Resource-bounded Information Extraction: Acquiring Missing Feature Values On Demand

Pallika Kanani
UMass, Amherst
USA
pallika@cs.umass.edu

Andrew McCallum
UMass, Amherst
USA
mccallum@cs.umass.edu

Shaohan Hu
Dartmouth College
USA
shaohan.hu@dartmouth.edu

ABSTRACT

We present a general framework for the task of extracting specific information "on demand" from a large corpus such as the Web under resource-constraints. Given a database with missing or uncertain information, the proposed system automatically formulates queries, issues them to a search interface, selects a subset of the documents, extracts the required information from them, and fills the missing values in the original database. We also exploit inherent dependency within the data to obtain more useful information with fewer computational resources. We build such a system in the citation database domain that extracts the missing year of publications using limited resources from the Web. We discuss a probabilistic approach for this task and present first results. The main contribution of this paper is to propose a general, comprehensive architecture for designing a system that can be adapted to many different domains.

1. INTRODUCTION

The goal of traditional information extraction is to accurately extract as many fields or records as possible from a collection of unstructured or semi-structured text documents. In many scenarios, however, we already have a partial database and we need only fill in its holes. This paper proposes methods for finding such information in a large collection of external documents, and doing so efficiently with limited computational resources. For instance, this small piece of information may be a missing record, or a missing field in a database that would be acquired by searching a very large collection of documents, such as the Web. Using traditional models of information extraction for this task is wasteful, and in most cases computationally intractable. A more feasible approach for obtaining the required information is to automatically issue appropriate queries to the external source, select a subset of the retrieved documents for processing and then extract the specified field in a focussed and efficient manner. We can further enhance the efficiency of our system by exploiting the inherent relational nature of the database. We call this process of searching and extracting for specific pieces of information, on demand, *Resource-bounded Information Extraction* (RBIE). In this paper, we present the design of a general framework for Resource-bounded Information Extraction, discuss various important design choices involved and present some experimental results.

1.1 Example

Consider a database of scientific publication citations, such as Rexa, Citeseer or Google Scholar. The database is created by crawling the web, downloading papers, extracting citations from the bibliographies and then processing them by tagging and normalizing. In addition, the information from the paper header is also extracted. In order to make these citations and papers useful to the users, it is important to have the year of publication information available. Even after integrating the citation information with other publicly available databases, such as DBLP, a large fraction of the papers do not have a year of publication associated with them. This is because, often, the headers or the full text of the papers do not contain the date and venue of publication (especially for preprints available on the web). Approximately one third of the papers in Rexa are missing the year of publication field. Our goal is to fill in the missing years by extracting them from the web.

We chose this particular example task, since it demonstrates the relational aspect of RBIE. Along with extracting headers, the bibliographic information is often extracted, creating a citation network. This network information can be further exploited by noting that in almost all cases, a paper is published before (or the same year) as other papers that cite it. Using these temporal constraints, we obtain 87.7% of the original F1 by using only 13.2% of computational resources such as queries and documents.

1.2 Motivation

The knowledge discovery and data mining community has long struggled with the problem of missing information. Most real-world databases come with holes in the form of missing records or missing feature values. In some cases, the values exist, but there is considerable uncertainty about their correctness. Incompleteness in the database provides incomplete responses to user queries, as well as leads to less accurate data mining models and decision support systems. In order to make the best use of the existing information, it is desirable to acquire this missing information from an external source in an efficient manner. The external source can be another database that may be purchased, or a large

collection of free documents, such as the web. In the latter case, we may run information extraction to obtain the missing values in our database. However, the traditional models of information extraction can not be directly applied in this "on demand" setting.

Note that, in the setting described above, we are often not interested in obtaining the complete records on the database, but in just filling in the missing values. Also, the corpus of documents, such as the web, is extremely large. Moreover, in most real scenarios, we must work under pre-specified resource constraints. The resource constraints may be computational, such as processors, network bandwidth, or related to time and money. Any method that aims to extract required information in the described setting must be designed to work under the given resource constraints.

Many of these databases are relational in nature, e.g. obtaining the value of one field may provide useful information about the remaining fields. Similarly, if the records are part of a network structure with uncertain or missing values, as in the case of our example task, then information obtained for one node can reduce uncertainty in the entire network. We show that exploiting these kinds of dependencies can reduce the amount of resources required to complete the task significantly.

In this paper, we propose a general framework for resource-bounded information extraction, along with the design of a prototype system used to address the task of finding missing years of publication in citation records. We also present first results on this task.

2. RELATED WORK

Resource-bounded Information Extraction encompasses several different types of problems. It deals with extracting information from a large corpus, such as the web; it actively acquires this information under resource constraints; and it exploits the interdependency within the data for best performance. Here we discuss various related tasks and how RBIE is uniquely positioned between them.

2.1 Traditional Information Extraction

In the traditional information extraction settings, we are usually given a database schema, and a set of unstructured or semi-structured documents. The goal of the system is to automatically extract records from these documents, and fill in the values in the given database. These databases are then used for search, decision support and data mining. In recent years, there has been much work in developing sophisticated methods for performing information extraction over a closed collection of documents, e.g. [6]. Several different approaches have been proposed for different phases of information extraction task, such as segmentation, classification, association and coreference. Most of these proposed approaches make extensive use of statistical machine learning algorithms, which have improved significantly over the years. However, only some of these methods remain computationally tractable as the size of the document corpus grows. In fact, very few systems are designed to scale over a corpus as large as, say, the Web [5, 21].

2.2 Information Extraction From the Web

There are some large scale systems that extract information from the web. Among these are KnowItAll [5], InfoSleuth [16] and Kylin [20]. The goal of the KnowItAll system is a related, but different task called, "Open Information Extraction". In Open IE, the relations of interest are not known in advance, and the emphasis is on discovering new relations and new records through extensive web access. In contrast, in our task, what we are looking for is very specific and the corresponding schema is known. The emphasis is mostly on filling the missing fields in known records, using resource-bounded web querying. Hence, KnowItAll and RBIE frameworks have very different application domains. InfoSleuth focuses on gathering information from given sources, and Kylin focuses only on Wikipedia articles. These systems also do not aim to exploit the inherent dependency within the database for maximum utilization of resources.

The Information Retrieval community is rich with work in document relevance (TREC). However, traditional information retrieval solutions can not directly be used, since we first need to automate the query formulation for our task. Also, most search engine APIs return full documents or text snippets, rather than specific feature values.

A family of methods closely related to RBIE, is question answering systems [11]. These systems do retrieve a subset of relevant documents from the web, along with extracting a specific piece of information. However, they target a single piece of information requested by the user, whereas we target multiple, interdependent fields of a relational database. They formulate queries by interpreting a natural language question, whereas we formulate and rank them based on the utility of the information within the database. They do not address the problem of selecting and prioritizing instances or a subset of fields to query. This is why, even though some of the components in our system may appear similar to that of QA systems, their functionalities differ. The semantic web community has also been working on similar problems, but their focus is not targeted information extraction.

2.3 Active Information Acquisition

Learning and acquiring information under resource constraints has been studied in various forms. There are three different scenarios at training time. The most common scenario is *active learning* [3], which assumes access to unlabeled instances with complete feature values and attempts to select the most useful instances for which to acquire class labels while training. The next scenario is *active feature acquisition*, which explores the problem of learning models from incomplete instances by acquiring additional features. The general case of acquiring randomly-missing values in the instance-feature matrix is addressed in [15] . Under the budgeted learning scenario [12], the total cost to be spent towards acquisitions is determined a priori and the task is to identify the best set of acquisitions for this cost. Finally, more recent work [18] deals with learning models using noisy labels. At test time, the two common scenarios are selecting a subset of features to acquire [19, 1, 10], and selecting the subset of instances for which to acquire features [9, 8].

The interdependency within the data set is often conveniently modeled using graphs, but it poses interesting questions about selection of instances to query and propagating uncertainty through the graph [7]. In [8], the test instances are not independent of each other, and the impact of acquisition in the context of graph partitioning is studied. Similar problems are addressed in [2, 17]. Ideas from other fields, such as graph theory [4] and circuit design [13] can also be

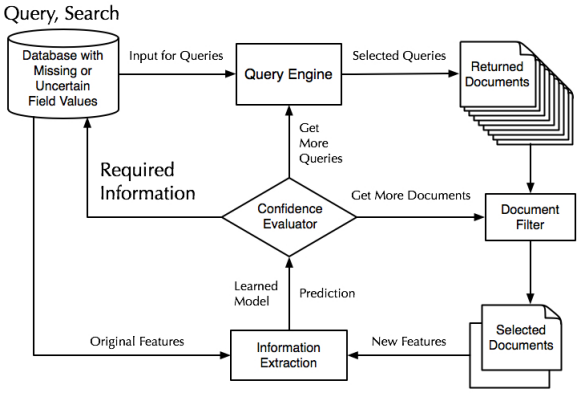


Figure 1: General Framework for Resource-bounded Information Extraction

borrowed in this context. The general RBIE framework described in this paper aims to leverage these methods for both train and test time for optimization of query and instance selection, depending on the application scenario.

In summary, RBIE requires a comprehensive architecture for efficiently integrating multiple functionalities, such as instance and query selection, automatic query formulation, and targeted information extraction by exploiting inherent data dependency under limited resources. This leads us to the new framework presented in this paper.

3. A GENERAL FRAMEWORK FOR RESOURCE-BOUNDED INFORMATION EXTRACTION

As described in the previous section, we need a new framework for performing information extraction to automatically acquire specific pieces of information from a very large corpus of unstructured documents. Fig. 1 shows a top-level architecture of our proposed framework. In this section, we discuss the general ideas for designing a resource-bounded information extraction system. Each of these modules may be adapted to suit the needs of a specific application, as we shall see for our example task.

3.1 Overview of the Architecture

We start with a database containing missing values. In general, the missing information can either be a complete record, or values of a subset of the features for all records, or a subset of the records. We may also have uncertainty over the existing feature values that can be reduced by integrating external information. We assume that the external corpus provides a search interface that can be accessed automatically, such as a search engine API.

The information already available in the database is used as an input to the *Query Engine*. The basic function of the query engine is to automatically formulate queries, prioritize them optimally, and issue them to a search interface. The documents returned by the search interface are then passed on to the *Document Filter*. Document Filter removes documents that are not relevant to the original database and ranks the remaining documents according to the usefulness of each document in extracting the required information.

A machine learning based information extraction system extracts relevant features from the documents obtained from the Document Filter, and combines them with the features obtained from the original database. Hence, information from the original database and the external source is now merged, to build a new model that predicts the values of the missing information. In general, we may have resource constraints at both training and test times. In the training phase, the learned model is passed to the *Confidence Evaluation System*, which evaluates the effectiveness of the model learned so far and recommends obtaining more documents through Document Filter, or issuing more queries through the Query Engine in order to improve the model. In the test phase, the prediction made by the learned model is tested by the Confidence Evaluation System. If the model's confidence in the predicted value crosses a threshold, then it is used to fill (or to replace a less certain value) in the original database. Otherwise, the Confidence Evaluation System requests a new document or a new query to improve the current prediction. This loop is continued until either all the required information is satisfactorily obtained, or we run out of a required resource. Additionally, feedback loops can be designed to help improve performance of Query Engine and Document Filter.

This gives a general overview of the proposed architecture. We now turn to a more detailed description for each module, along with the many design choices involved while designing a system for our specific task.

3.2 Task Example: Finding Paper's Year of Publication

We present a concrete resource-bounded information extraction task and a probabilistic approach to instantiate the framework described above.

We are given a set of citations from a database such as Rexa, with fields, such as, paper title, author names, contact information available, but the year of publication is missing. The goal is to search the web and extract this information from web documents to fill in the missing year values. We evaluate the performance of our system by measuring the precision, recall and F1 values at different confidence levels. The following sections describe the architecture of our prototype system, along with possible future extensions.

3.3 Query Engine

The basic function of query engine is to automatically formulate queries, prioritize them optimally, and issue them to a search interface. There are three modules of query engine.

As discussed in the previous section, the available resources may allow us to acquire the values for only a subset of the fields, for a subset of the records. Input selection module decides which feature values should be acquired from the external source to optimize the overall utility of the database. The query formulation module combines input values selected from the database with some domain knowledge, and automatically formulates queries. For instance, a subset of the available fields in the record, combined with a few keywords provided by the user, can form useful queries. Out of all the queries formulated, some queries are more successful than others in obtaining the required information. By ranking the queries in an optimal order, we require fewer queries to obtain the missing values. In the future, we would like to explore sophisticated query ranking methods, based

on the feedback from other components of the system.

In our system, we use existing fields of the citation, such as paper title and names of author, and combine them with keywords such as "cv", "publication list", etc. to formulate the queries. We experiment with the order in which we select which citations to query. In one method, the nodes with most incoming and outgoing citation links are queried first. We issue these queries to a search engine API and the top n hits (where n depends on the available resources) are obtained. The documents are first tokenized, the tokens which are probable years are tagged, and the tagged documents are passed on to the document filter.

3.4 Document Filter

The primary function of the document filter is to remove irrelevant documents and prioritize the remaining documents, so that most useful documents get processed first. Following are the two main components of the Document Filter.

Even though queries are formed using the fields in the database, some documents may be irrelevant. This may be due to the ambiguities in the data (e.g. person name coreference), or simply imperfections in retrieval engine. In many scenarios, it may be efficient to run an initial filter to remove those documents which are completely irrelevant to the original database. The remaining documents can then be ranked, based on their relevance to the original database. Remember that the relevance used by the search interface is with respect to the queries, which may not necessarily be the same as the relevance with respect to the original database (depending on the type of query). In the future, we would like to learn a ranking model, based on the feedback from the information extraction module (via *Confidence Evaluation System*) about how useful the document was in making the actual prediction.

In our system, many of the documents returned as a result of queries are not relevant to the original citation record. For example, a query formed by a combination of an author name along with the keyword "resume" may return several resumes of people with similar names, who are different than the paper author. Hence, even though these documents are relevant to an otherwise useful query, they are irrelevant to the original citation. Another example of irrelevant document is when the returned document does not contain any year information. The document filter recognizes these cases by soft matching the title with body of the document to decide if this document is relevant to the original citation. The document filter also removes web pages without any year on the page.

3.5 Information Extraction

The design of this module differs from traditional information extraction in the following ways. Features from the original database are merged with the features obtained from the external source. Depending on the application, different integration schemes are possible. The training algorithm updates its own parameters in an incremental fashion, as new documents arrive. Similarly, the confidence in the prediction made by the system must be updated efficiently as new information arrives. Hence, the design of this module poses many interesting challenges.

3.5.1 Probabilistic Prediction Model

In our task, the field with missing values can take one of a finite number of possible values (i.e. a finite number of years). Hence, we can view this extraction task as a multi-class classification problem. We assume that we are given a range of years of publication and we classify each citation as belonging to one of the possible classes. Features from both the original citation as well as the documents obtained from the *Document Classifier* are combined to make the prediction using a maximum entropy classifer.

We use c_i to denote a citation ($i = 1, \ldots, n$), q_{ij} to denote a query formed using input from citation c_i and d_{ijk} to denote a document obtained as a result of q_{ij}. Assuming that we exhaustively use all the queries, and the documents pass through the document filter, we drop the index j and use d_{ik} to denote a document relevant to citation c_i. Let y_i represent a random variable that assigns a label to the citation c_i. We also define a variable y_{ik} to assign a label to the document d_{ik}. Note that, if Y is the set of all years in the given range, then $y_i, y_{ik} \in Y$. For each c_i, we define a set of m feature functions $f_m(c_i, y_i)$. For each d_{ik}, we define a set of l feature functions $f_{lk}(c_i, d_{ik}, y_{ik})$ on the document. For our model, we assume that $f_m(c_i, y_i)$ is empty. This is because the information from the citation by itself is not useful in predicting the year of publication. In the future, we would like to design a more general model that takes these features into account. We can now construct a model given by

$$P(y_{ik}|c_i, d_{ik}) = \frac{1}{Z_d} \exp(\lambda_l f_l(c_i, d_{ik}, y_{ik})), \qquad (1)$$

where $Z_d = \sum_y \exp(\lambda_l f_l(c_i, d_{ik}, y_{ik}))$

3.5.2 Combining Evidence in Feature Space v.s. Output Space

The above model outputs y_{ik} instead of the required y_i. We have two options to model what we want. We can either merge all the features $f_{lk}(c_i, d_{ik}, y_{ik})$ from d_{ik}'s to form a single feature function. This is equivalent to combining all the evidence for a single citation in the feature space. Alternatively, we can combine the evidence from different d_{ik}'s in the output space. Following are two possible schemes for combining the evidence in the output space. In the first scheme, we take a *majority vote*, i.e., the class with the highest number of y_{ik} is predicted as the winning class and assigned to y_i. In the second scheme, which we call as *highest confidence* scheme, we take the most confident vote, i.e.,

$$y_i = argmax_{y_{ik}} P(y_{ik}|c_i, d_{ik}) \qquad (2)$$

3.6 Uncertainty Propagation in Citation Graph

The inherent dependency within the given data set can be exploited for better utilization of resources. In our case, we have the citation link structure, which can be used for inferring temporal constraints. For example, if paper A cites paper B, then assuming that papers from future can not be cited, we can infer the constraint that paper B must have been published in the same or earlier year than paper A.

Initially, we have no information about the year of publication for any citation. As information from the web arrives in the form of documents, this uncertainty is reduced. If we propagate this reduction in uncertainty (or belief) for one of the nodes through the entire graph, we may need fewer documents (or fewer queries) to predict the year of publication for the remaining nodes. Furthermore, if we select the citations to query in an effective order, we may further

improve our predictions.

This is a very interesting aspect of the problem and can lead to many different solutions. Here we describe a few different approaches. In the future, we would like to experiment with a more formal, belief-propagation like method.

3.6.1 Notation

To explain the methods for uncertainty propagation, we employ the following notation. Let $c \in C$ be the citation which is currently being queried. Let $a \rightarrow b$ denote that citation a cites citation b. Let $C_B = \{c_b | c_b \rightarrow c\}$ and $C_A = \{c_a | c \rightarrow a\}$. Let X be the random variable that represents year of publication of c; $P_c(X = x)$ be the probability that it takes one of finite values in the given range, and $P'(X = x)$ be the posterior probability obtained from the *Document Classifier*.

3.6.2 Propagation Methods

The method *Best Index* passes the uncertainty message to the neighbors of c as follows:

$$\forall c_b \in C_B P_{c_b}(X = x) = P(X = x | x \geq y) \tag{3}$$

$$\forall c_a \in C_A P_{c_a}(X = x) = P(X = x | x < y) \tag{4}$$

Where $y = argmax_y P'_c(X = y)$. $P(X = x|x \geq y)$ and $P(X = x|x < y)$ are given by one of the update methods described below.

The method *Weighted Average* takes a weighted average over all possible $y's$:

$$\forall c_b \in C_B P_{c_b}(X = x) = P'_c(X = y) \sum_y P(X = x | x \geq y) \tag{5}$$

$$\forall c_a \in C_A P_{c_a}(X = x) = P'_c(X = y) \sum_y P(X = x | x < y) \tag{6}$$

3.6.3 Update Methods

The basic idea behind these update methods is as follows. If we know that the given paper was published after a certain year, then we can set the probability mass from before the corresponding index to zero and redistribute it to the years after the index. We only show update in one direction here for brevity.

The first update method, called the *Uniform Update*, simply redistributes the probability mass, $P(x \geq y)$ uniformly to the remaining years.

$$P(X = x | x \geq y) = 0, x < y \tag{7}$$

$$= P(X = x) + \frac{1}{P(x \geq y)}, x \geq y \tag{8}$$

The second update method, called the *Scale Update*, uses conditional probability:

$$P(X = x | x \geq y) = 0, x < y \tag{9}$$

$$= \frac{P(X=x)}{P(x \geq y)}, x \geq y \tag{10}$$

3.6.4 Combination Methods

Along with passing message to its neighbors, the node updates itself by combining the information from the *Document Classifier* and the graph structure. The following options can be used. The *Basic*,

$$P_c(X = x) = P(X = x | x = y) \tag{11}$$

$$P_c(X = x) = P'_c(X = y) \sum_y P(X = x | x = y) \tag{12}$$

The other two options are *Product* $P_c(X = x) = P_c(X = x) * P'_c(X = x)$ and $SumP_c(X = x) = P_c(X = x) + P'_c(X = x) - P_c(X = x) * P'_c(X = x)$

3.7 Confidence Evaluation System

Following are the important functions of Confidence Evaluation System.

In the case of training under resource constraints, after adding each new training document, it can measure the 'goodness' of the model by evaluating it on a validation set. At test time, as more information is obtained from the external source, confidence in the prediction improves. It sets a threshold on the confidence in the prediction, to either return the required information to the database, or to request more information. It also makes the choice between obtaining a new document or to issue a new query at each iteration, by taking into account the cost and utility factors. Finally, it keeps track of the effectiveness of queries and documents in making a correct prediction. This information is useful for learning better ranking models for Query Engine and Document Filter.

In our system, we train our model in a batch manner, using all available resources. We focus on evaluating test time confidence. For merging evidence in the output space, we employ two schemes to make this decision. In the first scheme, which we call *max votes*, if the percentage of documents in the winning class crosses a given threshold, then we make a prediction. In the second scheme, which we call *highest confidence*, if $P(y_{ik}|c_i, d_{ik})$ value of the document with the highest P in the winning class passes a threshold, we make a prediction. These two schemes provide useful methods for determining if we have completed the task satisfactorily. For combining evidence in feature space, we use the *Entropy Method*, in which we compute the value $H = -\sum_i p_i \log p_i$ of the current distribution, and compare it against the confidence threshold.

4. EXPERIMENTAL DESCRIPTION AND RESULTS

4.1 Dataset and Setup

Our data set consists of 462 citations from the Rexa corpus, with years of publication ranging from 1989 to 2008. In order to capture the citation links between these citations, we use Algorithm 1 to sample five citation graphs. This method ensures that we sample graphs of reasonable size and dense enough citation structure to work with. We use five-fold cross validation on these data sets for all our experiments.

We use the Mallet [14] infrastructure for training and testing, and the Google search API to issue queries. The

Algorithm 1 Dataset Sampling Method

1: Start with all the citations, and create an inverse citation index
2: Keep all papers that have at least 6 papers that it cites or 3 papers that cite it.
3: Keep all papers that fall within the given 20-year range ('89 to '08)
4: **repeat**
5: Randomly pick a seed citation and add it to the queue
6: **for all** Elements in the queue **do**
7: If the citation is found in the filtered dataset, add it to the graph.
8: Add all the papers it cites and all the papers that cite it.
9: If the number of citations added to the graph exceeds a cutoff (=100), stop.
10: **end for**
11: If the resulting graph is smaller than another cutoff (=20), discard it.
12: **until** Required number (=5) of graphs are generated:

title
title in quotes
title in quotes + "publication list"
title in quotes + "resume"
title in quotes + "cv"
title in quotes + "year"
title in quotes + "year of publication"
author + "publication list"
author + "resume"
author + "cv"
author + title

Figure 2: Types of Queries

queries formed using the information from input citations include the raw title, title in quotes, and author names combined with keywords like "publication list", "resume", "cv" , "year" and "year of publication". Fig. 2 shows a list of queries issued. We issue all the queries in a random order. In these experiments, we obtain the top 10 hits cached by google. In our dataset, we use 6936 queries and obtain 14999 documents after removing unrelated documents. The documents are tokenized and tokens are tagged to be possible years using a regular expression. The document filter uses two different criteria to discard a document. If there is no year information found on the web page at all, the document is discarded. The second criteria uses a soft match between the title in the citation and all n-grams of length equal to the title in the body of the page. If there is at least one n-gram with more than 75% overlap with title tokens, then the document is retained.

The documents that pass these two tests are passed on to the MaxEnt classification model in a random order. Following features are used for classification. Distances and counts of multiple occurrences of the same feature are discretized.

Year On Page True for all the occurrences of years on the webpage.

Number of Different Years The total number of occurrences of different years on the webpage.

Years In Order True if the years found on the webpage are in any of the following three particular orders: descending, ascending, or uniform.

Following / Preceding Year These features correspond to the years that immediately follow or precede the title matches found on the webpage.

Following / Preceding Year And Distance These two features record the distance (number of tokens) between the following (or preceding) year and its corresponding title match.

Between The Same Year True if a title match has the same following and preceding year.

Entropy Threshold	Precision	Recall	F1	#Queries	#Docs
0.1	0.9357	0.7358	0.8204	4497	9564
0.3	0.9183	0.8220	0.8666	3752	8010
0.5	0.9013	0.8718	0.8854	3309	7158
0.7	0.8809	0.9041	0.8909	2987	6535
0.9	0.8625	0.9171	0.8871	2768	6088

Table 1: Baseline results

4.2 Results and Discussion

We first run our RBIE system without exploiting the citation network information. We first present the results for combining evidence in the feature space. We measure Precision, Recall and F1 based on using a confidence threshold, where F1 is defined as follows.

$$F1 = 2.\frac{precision.recall}{precision + recall} \tag{13}$$

As seen in table 2, as we increase the entropy threshold, precision drops, as expected. F1 peaks at threshold 0.7. Note that the number of documents is proportional to the number of queries, because in our experiments, we stop obtaining more documents or queries when the threshold is reached.

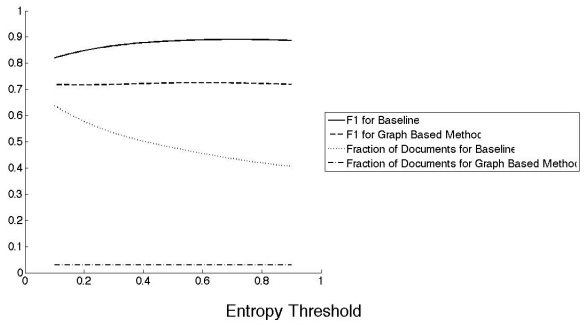
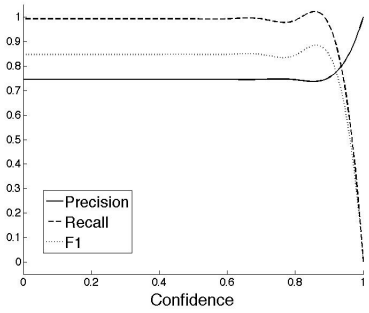
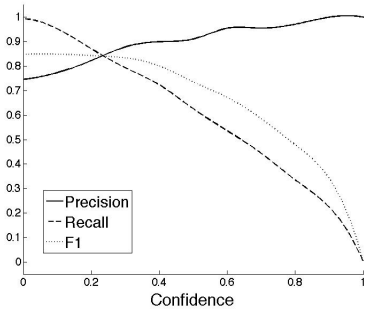


Figure 3: The change in F1 v.s. the change in use of resources

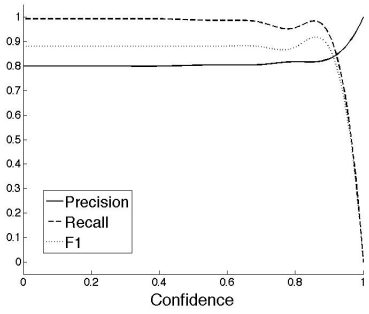
Next, we present the results of exploiting citation network information for better resource utilization. Fig. 3 shows F1 as well as the fraction of the total documents used for the



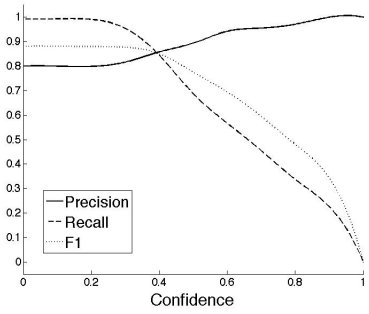
(a) Highest Confidence Vote, Highest Confidence



(b) Highest Confidence Vote, Max Votes Confidence



(c) Majority Vote, Highest Confidence



(d) Majority Vote, Max Votes Confidence

Figure 4: Precision, Recall and F1 values for different combinations of voting and confidence evaluation schemes.

Propagation	Update	Combination	F1
Best Index	Uniform	Basic	0.7192
Best Index	Uniform	Sum	0.7273
Best Index	Uniform	Product	0.6475
Best Index	Scaling	Basic	0.7249
Best Index	Scaling	Sum	0.6875
Best Index	Scaling	Product	0.6295
Weighted Avg	Uniform	Basic	0.7249
Weighted Avg	Uniform	Sum	0.5827
Weighted Avg	Uniform	Product	0.3460
Weighted Avg	Scaling	Basic	0.7249
Weighted Avg	Scaling	Sum	0.5365
Weighted Avg	Scaling	Product	0.4306

Table 2: Comparison of Uncertainty Propagation Methods

baseline method, and for one of the graph based method (*Weighted Average* propagation, *Scaling* update, and *Basic* combination). The F1 values are smaller compared to the baseline because we use far fewer resources, and the uncertainty propagation methods are not perfect. Using this method, we are able to achieve 87.7% of the baseline F1, by using only 13.2% of the documents. This demonstrates the effectiveness of exploiting the relational nature of the data. Table 4 shows the results of different uncertainty propagation methods at entropy threshold 0.7.

We also experiment with combining evidence in the output space using the two schemes described in section 3.5, and the confidence evaluation schemes described in section 3.7. Fig. 4 shows the four precision-recall curves. We see that for *High Confidence Confidence* evaluation scheme (fig. 4(a),(c)), we obtain high values of precision and recall for reasonable values of confidence. That is, in the confidence region below 0.9, we obtain a good F1 value. Especially, the *Majority Vote - High Confidence* scheme (fig. 4(c)) performs exceptionally well in making predictions. However, in the confidence region between 0.9 to 1.0, the *Max Vote* scheme (fig. 4(b),(d)) gives a better degradation performance.

5. CONCLUSION AND FUTURE WORK

We propose a new framework for targeted information extraction under resource constraints to fill missing values in a database. We present first results on an example task of extracting missing years of publication of scientific paper, along with exploiting the underlying citation network for better resource utilization. The overall framework is flexible, and can be applied to a variety of problem domains and individual system components can be adapted to the task. The specific methods recommended here can be also be generalized in many different relational domains, especially when the dataset has an underlying network structure. In future, we would like to explore more sophisticated uncertainty propagation methods, such as belief-propagation. We would also like to develop individual components like *Query Engine* and *Document Filter*, by using good ranking procedures. Finally, it would be interesting to see how these methods extend to extracting multiple interdependent fields.

6. ACKNOWLEDGMENTS

We thank Google for providing a special access to their search interface.

7. REFERENCES

[1] M. Bilgic and L. Getoor. Voila: Efficient feature-value acquisition for classification. In *AAAI*, pages 1225–1230. AAAI Press, 2007.

[2] M. Bilgic and L. Getoor. Effective label acquisition for collective classification. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 43–51, 2008. Winner of the KDD'08 Best Student Paper Award.

[3] D. Cohn, L. Atlas, and R. Ladner. Improving generalization with active learning. *ML*, 15(2):201–221, 1994.

[4] C. Daskalakis, R. M. Karp, E. Mossel, S. Riesenfeld, and E. Verbin. Sorting and selection in posets. *CoRR*, abs/0707.1532, 2007.

[5] O. Etzioni, M. Cafarella, D. Downey, S. Kok, A. Popescu, T. Shaked, S. Soderland, D. Weld, and A. Yates. Web-scale information extraction in knowitall. In *Proceedings of the International WWW Conference, New York*. ACM, May 2004.

[6] R. Grishman and B. Sundheim. Message understanding conference-6: a brief history. In *Proceedings of the 16th conference on Computational linguistics*, pages 466–471, Morristown, NJ, USA, 1996. Association for Computational Linguistics.

[7] P. Kanani and A. McCallum. Resource-bounded information gathering for correlation clustering. In *Computational Learning Theory 07, Open Problems Track, COLT 2007*, pages 625–627, 2007.

[8] P. Kanani, A. McCallum, and C. Pal. Improving author coreference by resource-bounded information gathering from the web. In *Proceedings of IJCAI*, 2007.

[9] P. Kanani and P. Melville. Prediction-time active feature-value acquisition for customer targeting. In *Proceedings of the Workshop on Cost Sensitive Learning, NIPS 2008*, 2008.

[10] A. Krause and C. Guestrin. Near-optimal nonmyopic value of information in graphical models. In *Twenty-first Conference on Uncertainty in Artificial Intelligence (UAI*, page 05, 2005.

[11] J. Lin, A. Fernandes, B. Katz, G. Marton, and S. Tellex. Extracting answers from the web using knowledge annotation and knowledge mining techniques, 2002.

[12] D. Lizotte, O. Madani, and R. Greiner. Budgeted learning of naive-Bayes classifiers. In *UAI03*, Acapulco, Mexico, 2003.

[13] G. Malewicz. Parallel scheduling of complex dags under uncertainty. In *SPAA*, pages 66–75, 2005.

[14] A. K. McCallum. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu, 2002.

[15] P. Melville, M. Saar-Tsechansky, F. Provost, and R. Mooney. An expected utility approach to active feature-value acquisition. In *Proceedings of the International Conference on Data Mining*, pages 745–748, Houston, TX, November 2005.

[16] M. H. Nodine, J. Fowler, T. Ksiezyk, B. Perry, M. Taylor, and A. Unruh. Active information gathering in infosleuth. *International Journal of Cooperative Information Systems*, 9(1-2):3–28, 2000.

[17] M. J. Rattigan, M. Maier, and D. Jensen. Exploiting network structure for active inference in collective classification. *Data Mining Workshops, International Conference on*, 0:429–434, 2007.

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

[19] V. S. Sheng and C. X. Ling. Feature value acquisition in testing: a sequential batch test algorithm. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 809–816, New York, NY, USA, 2006. ACM.

[20] F. Wu, R. Hoffmann, and D. S. Weld. Information extraction from wikipedia: moving down the long tail. In *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 731–739, New York, NY, USA, 2008. ACM.

[21] S. Zhao and J. Betz. Corroborate and learn facts from the web. In *KDD*, pages 995–1003, 2007.