \tag{3}$$

where the sum is over the range of labels L and $P(y_i = l)$ is the marginal probability that token $\mathbf{x_i}$ is 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* to select those tokens whose result will have the greatest significance on its neighbors 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, Akella, and Zhang 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.

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

$$I(\mathcal{X};\mathcal{Y}) = H(\mathcal{X}) + H(\mathcal{Y}) - H(\mathcal{X},\mathcal{Y}). \tag{4}$$

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

$$\phi^{MI}(\mathbf{x_i}) = H(\mathbf{x_i}) + H(\mathbf{x_1}, \ldots, \mathbf{x_n} \backslash \mathbf{x_i}) - H(\mathbf{x_1}, \ldots, \mathbf{x_n}), \quad (5)$$

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 loosely states that information processed along a Markov chain cannot increase, ie. for a chain $X \rightarrow Y \rightarrow Z$,

$$I(X; Y) \geqslant I(X; Z) \quad (6)$$

We approximate 5 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 most impact on its most immediate neighbors $\mathbf{x_{i-1}}$ and $\mathbf{x_{i+1}}$,

$$\begin{aligned}\phi^{MIapprx}(\mathbf{x_i}) =& I(\mathbf{x_{i-1}}; \mathbf{x_i}) + I(\mathbf{x_i}; \mathbf{x_{i+1}}) \\ =& H(\mathbf{x_{i-1}}) + 2H(\mathbf{x_i}) + H(\mathbf{x_{i+1}}) \\ & - H(\mathbf{x_{i-1}}, \mathbf{x_i}) - H(\mathbf{x_i}, \mathbf{x_{i+1}}). \quad (7)\end{aligned}$$

The entropies in 7 can be efficiently computed using the *forward-backward* algorithm (Rabiner 1989) to compute the marginal and joint probabilities, then calculating the entropy 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 the frequency of a token's appearance along with its uncertainty.

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. This fails to harness the parallelizability of the crowd ecosystem. It is much more effective to 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. The final batch must be *diverse* and contain only one token from each cluster. It must also be *dense* and comprise the largest clusters whose labeling will have the greatest effect.

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 equivalent out of context. This is less than desirable in a text segmentation problem 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 only 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, only a single "representative token" is chosen (the one with the highest mutual information or token entropy) and its crowd-annotation applied to all the middle tokens in the cluster. Figure 2(a) shows an example token trigram clustering.

In certain 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's 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. The 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.

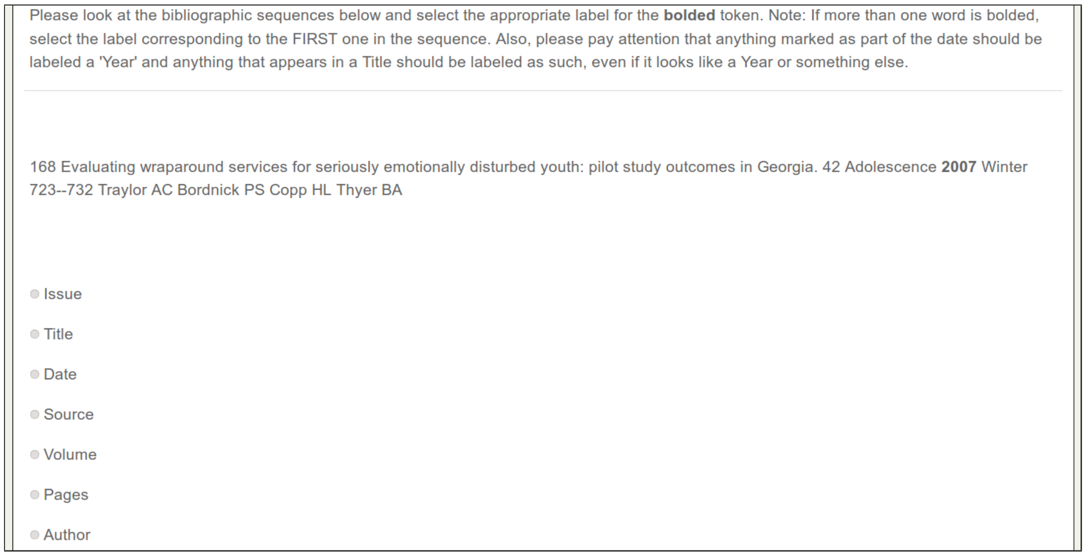
A token $\mathbf{x_i}$ has a set of associated labels $\mathbf{l_{i,j}}$, where i again denotes label position and j some numerical representation of the classifier type. For example, $\mathbf{l_{i,0}}$ might be the POS tag associated with $\mathbf{x_i}$ while $\mathbf{l_{i,1}}$ might be a segmentation tag. A **label trigram** clustering model consists of tokens that share some specified set of label trigrams. One possible cluster would be $(\mathbf{x_i}, (\mathbf{l_{i-1,1}}, \mathbf{l_{i,1}}, \mathbf{l_{i+1,1}}))$, 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 tokens $\mathbf{x_i}$ should still be the same. Figure 2(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 slightly increased 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).

Ranking by Total Information Gain

Given a limited budget of questions, clusters should be ordered to facilitate selection of the top-*k*. We experimented


Please look at the bibliographic sequences below and select the appropriate label for the **bolded** token. Note: If more than one word is bolded, select the label corresponding to the FIRST one in the sequence. Also, please pay attention that anything marked as part of the date should be labeled a 'Year' and anything that appears in a Title should be labeled as such, even if it looks like a Year or something else.

168 Evaluating wraparound services for seriously emotionally disturbed youth: pilot study outcomes in Georgia. 42 Adolescence **2007** Winter 723--732 Traylor AC Bordnick PS Copp HL Thyer BA

- Issue
- Title
- Date
- Source
- Volume
- Pages
- Author

Figure 4: Sample Amazon Mechanical Turk HIT interface. The question asks for the true label of the bolded token '2007', which is ambiguous because '2007' could refer to the Date field or be part of the Source title. In this example it's the latter.

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. Total information gain represents a melding of the both cluster size and MI score and outperformed the baselines. For the remainder of this paper, ranking is by total information gain.

5 HIT Management

The HIT Management component interacts with the Amazon Mechanical Turk API for processing, submitting, and retrieving HIT tasks. Tokens selected for crowd annotation are mapped into multiple choice questions, an example of which is given in Figure 4. As Turkers complete the tasks, answers are retrieved and immediately sent back to the Extraction & Labeling component where constrained inference is performed. The management component has settings for the price of each HIT, length of time for which answers are allowed, and number of Turkers assigned to each HIT.

Quality control is an important issue in AMT tasks. To reduce the likelihood of malicious or poor workers, we required an unpaid qualification test be submitted for any additional work may be completed. This test both trains workers and weeds out those unable to complete the task effectively. As an additional measure of quality control, the HIT Management component assigns a redundant number of HITs for each question and takes a majority vote of all Turkers assigned or however many were completed within the prescribed time limit. While we stick to multiple choice questions in this paper, as a future work this component will eventually be expected to handle multiple question types each with their own interfaces and settings. We also expect to handle more complex quality control measures, such as the currently popular Dawid & Skene method (Ipeirotis, Provost, and Wang 2010).

6 Experiments

In this section we demonstrate the effectiveness of applying mutual information and density trigram models for selection on the task of automatic bibliographic segmentation.

Setup and Data Set

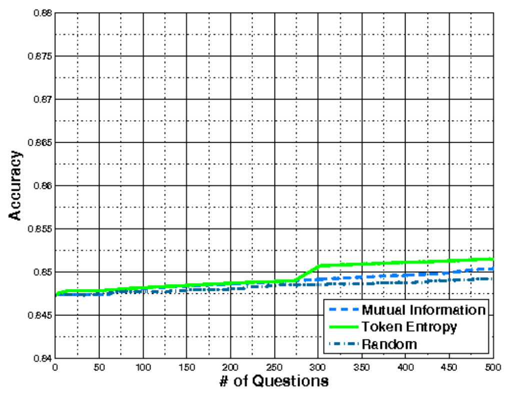
We extracted 14,000 entries from the DBLP[1] database. We produced a set of 7,000 labeled citations for training and 7,000 unlabled for testing by extracting fields from the database and concatenating them into full citations. Order of fields was occasionally mixed in keeping with real-life inconsistency of citation structure.

We tested the efficacy of our system using a set of ”end-to-end” experiments. That is, we started with a set of unlabeled data and proceeded through each of the system components: initial labeling by a trained CRF[2], selection and clustering using the methods defined earlier, submission to AMT, and finally retrieval and constrained inference.

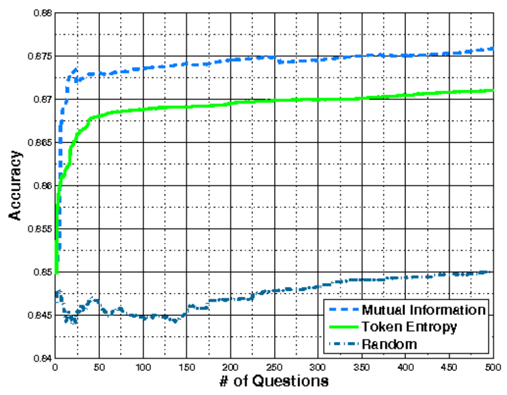
From a total testing set of 36,321 tokens, the top 500 using a combination of different methods were selected and sent to the crowd as multiple choice questions. Individual HITs were batched to contain 10 questions for which we awarded $.10 per Turker. Five Turkers passing a qualification test were assigned to each question for quality control and we set a limit of 3 days to allow enough time for completion by all assigned Turkers. Our quality control methods proved effective with Turkers providing answers to questions with 97% accuracy. The next section shows our results when those answers are integrated back into the database.

[1]http://kdl.cs.umass.edu/data/dblp/dblp-info.html

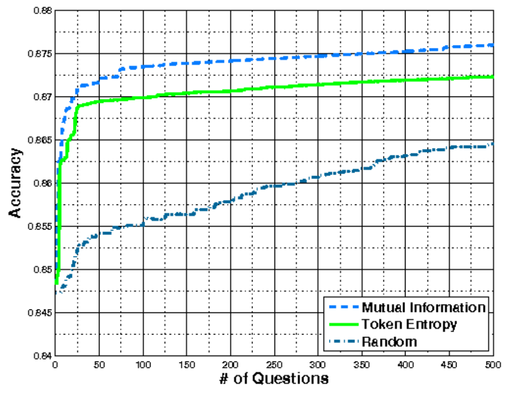
[2]http://crf.sourceforge.net/



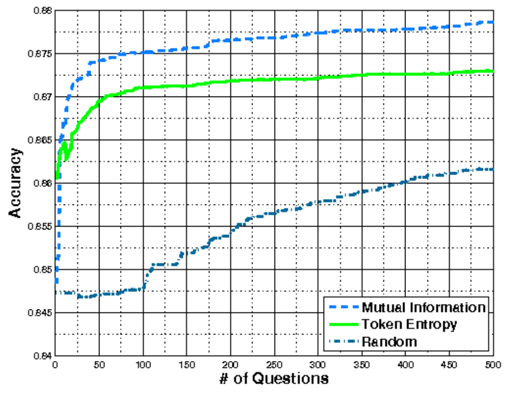
(a)



(b)



(c)



(d)

Figure 5: A comparison of different selection methods using (a) no clustering, (b) no context clustering, (c) same token trigram clustering, and (d) same label trigram clustering.

End-to-end Results

The purpose of CASTLE's human component is to provide evidence that constrained inference can use to improve the extraction results. We gauged performance on the individual token accuracy before obtaining evidence and after. The token accuracy is the number of individual tokens correctly labeled compared to the total in the corpus. Initially, the CRF labeled 84.6% of the tokens correctly.

Figures 5(a) - 5(d) show the ability of the constrained inference process to improve the extraction results for various combinations of selection and clustering methods. Each point on the x-axis corresponds to selecting that number of questions to obtain evidence for, ordered by clusters with the largest total token entropy or mutual information. Where random selection was done, clusters were ordered by cluster size. For each of the 500 answers received, all tokens belonging to the same cluster were given the annotation of the ”representative”. This resulted in large gains for some of the biggest clusters with one answer in particular addressing 390 individual tokens in the best case. This is the cause for the large jumps early on as the biggest clusters provide the biggest accuracy gains.

Both token entropy and mutual information filtering and ranking methods show sufficiently large gains over a random baseline, with mutual information showing the strongest improvement in accuracy. Use of any the clustering methods (no context, token trigram, or label trigram) lead to large gains in accuracy compared to a lack of clustering. The combination of filtering and clustering producing the strongest results is mutual information combined with label trigram clustering as seen in Figure 5(d). After only 50 questions, corresponding to correcting only .001% (50/36,000) of total tokens in the database, we achieved a reduction in error of just under 20% (84.6% to 87.5%). While returns begin to diminish, the overall reduction in error is as large as 25%. The total cost for this large error reduction was only $25. While we truncated our experiments at 500 questions, in principle one could continue asking questions and see even larger, if less efficient, gains.

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, 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.

8 Conclusion

In this paper we introduced CASTLE, a crowd-assisted SML-based IE system that is able to obtain additional evidence from a crowdsourced workforce to improve the accuracy of its automated results. There has been previous research into automating crowdsourcing for additional evidence retrieval in SML models, but CASTLE is the first to incorporate it into a cohesive system with a probabilistic data model as its backbone. Question selection strategies including highest mutual information and largest information density through trigram clustering produce questions with the benefit orders of magnitude greater than random selection. We believe CASTLE represents an important step in efficiently combining the benefits of human and machine computation into a singular system.

9 Acknowledgements

This work was partially supported by DARPA under FA8750-12-2-0348-2 (DEFT/CUBISM). We also thank Jeff Depree for his assistance in the database implementation and Sunny Khatri for work done on the CRF.

References

Boim, R.; Greenshpan, O.; Milo, T.; Novgorodov, S.; Polyzotis, N.; and Tan, W. C. 2012. Asking the right questions in crowd data sourcing. In Kementsietsidis, A., and Salles, M. A. V., eds., *ICDE*, 1261–1264. IEEE Computer Society.

Cheng, H.; Zhang, R.; Peng, Y.; Mao, J.; and Tan, P.-N. 2008. Maximum margin active learning for sequence labeling with different length. In Perner, P., ed., *ICDM*, volume 5077 of *Lecture Notes in Computer Science*, 345–359. Springer.

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

DeRose, P.; Shen, W.; Chen, F.; Lee, Y.; Burdick, D.; Doan, A.; and Ramakrishnan, R. 2007. Dblife: A community info. mgmt platform for the database research community (demo). In *CIDR*, 169–172.

Forney, G. D. 1973. The viterbi algorithm. *Proceedings of the IEEE* 61(3):268–278.

Franklin, M. J.; Kossmann, D.; Kraska, T.; Ramesh, S.; and Xin, R. 2011. Crowddb: answering queries with crowdsourcing. In Sellis et al. (2011), 61–72.

Guo, Y., and Greiner, R. 2007. Optimistic active-learning using mutual information. In Veloso, M. M., ed., *IJCAI*, 823–829.

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

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

Lafferty, J. D.; McCallum, A.; and Pereira, F. C. N. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, 282–289.

Marcus, A.; Wu, E.; Karger, D. R.; Madden, S.; and Miller, R. C. 2011. Demonstration of qurk: a query processor for humanoperators. In Sellis et al. (2011), 1315–1318.

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

Rabiner, L. 1989. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE* 77(2):257–286.

Sellis, T. K.; Miller, R. J.; Kementsietsidis, A.; and Velegrakis, Y., eds. 2011. *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2011, Athens, Greece, June 12-16, 2011*. ACM.

Settles, B., and Craven, M. 2008. An analysis of active learning strategies for sequence labeling tasks. In *EMNLP*, 1070–1079. ACL.

Sutton, C., and Mccallum, A. 2006. *Introduction to Conditional Random Fields for Relational Learning*. MIT Press.

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

Wang, J.; Kraska, T.; Franklin, M. J.; and Feng, J. 2012. Crowder: Crowdsourcing entity resolution. *PVLDB* 5(11):1483–1494.

Xu, Z.; Akella, R.; and Zhang, Y. 2007. Incorporating diversity and density in active learning for relevance feedback. In Amati, G.; Carpineto, C.; and Romano, G., eds., *ECIR*, volume 4425 of *Lecture Notes in Computer Science*, 246–257. Springer.

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