ARISE-PIE: A People Information Integration Engine over the Web

Vincent W. Zheng[1], Tao Hoang[2], Penghe Chen[1], Yuan Fang[3], Xiaoyan Yang[1],
Kevin Chen-Chuan Chang[4]
[1]Advanced Digital Sciences Center, Singapore
[2]University of South Australia, Australia
[3]Institute for Infocomm Research, A*STAR, Singapore
[4]University of Illinois at Urbana-Champaign, USA
{vincent.zheng, penghe.c, xiaoyan.yang}@adsc.com.sg,
hoatn002@mymail.unisa.edu.au, fang2@i2r.a-star.edu.sg, kcchang@illinois.edu

ABSTRACT

Searching for people information on the Web is a common practice in life. However, it is time consuming to search for such information manually. In this paper, we aim to develop an automatic people information search system, named ARISE-PIE. To build such a system, we tackle two major technical challenges: data harvesting and data integration. For data harvesting, we study how to leverage search engine to help crawl the relevant Web pages for a target entity; then we propose a novel learning to query model that can automatically select a set of "best" queries to maximize collective utility (*e.g.*, precision or recall). For data integration, we study how to leverage flexible forms of constraints as weak supervision to achieve collective information extraction from a target entity's Web page corpus; then we propose a novel conditional probabilistic formulation to model constraints and an efficient realization to enable the inference with constraints. We evaluate our data harvesting and data integration solutions on the real-world data sets, and show that they both achieve better performance than the state-of-the-art baselines. We also evaluate our system on a benchmark data set and with a user study, in which we both show promising results.

CCS Concepts

•**Information systems** → **Web crawling; Data extraction and integration;** *Data mining;*

1. INTRODUCTION

Searching for people information from the Web is a common practice in our daily life. For example, Bing shows that people searches account for about 10 percent of all searches on Bing[1]. However, searching for people information is a time-consuming task, because: 1) the information of interest often scatters across many different pages on the Web, such that users have to find a good way to search and sift through many Web pages for the relevant information; 2) the information of interest is often unstructured, such that users have to spend enormous effort to read through the content of many Web pages. Being able to automatically search, extract and integrate the information of interest from the Web for a person query is very useful. In addition to supporting ad-hoc people search by the individual users, it can also enable many downstream applications for business; e.g., in the field of talent acquisition, some example applications are

- *Talent verification*. Given a talent candidate for hiring, we can first search, extract and integrate her information of interest, such as PUBLICATIONS, AWARDS and SOCIAL ACCOUNTS; then we can use the found information to complete and verify the curriculum vitae submitted by the talent candidate. In this way, we can minimize the talent hiring cost by screening the candidates.
- *Talent management*. Once we manage to compile a talent database with integrated information, we can then index and rank the talents according to their skills, as well as their levels of position matching. In this way, we can maximize the talent hiring success rate by finding the most relevant candidates.

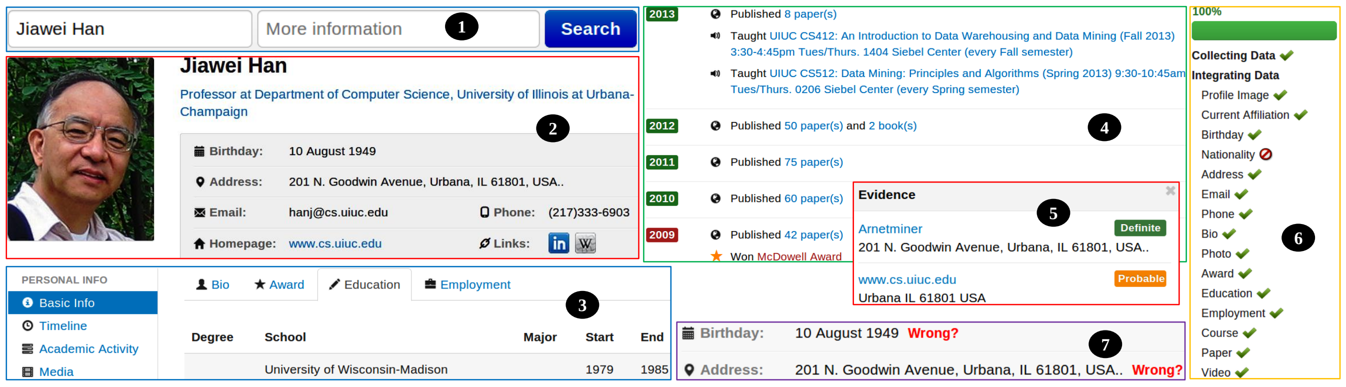
Technical challenges. As we discussed earlier, manual searching of people information mainly suffers from two disadvantages: 1) manual Web search is tedious and may be ineffective; 2) manual Web extraction and integration is difficult. As a result, to automate the people information search, extraction and integration process, we need to answer two fundamental questions:

- (Data Harvesting) *How to effectively search for Web pages w.r.t. an* entity *(e.g., "Jiawei Han" from UIUC) and an* aspect *(e.g.,* RESEARCH*)*?
- (Data Integration) *How to accurately extract the information about multiple aspects of an entity from various Web pages*?

For data harvesting, we study how to leverage search engine to help crawling the relevant information. Crawling Web pages with certain relevant information is generally known as *focused crawling* [5]. Traditional focused crawling relies on the hyperlinks on the Web, and largely overlooks the search engine which has indexed the Web by content keywords. We can use the search engine to fast pinpoint the relevant information on the Web. Ideally, we can combine search engine and hyperlink traversal to search for relevant Web pages, but in this study we only focus on using search engine to find the relevant Web pages, as it is hardly studied. Then the

[1]http://www.bing.com/blogs/site_blogs/b/search/archive/2013/03/21/satorii.aspx

CIKM '16 Workshop on Data-Driven Talent Acquisition
© 2016 ACM. ISBN 978-1-4503-4073-1/16/10…$15.00
DOI: 10.475/123_4



Jiawei Han | More information | Search

Jiawei Han

Professor at Department of Computer Science, University of Illinois at Urbana-Champaign

Birthday: 10 August 1949

Address: 201 N. Goodwin Avenue, Urbana, IL 61801, USA..

Email: hanj@cs.uiuc.edu Phone: (217)333-6903

Homepage: www.cs.uiuc.edu Links:

PERSONAL INFO: Basic Info, Timeline, Academic Activity, Media

Bio | Award | Education | Employment

Degree	School	Major	Start	End
	University of Wisconsin-Madison		1979	1985

2013 Published 8 paper(s)

Taught UIUC CS412: An Introduction to Data Warehousing and Data Mining (Fall 2013) 3:30-4:45pm Tues/Thurs. 1404 Siebel Center (every Fall semester)

Taught UIUC CS512: Data Mining: Principles and Algorithms (Spring 2013) 9:30-10:45am Tues/Thurs. 0206 Siebel Center (every Spring semester)

2012 Published 50 paper(s) and 2 book(s)

2011 Published 75 paper(s)

2010 Published 60 paper(s)

2009 Published 42 paper(s)

Won McDowell Award

Evidence

Arnetminer Definite
201 N. Goodwin Avenue, Urbana, IL 61801, USA..

www.cs.uiuc.edu Probable
Urbana IL 61801 USA

Birthday: 10 August 1949 Wrong?

Address: 201 N. Goodwin Avenue, Urbana, IL 61801, USA.. Wrong?

100%

Collecting Data ✓

Integrating Data

Profile Image ✓
Current Affiliation ✓
Birthday ✓
Nationality ⊘
Address ✓
Email ✓
Phone ✓
Bio ✓
Photo ✓
Award ✓
Education ✓
Employment ✓
Course ✓
Paper ✓
Video ✓

Figure 1: The user interface of ARISE-PIE, by components.

question becomes how to automatically find a set of "best" queries w.r.t. a budget (*e.g.*, number of search engine API calls).

For data integration, we study how to leverage flexible forms of constraints to help extract information for multiple aspects of an entity in a Web page corpus. Extracting multiple fields of information from the text in a collective manner is generally known as *collective extraction* [3]. Traditional collective extraction models the interdependence among multiple extractions as "features" and relies on sufficient labeled data to train the weights for these features. In practice, labeled data is limited, thus a viable way is to model the interdependence as "constraints", and use them as "labeled features" to weakly supervise unlabeled data for semi-supervised learning. Then the question becomes how to formulate flexible forms of "constraints", and then efficiently realize them in inference with the unlabeled data.

Data havesting solution. For data harvesting, our insights are three-fold. First, to select the best query, we must quantify what is considered a good query, by estimating the *utility* of each candidate query. Since the ultimate purpose of a query is to retrieve relevant pages from the Web, the utility of a query should reflect how well it can accomplish this purpose, such as the precision and recall (or some combination of them) of the retrieved pages w.r.t. the target entity and aspect. The utility should be inferred *without* actually firing any candidate query. Second, an entity does not exist in isolation. There are often a large number of peer entities in the same *domain* (i.e., other researchers), which can reveal useful insights of the domain. Thus, it is necessary to be *domain aware*: leveraging the domain of an entity to bootstrap at the beginning when little about the target entity is known, as well as to enhance learning during the entire querying process. Third, a query does not exist in isolation. Multiple queries are needed to gather more target pages. That is, there exist a context of past queries that were already fired for the target entity. Given the time, bandwidth and sometimes financial costs to query through a commercial search engine, it is imperative to become *context aware*: accounting for the context of past queries to eliminate redundancy between queries.

To solve data harvesting, we propose a novel *Learning to Query* (L2Q) model [6], which is able to: 1) estimate a query's utility by probabilistic precision and recall; 2) leverage the domain awareness to adapt queries for different entities; 3) leverage the context awareness to select queries for maximizing collective utility.

Data integration solution. For data integration, our insights are two-fold. First, constraints are often conditional (thus having flexible forms) and probabilistic. A constraint is commonly expressed as an *if-then* statement; e.g., if two text snippets are both BIOGRAPHY of a researcher, then they are similar. The *if*-part describes the condition. Some constraint's condition depends on only observation x's (*i.e.*, text content), thus we call it an x-type constraint. Other constraint's condition depends on hidden variable y's (*i.e.*, aspect assignments), thus we call it a y-type constraint (an example is the BIOGRAPHY constraint). Besides, a constraint is probabilistic; e.g., the BIOGRAPHY of a researcher may vary in different Websites. Second, constraints can be used as weak supervision on the unlabeled data for semi-supervised learning [4, 8, 9], but y-type constraints often make this learning difficult due to its inference complication. E.g., a brute-force evaluation of the BIOGRAPHY constraint checks the aspect assignments of every two text snippets. This creates a complete graph over the unlabeled text snippets, which is hard for inference. But since the constraint is conditional, we only care about those text snippets that are truly "relevant"; if we can guess which snippets are BIOGRAPHY, then we can save a lot of effort by selectively evaluating them.

To solve data integration, we propose a novel *Conditional Probabilistic Formulation* (CPF) [15], which is able to: 1) model flexible forms of constraints with explicit notions of constraint condition and probability; 2) use constraints to weakly supervise the unlabeled data for building a "general" (instead of logic-based [10]) semi-supervised extractor; 3) achieve efficient inference by selectively evaluating the relevant instances from the corpus.

ARISE-PIE system. Based on our innovation on data harvesting and data integration technologies, we develop a system named ARISE[2] People information Integration Engine (ARISE-PIE)[3], particularly for the researcher domain. As shown in Fig. 1, our system takes a person entity query (*e.g.*, person name "Jiawei Han" and some optional information "University of Illinois") as input. It outputs the integrated information according to some predefined *aspects*, such as CONTACT, PUBLICATIONS and so on. To harvest the information about the queried entity, we leverage search engine (*e.g.*, Google) and use learning to query to iteratively construct a set of queries to find the relevant Web pages for each aspect. As the collected Web pages are generally unstructured, we then try to extract and integrate the aspect information from the Web pages for the queried entity. Specifically, we use the conditional probabilistic formulation to model the inter-dependencies among multiple extractions within the queried entity's Web page corpus as constraints, and then do semi-supervised collective extraction. In addition to the unstructured Web pages, we also leverage structured data

[2]It stands for "Augmented Reality Information Search Engine".
[3]System demo is available at https://vimeo.com/82167291.

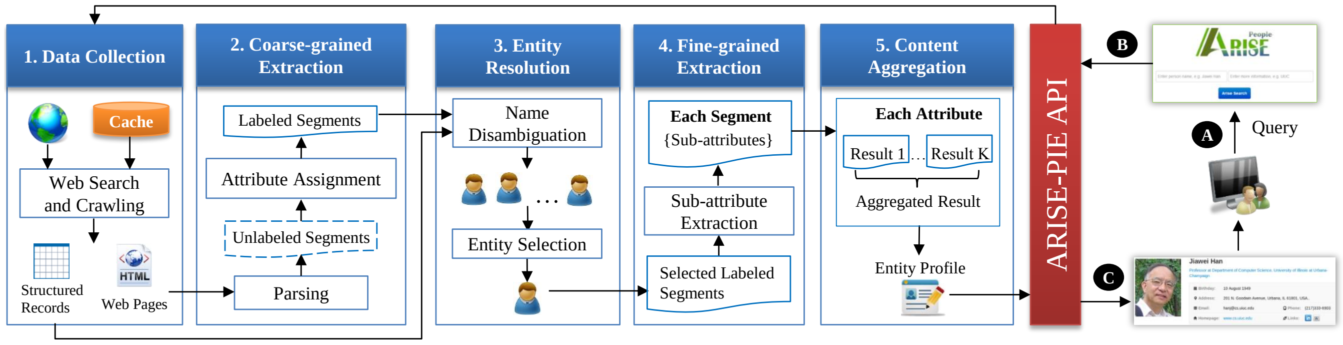


Figure 2: The system architecture of ARISE-PIE.

sources, such as different people listing services (including DBLP, Freebase, LinkedIn, etc.), to extract the entity information.

Our ARISE-PIE system is novel in terms of being able to: 1) automatically harvest Web pages w.r.t. a queried entity and a queried aspect through search engines, thus it can find more information from more diverse sources; 2) collectively extract and integrate the information from the unstructured Web page corpus in a more effective (by enforcing collective extraction through flexible forms of constraints) and more practical (by leveraging unlabeled data for semi-supervised learning and selective evaluation for efficient inference) manner. Comparatively, some popular academic search systems, such as ArnetMiner (aminer.org), DBLife (dblife.cs.wisc.edu) and Microsoft Academic Search (academic.research.microsoft.com), tend to extract information from limited information sources (*e.g.*, publisher databases, researcher homepages) instead of the general Web. Some other commercial systems, such as Intelius (intelius.com), Spokeo (spokeo.com) and ZoomInfo (zoominfo.com) also tend to largely rely on the offline census records and extract information from limited online sources.

2. OVERVIEW OF OUR SYSTEM

In this section, we introduce the system functionalities and the architecture design of our ARISE-PIE system.

2.1 System Functionality

Users can freely access our system to search for researchers of interest, and our system can automatically integrate the found Web information for the queried researchers on the fly. As a running example, suppose we aim to search for the information about Jiawei Han from University of Illinois. Then, we can enter "Jiawei Han" in the first query box and optionally "University of Illinois" in the second query box, as shown in Box 1 of Figure 1. Once receiving the query, our system starts to harvest the Web data. Then, it automatically extracts and integrates the information. Finally, it returns the results in an entity profile page, which consists of several components (highlighted as boxes in Figure 1) as follows.

Profile Snapshot (Box 2). It presents the portrait and some short-content aspects such as current EMPLOYMENT and CONTACT. This gives a brief overview of the queried entity.

Profile Details (Box 3). It gives more details EMPLOYMENT, EDUCATION, AWARDS and academic activities such as TEACHING and PUBLICATIONS. We organize the information in tables.

Event Timeline (Box 4). It organizes the time-sensitive events such as graduation, employment change, award winning, publication and so on. We organize them in a reversed chronological order.

Information Source (Box 5). It lists the sources where we extracted information, thus allowing users to track back. We organize the sources in a descending order of the extraction confidences.

Progress Tracker (Box 6). It keeps track of the data harvesting and integration progress. The aspects that have information extracted will appear with ticks in the tracker.

Error Reporter (Box 7). It allows users to report errors of the integration results by simply clicking a button.

2.2 Architecture Design

We illustrate the architecture of ARISE-PIE in Figure 2. The system begins with the user's query input to the Web-based interface in step *A*. The query is then forwarded to the system API to process in step *B*. After the processing is completed, the system API returns the results in step *C*. The core of our system is in step *B*. It follows a general workflow of data harvesting and integration in Figure 2:

- *Data Collection* (Box 1): harvest Web data for the query;
- *Information Extraction* (Box 2, Box 4): extract information from the collected data. As will be discussed soon, we design a two-level extraction framework with *coarse-grained extraction* (Box 2) and *fine-grained extraction* (Box 4);
- *Information Aggregation* (Box 3, Box 5): aggregate the extracted information by 1) *Entity Resolution* (Box 3) to disambiguate the information to the right entities; 2) *Content Aggregation* (Box 5) to merge multiple pieces of content.

A Two-level Extraction Framework. For efficiency consideration, we decompose the extraction task into coarse-grained extraction and fine-grained extraction. For coarse-grained extraction, the idea is to fast locate the relevant information in the page. It is done by parsing a page into a set of text snippets, and then assigning a aspect label to each snippet. For fine-grained extraction, the idea is to further extract the "sub-aspect" (i.e., different data fields of an aspect) from the text snippets for display. For example, we extract the job title and start-end dates for EMPLOYMENT in Box 5 of Figure 1. This two-level extraction avoids trying to extract fine-grained aspect information from irrelevant snippets.

Interleaving between Extraction and Aggregation. In general, we have the option to do aggregation after all the extractions are done, but in this system we choose to interleave between them. Specifically, we do entity resolution right after the coarse-grained extraction and before the fine-grained extraction. This is because in our system, we try to return one single entity who is most relevant to the query and also possesses the most information on the Web. This is similar to Google's "*I'm feeling lucky*" function, which tries to quickly navigate the user to the results. If the results happen to mismatch the user's intention, then the user needs to improve

(a) Example pages (Y is RESEARCH)

	Content	$Y(p_i)$
p_1	He conducts research on data mining & db.	1
p_2	He writes papers in data mining & db research.	1
p_3	His studies data mining & info network.	1
p_4	He also studies info network at U Illinois.	1
p_5	Visit him at Siebel Center, U Illinois.	0
p_6	He was a professor in Simon Fraser U.	0

(b) Example queries

Query	Retrievable pages
q_1: `data mining research`	$p_1 : 1$, $p_2 : 1$, $p_3 : 1$
q_2: `db research`	$p_1 : 1$, $p_2 : 1$
q_3: `info network`	$p_3 : 1$, $p_4 : 1$
q_4: `u illinois`	$p_4 : 1$, $p_5 : 1$, $p_6 : 0$
q_5: `simon fraser u`	$p_6 : 1$

(c) Reinforcement graph

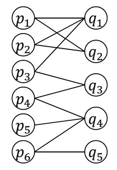


Figure 3: Running illustration for "Jiawei Han".

the query with more specific information for another search. As a result, we do not bother doing fine-extraction on the other entities' results, and thus we do entity resolution in advance.

3. DATA HARVESTING BY L2Q

We aim to build a model that can automatically discover and iteratively select a set of queries, such that we can fire these queries in a search engine to find as many relevant Web pages as possible w.r.t. an entity $e \in \mathcal{E}$ and an aspect $Y \in \mathcal{Y}$. As a result, we have to estimate the utility of each single query, based on how many relevant Web pages it can retrieve and what are the previous queries.

3.1 Utility Estimation

To estimate the utility of a single query about how many relevant Web pages it can retrieve, we hinge upon the intuition of mutual reinforcement between pages and queries. In general, a "useful" page p for the target aspect Y contains useful queries for Y, and a useful query q can retrieve useful pages for Y. Denote $\mathcal{U}(*)$ as a utility function. Then, for a page p and a query q, high $\mathcal{U}(p)$ implies high $\mathcal{U}(q)$ if p contains q, and vice versa.

Given the ultimate goal to harvest Web pages, we measure two complementary forms of utility: precision and recall of the retrieved pages w.r.t. Y. Denote a universe of pages as P. These pages correspond to a universe of candidate queries Q (e.g., all the n-grams in P). Let Ω denote a mapping from each of the following notions to a set of pages in the domain, *i.e.*, $\Omega(*) \subseteq P$. Specifically, we have $\Omega(Y)$ as the pages relevant to the target aspect Y; $\Omega(q)$ are the pages that can be retrieved by query q; $\Omega(p)$ is the page p itself, *i.e.*, $\Omega(p) = \{p\}$. Consider a running example in Fig. 3(a)–(b). For Y as RESEARCH, suppose there are six pages $p_1, \dots, p_6$ in total. $p_1, \dots, p_4$ are relevant, *i.e.*, $\Omega(Y) = \{p_1, \dots, p_4\}$. Besides, as an example, query q_2 retrieves $\Omega(q_2) = \{p_1, p_2\}$. Given $\Omega(v)$, $\forall v \in P \cup Q$, we can compute v's precision $\mathcal{P}(v)$ or recall $\mathcal{R}(v)$ w.r.t. Y by probability:

$$\mathcal{P}(v) \triangleq P(\omega \in \Omega(Y) | \omega \in \Omega(v)), \quad (1)$$

$$\mathcal{R}(v) \triangleq P(\omega \in \Omega(v) | \omega \in \Omega(Y)), \quad (2)$$

where ω is a random page from $\Omega(*)$.

We can model the mutual reinforcement between pages and queries by a graph $G = (V, E)$, as shown in Fig. 3(c). The vertex set $V = P \cup Q$, and the edge set E is described by an adjacency matrix W, such that $W_{pq} = W_{qp} = 1$ if and only if page p can be retrieved by query q, and $W_{pq} = W_{qp} = 0$ otherwise. A useful page (say p_1) can induce useful queries (q_1 and q_2 which are neighbors of p_1), and a useful query (*e.g.*, q_2) can retrieve useful pages (p_1 and p_2 which are neighbors of q_2). More quantitatively, $\mathcal{U}(q)$ can be expressed in terms of $\mathcal{U}(p)$, where p is a neighboring page of q, and vice versa. Denote the neighbor set of v on the graph by $N(v)$, *e.g.*, $N(p_1) = \{q_1, q_2\}$. Then, after some derivation (details are in [6]), we can show that: 1) the precision of a query q is the average precision of the pages that q can retrieve, weighted by q's probability of retrieving each page, *i.e.*,

(a) Example templates

Query	Template
q_1: `data mining research`	t_1: ⟨topic⟩ `research`
q_2: `db research`	t_1: ⟨topic⟩ `research`
q_3: `info network`	t_2: ⟨topic⟩
q_4: `u illinois`	t_3: ⟨institute⟩
q_5: `simon fraser u`	t_3: ⟨institute⟩

(b) Reinforcement graph

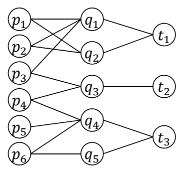


Figure 4: Running illustration extended with templates.

$$\mathcal{P}(q) = \textstyle\sum_{p \in N(q)} \frac{W_{pq}}{\sum_{p' \in N(q)} W_{p'q}} \mathcal{P}(p); \quad (3)$$

2) the recall of a query q is the sum of weighted recalls of the pages that q can retrieve, such that each page only contributes a part of its recall according to its probability of being retrieved by q, *i.e.*,

$$\mathcal{R}(q) = \textstyle\sum_{p \in N(q)} \frac{W_{pq}}{\sum_{q' \in N(p)} W_{pq'}} \mathcal{R}(p). \quad (4)$$

Similarly, we can express $\mathcal{P}(p)$ and $\mathcal{R}(p)$ in terms of $\mathcal{P}(q)$ and $\mathcal{R}(q)$ respectively. In fact, we can unify the reinforcement between queries and pages as: $\forall v \in V$,

$$\mathcal{U}(v) = F(\{\mathcal{U}(v') | v' \in N(v)\}), \quad (5)$$

where F is an aggregation function over the neighbors' utilities $\{\mathcal{U}(v') | v' \in N(v)\}$, which can be either precision or recall.

In training, we have some labeled Web pages, for which we know their empirical precision and recall. Denote $\widehat{\mathcal{P}}(p) = Y(p)$ and $\widehat{\mathcal{R}}(p) = Y(p) / \sum_{p' \in P} Y(p')$ if a page p is labeled. For a page v that is unlabled, we set $\widehat{\mathcal{P}}(v) = \widehat{\mathcal{R}}(v) = 0$. Then, we propagate the known utilities $\widehat{\mathcal{P}}(p)$ or $\widehat{\mathcal{R}}(p)$ to other unknown queries and pages:

$$\mathcal{U}(v) = (1 - \alpha) F(\{\mathcal{U}(v') | v' \in N(v)\}) + \alpha \, \widehat{\mathcal{U}}(v), \quad (6)$$

where $\alpha \in (0, 1)$ is the regularization parameter, and $\widehat{\mathcal{U}}(v)$ is the utility regularization for v representing either $\widehat{\mathcal{P}}(v)$ or $\widehat{\mathcal{R}}(v)$.

3.2 Domain-awareness

Different entities often require different queries. For example, `data mining` is a useful query for Jiawei, but not for Marc Snir, who is a professor working on `parallel computing`. In training, we cannot see all the possible entities, thus directly learning useful queries from the training entities is not enough. To address such entity variations, we observe that queries for different entities often match similar abstractions. For example, we can see both `data mining` and `parallel computing` represent research topics. We name such query abstractions *templates*. A template is a sequence of units $t = (u_1, u_2, \dots, u_\ell)$, where each unit u_i is either a word $w \in \mathcal{W}$ or a type $c \in \mathcal{C}$. A type c can be regular expressions or some predefined categories from knowledge bases. Consequently, we can learn the usefulness of templates, and apply it to the new entities in the same domain.

We estimate the utilities of templates through the reinforcement between templates and queries. Consider a template universe T, which can be enumerated from queries with a given set of types.

Based on the running example in Fig. 3, we obtain $T = \{t_1, t_2, t_3\}$ in Fig. 4(a). Furthermore, we can extend the reinforcement graph $G = (V, E)$ with templates, such that $V = P \cup Q \cup T$ and E now also captures the reinforcement between Q and T, as shown in Fig. 4(b). Denote $\Omega(t)$ as the set of pages that can be indirectly "retrieved" by template t through any of its abstracted queries. For instance, through q_1 and q_2, t_1 can retrieve $\Omega(t_1) = \{p_1, p_2, p_3\}$. Then, we can estimate the utilities of t, $\mathcal{P}(t)$ and $\mathcal{R}(t)$ based on the mutual reinforcement between Q and T. As each query is now connected to both pages and templates, we denote its page neighbors by $NP(q)$ and template neighbors by $NT(q)$. Finally, we have

$$\mathcal{P}(t) = \sum_{q \in N(t)} \frac{W_{qt}}{\sum_{q' \in N(t)} W_{q't}} \mathcal{P}(q) \quad (7)$$

$$\mathcal{R}(t) = \sum_{q \in N(t)} \frac{W_{qt}}{\sum_{t' \in NT(q)} W_{qt'}} \mathcal{R}(q) \quad (8)$$

Similarly, we can express $\mathcal{P}(q)$ and $\mathcal{R}(q)$ in terms of $\mathcal{P}(t)$ and $\mathcal{R}(t)$ respectively. As $\mathcal{P}(q)$ and $\mathcal{R}(q)$ can be expressed in terms of $\mathcal{P}(p)$ and $\mathcal{R}(p)$, we can combine both expressions by taking the average of them as the final utility for q.

In training, now with templates we can construct a *domain graph* of pages-queries-templates, and estimate the utility of each template by propagating the known page utilities on the domain graph according to Eq.6. Later in testing we can propagate the template utilities back to the candidate queries.

3.3 Context-awareness

In testing, we look for a series of queries together to find as many relevant Web pages for an entity and an aspect, therefore we need to consider the redundancy among the queries. Denote the queries that were fired before iteration-i as $\Phi \triangleq \{q^{(0)}, q^{(1)}, \ldots, q^{(i-1)}\}$. Denote the Web pages collected in training as P_D, and the Web pages collected by Φ in testing as P_E. We can extract candidate queries from P_E and construct an *entity graph* of pages-queries-templates. Due to the query redundancy, we choose a query q^* from a candidate set Q_E to maximize a collective utility $\mathcal{U}_E(\Phi \cup \{q\})$ based on P_E and P_D:

$$q^* = \arg\max_{q \in Q_E} \mathcal{U}_E(\Phi \cup \{q\} | P_E, P_D). \quad (9)$$

Parallel to the probabilistic utilities of one query (Eq. 1–2), we define below the collective utilities of a set of queries probabilistically, where $\Omega(Q) \equiv \bigcup_{q \in Q} \Omega(q)$.

$$\mathcal{P}_E(\Phi \cup \{q\}) \triangleq P(\omega \in \Omega(Y) \mid \omega \in \Omega(\Phi \cup \{q\})), \quad (10)$$

$$\mathcal{R}_E(\Phi \cup \{q\}) \triangleq P(\omega \in \Omega(\Phi \cup \{q\}) \mid \omega \in \Omega(Y)). \quad (11)$$

Our intuition to estimate the collective recall is that, the pages retrieved by $\Phi \cup \{q\}$ equal the pages retrieved by Φ and the pages retrieved by q, but subtracting their overlap. In other words, after some derivations (details are in [6]), we should be able to derive

$$\mathcal{R}_E(\Phi \cup \{q\}) = \mathcal{R}_E(\Phi) + \mathcal{R}_E(q) - \Delta(\Phi, q), \quad (12)$$

where $\Delta(\Phi, q)$ is the recall overlap between q and Φ. We can also derive the estimation of $\Delta(\Phi, q)$ as

$$\Delta(\Phi, q) = \mathcal{R}_E^{(\widetilde{Y})}(q) \cdot \mathcal{R}_E(\Phi), \quad (13)$$

where $\mathcal{R}_E^{(\widetilde{Y})}(q)$ is the recall estimated on the entity graph, given the known page utilities $\widehat{\mathcal{R}}_E^{(\widetilde{Y})}(p) = \widetilde{Y}(p) / \sum_{p' \in P_E} \widetilde{Y}(p'), \forall p \in P_E$ with $\widetilde{Y}(p) = 1$ iff $Y(p) = 1$ and $p \in P_E$. Note that, $\mathcal{R}_E(\Phi)$ can be recursively computed by decomposing $\Phi = \{q^{(0)}, q^{(1)}, \ldots, q^{(i-1)}\}$ into $q^{(i-1)}$ and $q^{(i-1)}$'s context queries $q^{(0)}, \ldots, q^{(i-2)}$. Thus, we only need to determine the base case—$\mathcal{R}_E(q^{(0)})$, recall of the initial seed query $q^{(0)}$. As we have not gathered any page for the target entity in the beginning, there is no reliable way to estimate $\mathcal{R}_E(q^{(0)})$. Thus, we treat it as a parameter $r_0 \in (0, 1)$ to tune.

Table 1: Example constraints for the researcher domain.

ID	Constraint statement	Prob.
RC_1	If a snippet is EDUCATION, then it has no course/bio words	66%
RC_2	If two snippets are BIOGRAPHY, then they are similar	71%
RC_3	If two snippets are EMPLOYMENT and BIOGRAPHY, then they share organizations	61%

Our intuition to estimate the collective precision is as follows: to optimize collective precision, Φ and q should collectively retrieve as many relevant pages, but at the same time as few total pages (regardless of page relevance); thus we should be able to decompose collective precision into two components, one as the collective recall with regards to Y by Φ and q, and the other as the collective recall regardless of Y by Φ and q. Formally, after some derivations (details are in [6]), we derive

$$\mathcal{P}_E(\Phi \cup \{q\}) = \frac{\mathcal{R}_E(\Phi \cup \{q\})}{\mathcal{R}_E^{(Y^*)}(\Phi \cup \{q\})}, \quad (14)$$

where $\mathcal{R}_E(\Phi \cup \{q\})$ is the collective recall w.r.t. Y, while $\mathcal{R}_E^{(Y^*)}(\Phi \cup \{q\})$ is the collective recall estimated on the entity graph, given the known page utilities $\widehat{\mathcal{R}}_E^{(Y^*)}(p) = Y^*(p) / \sum_{p' \in P_E} Y^*(p'), \forall p \in P_E$ with $Y^*(p) = 1$ for any page p.

4. DATA INTEGRATION BY CPF

As our task is to extract information about an entity, we expect to see many different kinds of constraints within each aspect of the entity (e.g., constraints RC_1 and RC_2 in Table 1) and across different aspects of the entity (e.g., constraint RC_3). Being able to leverage such constraints is critical for ensuring the data integration accuracy. In this section, we summarize our CPF model to formulate constraints and further incorporate them as weak supervision with unlabeled data for semi-supervised learning.

4.1 Conditional Probabilistic Constraints

As can be seen from Table 1, the constraints are often conditional and probabilistic. Thus our first problem to tackle is to properly formulate the constraints, with explicit notions of constraint condition and constraint probability. Specifically, we define a constraint *condition function* g as a binary feature function $g : \mathcal{X}^{d_1} \times \mathcal{Y}^{d_2} \to \{0, 1\}$, where $d_1, d_2 \in \mathbb{N} \cup \{0\}$. g returns one if the constraint is applicable to an instance $\mathbf{x} \in \mathbb{R}^{d_1}$ and its labels $\mathbf{y} \in \mathbb{R}^{d_2}$, or zero otherwise. As a result, an x-type constraint is a constraint having its constraint condition g with $d_1 \in \mathbb{N}$, $d_2 = 0$; whereas a y-type constraint is a constraint having its condition g with $d_1 \in \mathbb{N} \cup \{0\}$, $d_2 \in \mathbb{N}$. To define the constraint probability, we first introduce a constraint *satisfiability function* f. f is a binary feature function $f : \mathcal{X}^d \times \mathcal{Y}^d \to \{0, 1\}$, $d \in \mathbb{N}$, which returns one if a constraint is satisfied on the instance $\mathbf{x}$ and its labels $\mathbf{y}$, or zero otherwise. Thus a constraint probability is the probability of a constraint being satisfied when its condition is true, i.e., $\Pr(f(\mathbf{x}, \mathbf{y}) = 1 | g(\mathbf{x}, \mathbf{y}) = 1)$. Finally, we propose a conditional probabilistic formulation (CPF) to unify x-type and y-type constraints.

DEFINITION 1 (CPF). *A constraint c is expressed as a triple $(g(\mathbf{x}, \mathbf{y}), f(\mathbf{x}, \mathbf{y}), \eta)$ with $P(f(\mathbf{x}, \mathbf{y}) = 1 | g(\mathbf{x}, \mathbf{y}) = 1) = \eta$, where $\eta \in [0, 1]$ is the empirical conditional probability of c.*

4.2 Weak Supervision with Constraints

We aim to train a model for collective extraction among multiple Web pages of a person entity w.r.t. multiple aspects. To formalize the problem, we often have a small set of labeled data $D_L = \{(X_L^{(i)}, Y_L^{(i)})|i = 1, ..., n_L\}$, where each $(X_L^{(i)}, Y_L^{(i)}) = \{(x_k^{(i)}, y_k^{(i)})|k = 1, ..., n_i\}$ is a Web page with multiple labeled text snippets. Each $x_k^{(i)} \in \mathcal{X}$ is a text snippet; $y_k^{(i)} \in \mathcal{Y}$ is the snippet's label. Besides, we can also have a set of unlabeled data $D_U = \{X_U^{(i)}|i = 1, ..., n_U\}$, with each $X_U^{(i)} = \{x_k^{(i)}|k = 1, ..., n_i\}$. Denote a set of CPF constraints as $C = \{c_1, ..., c_{n_C}\}$. In training, we aim to output a multi-class classifier $q(Y|X; C)$ trained from D_L, D_U and C, where X is a set of observations. Denote a set of test data as $D_T = \{(X_T^{(t)}, Y_T^{(t)})|i = 1, ..., n_T\}$, with each $(X_T^{(t)}, Y_T^{(t)}) = \{(x_k^{(t)}, y_k^{(t)})|k = 1, ..., n_t\}$. In testing, we classify each $X_T^{(t)} \in D_T$ by $Y \leftarrow \arg\max_{Y \in \mathcal{Y}^{n_t}} q(Y|X_T^{(t)}; C)$, and compare with the corresponding ground truth $Y_T^{(t)}$ for evaluation.

We use Conditional Random Fields (CRF) [7] to model the labeled data. We design a set of features $h_i(\mathbf{x}, \mathbf{y})$ to characterize the dependencies among $\mathbf{x}$ and $\mathbf{y}$. For notation simplicity, we denote $h_i(X, Y) = \sum_k h_i(\mathbf{x}_k, \mathbf{y}_k)$ as summing over all the possible $(\mathbf{x}_k, \mathbf{y}_k)$ from (X, Y). By stacking the $h_i(X, Y)$'s into a vector $\mathbf{h}(X, Y)$, CRF tries to find a θ that maximizes

$$P_\theta(Y|X) = \tfrac{1}{Z_\theta(X)} \exp\{\theta \cdot \mathbf{h}(X, Y)\}, \qquad (15)$$

where $Z_\theta(X) = \sum_Y \exp\{\theta \cdot \mathbf{h}(X, Y)\}$ is a normalization term. In general, given (X_L, Y_L), we can train the CRF model θ by optimizing the negative log-likelihood:

$$\mathcal{L}_\theta = -\tfrac{1}{n_L} \sum_{i=1}^{n_L} \log P_\theta(Y_L^{(i)}|X_L^{(i)}) + \tfrac{\gamma}{2}\|\theta\|_2^2, \qquad (16)$$

where $\|\cdot\|_2$ is a ℓ_2-norm and $\gamma \geq 0$ is a regularization parameter.

We use constraints as weak supervision to incorporate the unlabeled data. We first estimate the probability for a constraint c as

$$P(f_c(\mathbf{x}, \mathbf{y}) = 1|g_c(\mathbf{x}, \mathbf{y}) = 1) = \tfrac{\mathbb{E}_q[f_c(X_U, Y_U)]}{\mathbb{E}_q[g_c(X_U, Y_U)]},$$

where $\mathbb{E}_q[f_c(X_U, Y_U)]$ is the expected number of instances with c being satisfied, $\mathbb{E}_q[g_c(X_U, Y_U)]$ is the expected number of instances with c being applicable. Given n_C constraints and each of them having an empirical probability $\eta_c \in [0, 1]$, we denote $\boldsymbol{\eta} = [\eta_1, ..., \eta_{n_C}]$. We also denote $\mathbf{g}(X, Y) = [g_1(X, Y), ..., g_{n_C}(X, Y)]$ and $\mathbf{f}(X, Y) = [f_1(X, Y), ..., f_{n_C}(X, Y)]$. Thus, by making each $P(f_c(\mathbf{x}, \mathbf{y}) = 1|g_c(\mathbf{x}, \mathbf{y}) = 1)$ equal to its η_c, we have

$$\mathbb{E}_q[\mathbf{f}(X, Y)] = \boldsymbol{\eta} \circ \mathbb{E}_q[\mathbf{g}(X, Y)], \qquad (17)$$

where $\circ$ is the element-wise product. For simplicity, we let $\boldsymbol{\zeta}(X, Y) = \mathbf{f}(X, Y) - \boldsymbol{\eta} \circ \mathbf{g}(X, Y)$; then $\mathbb{E}_q[\boldsymbol{\zeta}(X, Y)] = \mathbf{0}$.

Finally, we try to find a target distribution $q(Y|X; C)$ that approximates $P_\theta(Y|X)$ on the labeled data (through minimizing the KL-divergence between q and P_θ) and matches the constraints on the unlabeled data (through satisfying Eq. 17) at the same time:

$$\begin{aligned}\min_{\theta, q \in \Delta}\ & \mathcal{L}_\theta + \frac{\alpha_1}{n_U} \sum_{i=1}^{n_U} KL(q(Y|X^{(i)}; C)||P_\theta(Y|X^{(i)})) \\ & + \tfrac{\alpha_2}{2n_U} \sum_{i=1}^{n_U} \|\mathbb{E}_q[\boldsymbol{\zeta}(X^{(i)}, Y)]\|_2^2,\end{aligned} \qquad (18)$$

where Δ is a simplex s.t. $\sum_Y q(Y|X; C) = 1$.

4.3 Efficient Inference with Constraints

Optimizing Eq. 18 is hard due to the complication of y-type constraints. Take RC_2 as an example. As we do not know which pair of text snippets are BIOGRAPHY for the unlabeled data, we have to evaluate every pair of hidden variables, resulting in a complete (or at least densely connected) graph over the hidden variables. Inference with an over densely connected graph is hard, thus a efficient inference design is needed. Our intuition is that, actually we do not have to evaluate all the pairs of hidden variables, since we only care about those pairs that are truly BIOGRAPHY. If we can guess which text snippets are likely to be BIOGRAPHY, then we can focus on a much simpler graph. Formally, for a y-type constraint c, we estimate whether an instance $\mathbf{x}_j$ and its labels $\mathbf{y}_j$ are relevant to c by $\Pr(g_c(\mathbf{x}_j, \mathbf{y}_j) = 1)$. Denote $\mathbf{y}^c$ as the preferred labels for c, such that $g_c(\mathbf{x}_j, \mathbf{y}_j = \mathbf{y}^c) = 1$; e.g., the preferred labels for RC_2 are BIOGRAPHY's. Then, we estimate $\Pr(g_c(\mathbf{x}_j, \mathbf{y}_j) = 1)$ by $P_\theta(\mathbf{y}_j = \mathbf{y}^c|X)$, or $P_\theta(\mathbf{y}_j^c|X)$ for short. Denote ϵ_c as the selection threshold for c. Finally, we can introduce a selection indicator $\delta(\log P_\theta(\mathbf{y}_j^c|X) \geq \epsilon_c)$ for each y-type constraint c as

$$\zeta_c^* = \begin{cases} \delta(\log P_\theta(\mathbf{y}_j^c|X) \geq \epsilon_c)\zeta_c(\mathbf{x}_j, \mathbf{y}_j), & \text{if } c \text{ is y-type;} \\ \zeta_c(\mathbf{x}_j, \mathbf{y}_j), & \text{otherwise.} \end{cases} \qquad (19)$$

Thus, instead of $\boldsymbol{\zeta}$, we use $\boldsymbol{\zeta}^* = [\zeta_1^*, ..., \zeta_{n_C}^*]$ in Eq. 18 for a new objective function with selective evaluation.

We automatically learn the selection thresholds ϵ_c's for different constraints. Our intuition is that, if the labeled data and the unlabeled data follow the same distribution, then the percentage of relevant instances for a constraint c over the unlabeled data is close to that over the labeled data. Denote m_c as the number of instances for constraint c on unlabeled data (X, Y); i.e., $m_c = |\{(\mathbf{x}_j, \mathbf{y}_j)|\mathbf{x}_j \in X^d, \mathbf{y}_j \in Y^d\}|$. Denote π_c as the percentage of relevant instances over all the m_c instances. We can estimate π_c as

$$\pi_c = \tfrac{1}{m_c} \sum_{j=1}^{m_c} \delta(\log P_\theta(\mathbf{y}_j^c|X) \geq \epsilon_c) g(\mathbf{x}_j, \mathbf{y}_j^c).$$

Denote $\tilde{\pi}_c \in [0, 1]$ as the empirical percentage of relevant instances over all the labeled data. Then, we define a *meta-constraint* for constraint c as: $\pi_c = \tilde{\pi}_c$. For each unlabeled page $X_U^{(i)} \in D_U$, we estimate one $\pi_c^{(i)}$. Denote $\boldsymbol{\epsilon} = [\epsilon_1, ..., \epsilon_{n_C}]$ with each $\epsilon_c \leq 0$. In all, we derive a new objective function as

$$\begin{aligned}\min_{\substack{\theta, q \in \Delta, \\ \boldsymbol{\epsilon} < \mathbf{0}}} & \mathcal{L}_\theta + \tfrac{\alpha_1}{n_U} \sum_{i=1}^{n_u} KL(q(Y|X^{(i)}; C)||P_\theta(Y|X^{(i)})) \\ & + \tfrac{\alpha_2}{2n_U} \sum_{i=1}^{n_U} \|\mathbb{E}_q[\boldsymbol{\zeta}^*(X^{(i)}, Y)]\|_2^2 \\ & + \tfrac{\alpha_3}{2n_U} \sum_{i=1}^{n_U} \sum_{c \in C_y} [m_c(\pi_c^{(i)} - \tilde{\pi}_c)]^2,\end{aligned} \qquad (20)$$

where $\alpha_3 \geq 0$ is a trade-off parameter. Eq. 20 is our ultimate objective function. We leave the details of its optimization in [15].

5. EVALUATION

5.1 Evaluation for Data Harvesting

Data set. For repeatable results, we conduct experiments over a corpus collected from the Web in advance, and all queries will retrieve pages from this corpus only. We prepared corpora for the *researcher* domain. In total, we have 996 researchers randomly chosen from DBLP's most prolific authors[4]. For each entity, we attempted to collect 50 pages from the Web to construct the corpora. To retrieve pages from the corpora, we used a language model with Dirichlet smoothing [14] as the search engine. For each query, pages in the corpus are ranked and the top 5 are returned.

Evaluation methodology. We randomly reserved half of the entities as "domain entities" for training, and the remaining as "target entities" for testing. Target entities were further divided into two equal splits, one for parameter validation, and the other for testing. We repeated the split randomly for 10 times.

[4]http://dblp.uni-trier.de/statistics/prolific1.html

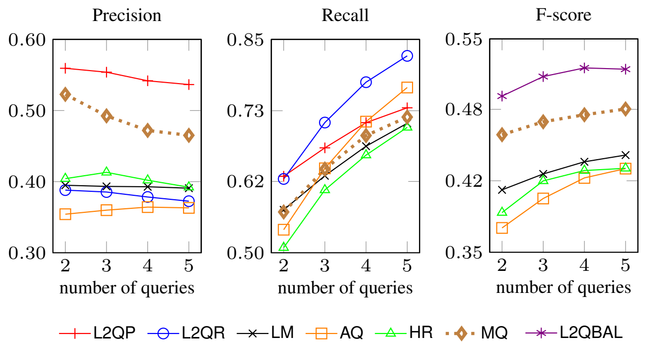


Figure 5: Using L2Q for data harvesting.

On the testing set, we evaluate the retrieved pages in terms of their actual precision, recall and F-score for every target entity and aspect. We then normalize the results against an *ideal* solution for fair comparison across different entities We design the ideal solution as selecting queries that maximize the product of their *actual* coverage and precision, which can be obtained by feeding each candidate query to the search engine. Given multiple target entities and test splits, we report the average results over all entities and splits. We further average the results across aspects, omitting the detailed performance for every aspect due to space constraint.

Performance. We compare our approaches L2QP (L2Q for Precision), L2QR (L2Q for Recall), and their combination L2QBAL (L2Q for F-score, where we choose queries based on their geometric mean of L2QP collective utility and L2QR collective utility) with four independent baselines: 1) *Language Model* (LM), based on the language feedback model [13]; 2) *Adaptive Querying* (AQ), based on the adaptive query selection policy [12]; 3) *Harvest Rate* (HR), based on the harvest rate heuristic [11]; 4) *Manual Querying* (MQ), based on human designed queries. The first three baselines are *algorithmic* methods adapted from related problems, since there is no previous work on our exact setting. The fourth baseline is a *manual* approach based on a user study.

In Fig. 5, we vary the number of queries, and report the results on precision, recall and F1. In terms of precision, L2QP achieves the best performance, surpassing not only the baselines, but also L2QR, since L2QP is designed to optimize precision. On average, L2QP beats the best algorithmic baselines by 28%, and the manual baseline by 14%. In terms of recall, L2QR likewise outperforms all the other methods, as it is designed to optimize recall. On average, L2QR beats the best algorithmic baseline by 11%, and the manual baseline by 14%. In terms of F-score, L2QBAL outperforms L2QP and L2QR; it is also consistently better than all the baselines for various number of queries. On average, it beats the best algorithmic baseline by 16%, and the manual baseline by 10%.

5.2 Evaluation for Data Integration

Data set. We prepared a corpus for the *researcher* domain, with in total 1,003 researchers. For each researcher, we collected a set of Web pages. For each page, we further parse it into a set of text snippets. In total, we got 3,002 pages and 48.2K snippets. We labeled[5] the text snippets of 100 entities as labeled training data, and the snippets of another 103 entities as test data. We leave the other 800 entities as unlabeled training data. There are 11 entity aspects used as the labels. We used the constraints in Table 1. These constraints

[5]The labeling was done by two human judges, who achieved an agreement of 84% for the researcher domain.

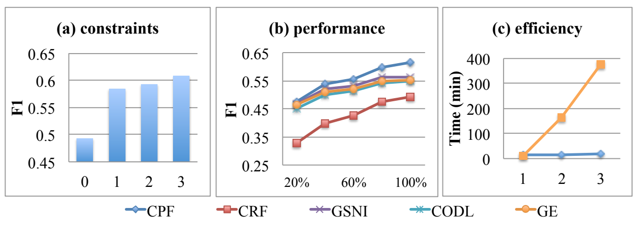


Figure 6: Using CPF for data integration.

were designed with two guidelines: first, as none of the existing work supports y-type constraints, we only focus on experimenting with y-type constraints; second, we consider y-type constraints as up to second order (*i.e.*, involving two hidden variables in one constraint), which are the most common data dependencies.

Evaluation methodology. As our task is classification, we use the F1 score to evaluate the performance on each class. Generally, there are two kinds of classes (i.e., labels): one is *constraint relevant* where the class is used by at least one constraint from Table 1 in the data set, the other is *constraint irrelevant* where the class is never used by any constraint. For example, in Table 1, the constraint relevant classes are EDUCATION, BIOGRAPHY and EMPLOYMENT; the constraint irrelevant classes are AWARDS, RESEARCH and so on. As we are more interested in the performance change on the constraint relevant classes, we combine all the constraint irrelevant classes as one big class *Others* in evaluation and define its performance as the average F1 score of all its constituting constraint irrelevant classes. Finally, given the F1 score of each constraint relevant class and the F1 score of *Others*, we define their average as a model's overall F1 score. We run experiments for five times and report the average of a model's overall F1 scores for comparison.

Performance. We compare our CPF model with the following baselines: 1) CRF [7], which is the basic structured classifier without constraint; 2) GSNI [1], CODL [4] and GE/PR [2], which are the existing constraint formulations that can work with non-logic structured classifiers. Note that as none of these baselines supports y-type constraints, we adapted them to use the constraints.

In Fig. 6(a), we evaluate the usefulness of y-type constraints RC_1, RC_2 and RC_3. As we can see, all the constraints improve the performance. In Fig. 6(b), we compare CPF with the baselines as training data amount changes from 20% to 100% for each entity. As we can see, CPF is generally better than the baselines. When using 100% data for each entity, CPF achieves 9.1%-24.2% relative F1 improvement over the baselines. All the improvements are significant, with t-test $p \leq 0.05$. In Fig. 6(c), we evaluate our efficient inference with y-type constraints. When using all the three constraints, CPF is 19.0× faster than the baselines.

5.3 Evaluation for ARISE-PIE System

We conduct system evaluations in order to measure: 1) the absolute performance on a benchmark data set; 2) the relative performance compared with human using Google to find information by themselves through a user study.

Benchmark data. We create a benchmark data set with 30 researchers in computer science. These researchers are picked to ensure the diversity in research fields, geographical regions and career stages. For each researcher, we collect and label at most 60 web pages from Google and the structured data from several people listing services. We use this data set to measure the macro average precision, recall and F1 across different attributes and different researchers, as the system's absolute performance.

Table 2: The system evaluation results. Some notations' meanings: "*P*"=Precision, "*R*"=Recall, "*Rel.*"=Relative).

Attributes	**Absolute Scores**			**Relative Scores**		
	P	*R*	*F1*	*Rel. P*	*Rel. R*	*Rel. F1*
Contact	0.52	0.53	0.52	0.92	0.92	0.92
Birthday	0.79	0.79	0.79	0.83	0.83	0.83
Nationality	0.84	0.88	0.84	1.11	1.11	1.11
Award	0.55	0.56	0.51	0.50	0.50	0.50
Employment	0.63	0.41	0.50	0.89	0.78	0.82
Education	0.63	0.64	0.62	0.89	0.89	0.89
Course	0.55	0.63	0.53	0.44	0.44	0.44
Publication	0.77	0.76	0.73	1.41	1.21	1.24
Social accounts	0.81	0.81	0.81	1.26	1.26	1.26
Book	0.73	0.88	0.73	1.67	1.67	1.67
Homepage	0.54	0.50	0.51	1.10	1.22	1.16
Average	0.68	0.70	0.67	0.93	0.90	0.91

User study. We design a user study to let six participants use Google to answer questions about the researchers in the benchmark data set. For each participant, we sample 18 questions from a big question pool we compiled. Each question covers one different researcher, and it has to be answered within five minutes to control the experiment time. The participants can freely use Google with as many different queries as they like, and they can read all the search results regardless of the formats (e.g., HTMLs, PDFs, etc.). For comparison, we also use our system to answer these questions. For each question, we allow at most three queries to try our system. We measure the relative precision, recall and F1 by dividing our results with the human results. The larger a score is, the better.

Performance. We summarize the results in Table 2. On the benchmark data set, our system can achieve a 0.67 absolute F1. In addition to the inevitable prediction errors from the system, there are several possible reasons to cause the performance deficiency. First, in the evaluation, we only collect 10 pages from Google. This may miss some information. Second, our system only processes HTML pages, but not other data formats (e.g., PDF résumés).

On the user study, our system can achieve a 0.91 relative F1 compared with the human results. This has two implications. First, our performance is lower than that of human results, which is expected since human can read more types of data, and their exactions are highly accurate. In some sense, they are an upper bound for us. Second, our performance is close to human results, but we are fully automatic once the queries are given.

6. CONCLUSION

In this paper, we study the problem of how to automatically search and integrate people information on the Web. We solve two major technical challenges: 1) data harvesting, which aims to automatically crawl relevant Web pages for a target entity through search engine. We propose a novel L2Q model to find a set of best queries to maximize some collective utility in terms of the crawled pages; 2) data integration, which aims to collectively extract the target entity's information from the crawled pages with the help of constraints. We propose a novel CPF model to formulate flexible forms of constraints and develop an efficient semi-supervised model by selectively evaluating the constraints on unlabeled data. Based on our two innovations on data harvesting and integration, we develop a people search system ARISE-PIE for the research domain. We evaluate our L2Q and CPF models on the real-world data sets. In data harvesting, L2Q achieves 16% and 11% relative F-score improvement than the best algorithmic baseline and the manual baseline respectively. In data integration, CPF achieves 9.1% relative F-score improvement and 19.0× speedup than the best baselines. We also evaluate our ARISE-PIE system with a benchmark data set and a user study. We achieve 0.67 absolute F-score and 0.91 relative F-score compared with the manual results.

In the future, we plan to exploit more types of data such as PDF and social media. We also plan to improve our extraction accuracy by leveraging more structured sources to generate labeled data.

Acknowledgement

This work is supported by the research grant for Human-Centered Cyber-physical Systems Programme at Advanced Digital Sciences Center from Singapore's Agency for Science, Technology and Research (A*STAR). V. Zheng is also supported by the research grant (#61502418) from the National Nature Science Foundation of China.

7. REFERENCES

[1] S. Anzaroot, A. Passos, D. Belanger, and A. McCallum. Learning soft linear constraints with application to citation field extraction. In *ACL*, pages 593–602, 2014.

[2] K. Bellare, G. Druck, and A. McCallum. Alternating projections for learning with expectation constraints. In *UAI*, pages 43–50, 2009.

[3] R. Bunescu and R. J. Mooney. Collective information extraction with relational markov networks. In *the 42nd Annual Meeting on Assoc. for Comp. Ling.*, ACL '04, 2004.

[4] M.-W. Chang, L. Ratinov, and D. Roth. Structured learning with constrained conditional models. *Mach. Learn.*, 88(3):399–431, Sept. 2012.

[5] M. Diligenti, F. Coetzee, S. Lawrence, C. L. Giles, and M. Gori. Focused crawling using context graphs. In *VLDB*, pages 527–534, 2000.

[6] Y. Fang, V. W. Zheng, and K. C. Chang. Learning to query: Focused web page harvesting for entity aspects. In *ICDE*, pages 1002–1013, 2016.

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

[8] G. S. Mann and A. McCallum. Generalized expectation criteria for semi-supervised learning with weakly labeled data. *Journal of Mach. Learn. Research*, 11:955–984, 2010.

[9] A. F. T. Martins. Transferring coreference resolvers with posterior regularization. In *ACL*, pages 1427–1437, 2015.

[10] M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62(1-2):107–136, 2006.

[11] P. Wu, J.-R. Wen, H. Liu, and W.-Y. Ma. Query selection techniques for efficient crawling of structured web sources. In *ICDE*, page 47, 2006.

[12] P. Zerfos, J. Cho, and A. Ntoulas. Downloading textual hidden web content through keyword queries. In *Digital Libraries, 2005*, pages 100–109, 2005.

[13] C. Zhai and J. Lafferty. Model-based feedback in the language modeling approach to information retrieval. In *CIKM*, pages 403–410, 2001.

[14] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *SIGIR*, pages 334–342, 2001.

[15] V. W. Zheng and K. C.-C. Chang. Regularizing structured classifier with conditional probabilistic constraints for semi-supervised learning. In *CIKM*, 2016.