5-2016

Learning to query: Focused web page harvesting for entity aspects

Yuan FANG

Singapore Management University, yfang@smu.edu.sg

Vincent W. ZHENG

Kevin Chen-Chuan CHANG

DOI: https://doi.org/10.1109/ICDE.2016.7498308

Learning to Query: Focused Web Page Harvesting for Entity Aspects*

Yuan Fang †, Vincent W. Zheng ‡, Kevin Chen-Chuan Chang #‡
yfang@i2r.a-star.edu.sg, vincent.zheng@adsc.com.sg, kcchang@illinois.edu
† Institute for Infocomm Research, Singapore
‡ Advanced Digital Sciences Center, Singapore
# University of Illinois at Urbana-Champaign, USA

***Abstract*—As the Web hosts rich information about real-world entities, our information quests become increasingly entity centric. In this paper, we study the problem of focused harvesting of Web pages for entity aspects, to support downstream applications such as business analytics and building a vertical portal. Given that search engines are the *de facto* gateways to assess information on the Web, we recognize the essence of our problem as *Learning to Query* (L2Q)—to intelligently select queries so that we can harvest pages, via a search engine, focused on an entity aspect of interest. Thus, it is crucial to quantify the utilities of the candidate queries w.r.t. some entity aspect. In order to better estimate the utilities, we identify two opportunities and address their challenges. First, a target entity in a given domain has many peers. We leverage these peer entities to become *domain aware*. Second, a candidate query may "overlap" with the past queries that have already been fired. We account for these past queries to become *context aware*. Empirical results show that our approach significantly outperforms both algorithmic and manual baselines by 16% and 10% in F-scores, respectively.**

I. INTRODUCTION

The Web is turning into a rich repository of all kinds of data. In particular, *entities* and their *aspects*, such as SPOUSE of celebrities and SAFETY of cars, form the majority of the information need of everyday Web users. Bing reported that people entities alone account for 10% of all their search volume [1]. Unfortunately, various aspects of the same entity often scatter across many different pages on the Web. For all intents and purposes, it is convenient to gather target pages through a search engine, as search engines can handle universal queries to cater for different entities and aspects.

However, querying a search engine and downloading the result pages in a large scale often require significant time and bandwidth, as well as a considerable financial cost to access commercial search APIs. Therefore, it is important to ask the "right" queries so that a search engine can return us the pages of interest (*i.e.*, target pages). Manually designing the queries is not feasible for two reasons. First, if we know little about the target entity to begin with, it is hard to come up with good queries. Second, a manual approach simply cannot scale to a large number of entities in different domains. A *domain* simply consists of a particular kind of entities, such as researchers or cars. Gathering a corpus for different entity aspects is often the first step towards building various data-centric applications, as the following examples illustrate.

- *Business analytics.* Analyzing pages that mention some specific aspect of a product, such as BATTERY or SCREEN of iPhone 6s, enables business to drill down and understand customers' needs in a finer granularity. In particular, sentiment analysis or opinion mining can be performed on the harvested pages.
- *Vertical portal or search.* There already exist a few such sites in different verticals (*i.e.*, domains), such as ArnetMiner.org for researchers and Edmunds.com for cars. To build these sites, information extraction is typically applied on the harvested pages, which must be comprehensive to cover many aspects of each entity, such as RESEARCH and AWARD of researchers, as well as PRICE and SAFETY of cars.

Towards these interesting applications, it is thus crucial to systematically and automatically formulate the right queries to enable large-scale page harvesting for entity aspects. Note that, although we can use generic queries for entities in the same domain (*e.g.*, entity name + `research` to find professors' RESEARCH aspect), entity-specific keywords (*e.g.*, `data mining` for Philip Yu and `parallel` for Marc Snir) can be far more superior, as our experiments will show.

It is natural to adopt an iterative process to harvest pages, which mimics everyday Web users. Suppose users are looking for Marc Snir's RESEARCH aspect. They often start with a *seed* query like `marc snir uiuc` to obtain a few initial pages. From these pages, they learn that Snir studies parallel computing. This new information enables users to start another iteration by reformulating the query to include `parallel`. It is a more specific keyword about Snir's RESEARCH, and thus is more likely to retrieve relevant pages. This process continues until some budget (*e.g.*, time) is reached.

Learning to query. The iterative querying process is outlined in Fig. 1. In each iteration, the core task is to select the best query among a pool of candidate queries based on current result pages. We call this core task *learning to query* (L2Q), which is the focus of this paper.

Firstly, 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

* This material is based upon work partially supported by the research grant for the Human-centered Cyber-physical Systems Programme at the Advanced Digital Sciences Center from Singapore's Agency for Science, Technology and Research. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the funding agencies.

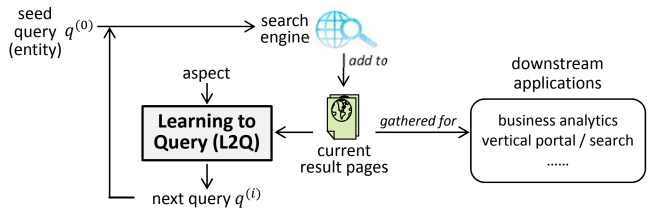


Fig. 1: High-level flow to harvest pages for entity aspects.

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. Of course, the utility should be inferred *without* actually firing any candidate query.

Secondly, an entity does not exist in isolation. There are often a large number of peer entities in the same domain, 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.

Thirdly, 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.

Problem formulation. While a few existing crawling approaches [2], [3], [4], [5] also use queries, they are not designed for entity aspects. Moreover, many of them have incomplete or ad-hoc utility models, and often fail to leverage the domain or context, as we will elaborate in Sect. II. In the following, we formalize the problem of L2Q, which boils down to domain and context-aware utility inference.

Data model. We view each page and each query as a bag of words, respectively. Each word is a term or phrase depending on the tokenization. Moreover, a query can retrieve a set of pages through an information retrieval model, such as a commercial search engine.

Input. Our goal is to gather pages for a target entity, focused on a given aspect.

The *target entity* is given by a *seed query* $q^{(0)}$ that uniquely identifies it. Examples are `marc snir uiuc` (name + institute) and `BMW 3 series 328i` (make + model). While more sophisticated entity disambiguation techniques exist [6], they are beyond the scope of this paper. Note that the seed query is not only the initial query to bootstrap the entire process, but also appended to subsequent queries when submitting them to the search engine, in order to focus on the target entity.

The *target aspect* is given by a function $Y : P \to \{1, 0\}$, which maps each page $p \in P$ to relevant (1) or irrelevant (0). Essentially, we classify entity pages based on the interest. For instance, if we are only interested in researchers' RESEARCH and EDUCATION, their pages about HOBBY would be perceived as irrelevant. More generally, Y can map a page to a real-valued relevance score, but for ease of discussion we keep to the binary function for now. (In implementation, as Sect. VI will further explain, we employ a pre-trained classifier for each aspect to materialize Y.)

Output. In each iteration of Fig. 1, we form the candidate query set Q for the target entity. We then select the best query

$$q^* = \arg\max_{q \in Q} \mathcal{U}^{(Y)}(q), \tag{1}$$

where $\mathcal{U}^{(Y)}(q)$ measures the utility of query q w.r.t. aspect Y. We will omit the superscript $^{(Y)}$ and only write $\mathcal{U}(*)$ hereafter to implicitly assume a target aspect Y. $\mathcal{U}(*)$ shall be inferred in a domain and context-aware manner, as follows.

Subproblem 1: Domain-aware L2Q. We are given the additional input of already gathered pages of other entities in the same domain, which we call *domain pages*.

However, exploiting domain pages is non-trivial. Entities often have different word distributions even in the same domain. For instance, Marc Snir is associated more with the word `parallel`, whereas Philip Yu is associated more with `data mining`, even though both belong to the domain of researchers. That is, useful queries for one entity (*e.g.*, `parallel` for Snir) do not necessarily align with another entity (*e.g.*, Yu) in the same domain. Thus, instead of directly learning queries from domain pages, we propose to learn some "templates," which are query abstractions consistent across entities. Based on these templates, we need to further infer the utilities of candidate queries for the target entity.

Subproblem 2: Context-aware L2Q. In iteration-i of Fig. 1, there is a context of past queries $q^{(0)}, q^{(1)}, \ldots, q^{(i-1)}$, which were fired in previous iterations.

However, different queries often retrieve redundant pages. Consider an example for Snir's RESEARCH aspect. `hpc` and `parallel` are both useful queries on their own, but their respective top 5 result pages from Google have 2 pages in common at the time of writing. Such redundancy implies that a set of individually best queries are not necessarily the best set of queries collectively. In other words, the collective utility of multiple queries is not a simple sum or average of their individual utilities. Thus, in addition to the candidate queries themselves, we propose to account for the context of past queries, in order to capture the effect of redundancy.

Contributions. We summarize the contributions of this paper.

First, we proposed the problem of L2Q. Note that our settings depart from previous query learning approaches.

Second, we developed models for domain and context-aware L2Q. We started with a basic utility inference model for queries, which was then enhanced with the domain and context. For domain awareness, we devised templates to address entity variations; for context awareness, we formulated collective utilities to manage multiple queries.

Third, we empirically evaluated L2Q in two real-world domains. Our approach significantly outperforms both algorithmic and manual query baselines, by 16% and 10% in average F-score, with only a minor computational overhead.

II. Related Work

Problem. Our problem can be viewed from two perspectives, namely Web crawling and query learning.

On the one hand, our setting differs from traditional Web crawling [7], [8], [9], which follow links in the gathered pages. In contrast, L2Q is driven by queries—we can intelligently formulate queries to cater to different entities and aspects via a search engine. L2Q also differs from deep Web crawling [2], [10], although they also harvest results through queries. They only deal with structured records and queries, instead of unstructured texts in our scenario. Thus, their query space can be sometimes exhaustively enumerated from the query interface [10]. However, our query space dynamically expands as we see new pages. Moreover, there also exist a few approaches for crawling text databases using free-text queries [3], [4], [5]. Unfortunately, all these query-based approaches fall short in utility inference or leveraging the domain or context, as we will discuss later.

On the other hand, our goal is distinct from that of query completion or suggestion [11], [12]. We aim to use additional queries to harvest more pages for the given entity aspect, analogous to using more links in traditional crawling. On the contrary, query completion or suggestion aim to redirect users when their original query is ineffective, through means of generalization, specialization or error correction of the original query, often based on patterns in a query log.

Utility inference. L2Q systematically measures both precision and recall as two complementary utilities for queries. In contrast, query-driven approaches [2], [10], [5] for the deep Web aim to gather the entire database and thus only optimize recall, not precision. Other methods [3], [4] lack an explicit formulation of utilities. Besides, some information extraction systems like Snowball [13] and KnowItAll [14] also estimate the utility of their extraction patterns, but as discussed elsewhere [15], they only estimate precision in an informal and heuristic way, and do not model recall. On the contrary, we model both precision and recall in a unified probabilistic framework. While our probabilistic framework is inspired by a limited number of studies [16], [15] (which solve different problems from ours), our solution of domain and context-awareness encompasses significant novelties beyond them.

Domain awareness. L2Q leverages domain data to bootstrap and enhance query learning. However, learning from the domain is non-trivial due to entity variations—different entities in the same domain can still have very different word distributions. Subsequently, we propose to bridge domain and target entities with templates. While some approaches for query-based crawling [4], [5] and related problems [16], [15] do not learn from a separate set of domain data at all, others [2], [3] do not need to bridge between domain and target data. Lastly, existing domain adaptation [17] and transfer learning [18], [19] techniques cannot be applied to our setting. These methods are intended for classifying or clustering instances from different domains, which have different feature distributions [17], [19] or even feature spaces [18]. On the contrary, in our work, the notion of instance does not exist. In particular, we do not perceive queries as instances, since we have no supervision or explicit feature representation for queries.

Context awareness. L2Q is also context-aware: each candidate query is considered in conjunction with the context of past queries, since different queries often retrieve redundant pages. However, the solutions for query-based crawling [3], [4] and related problems [16], [15] largely overlooked the issue of redundancy. Finally, while the paradigms of active learning [20] and relevance feedback [21], [22], [23] also appear to consider previous context, we emphasize the difference. Active learning selects the most ambiguous example relative to previously chosen examples, to re-train and improve their model. On the other hand, relevance feedback updates the query model using feedback on the previous query, to improve their retrieval accuracy. In either case, the objective is not to avoid redundancy with previous queries.

III. Utility Inference for L2Q

We begin with a conceptual model to infer the utilities for queries (Eq. 1), without the domain and context for now.

Insight. We hinge on the intuition of mutual reinforcement between pages and queries. In general, a "useful" page for the target aspect Y contains useful queries for Y, and a useful query can retrieve useful pages for Y. Thus, our measure of "usefulness," or utility $\mathcal{U}(*)$, shall uniformly apply to both pages and queries alike. That is, for a page p and a query q, high $\mathcal{U}(p)$ implies high $\mathcal{U}(q)$ if p contains q, and vice versa. The mutuality can be formalized to relate and estimate $\mathcal{U}(p)$ and $\mathcal{U}(q)$ in a unified model.

As the first step, we must quantify the usefulness of pages and queries. Given the ultimate goal to gather pages, we can measure two complementary forms of utility: precision and recall of the retrieved pages w.r.t. Y. From the utilities (precision or recall) of these pages, we can further infer the corresponding utilities of related queries.

Interestingly, the line of work on *probabilistic precision and recall* is amenable to the utility inference of pages and queries, although these previous studies address different problems such as user intent interpretation [16] and relation extraction [15]. In this paper, we adopt probabilistic precision and recall as our utility inference model, towards solving the distinct problem of L2Q. In particular, we will address issues beyond their considerations—the incorporation of domain and context into our learning.

Definition of utilities. Assume a universe of pages P. These pages correspond to a universe of candidate queries Q. (In other words, Q can be generated from P, such as by taking all n-grams in P as queries.) As we are concerned about retrieving pages, to quantify usefulness, we need to examine the pages behind each notion in the framework.

Let Ω denote a mapping from each of the following notions to a set of pages in the domain, *i.e.*, $\Omega(*) \subseteq P$.

- $\Omega(Y)$, pages that are relevant to the target aspect Y.
- $\Omega(q)$, pages that can be retrieved by query q.
- $\Omega(p)$, the page p itself, *i.e.*, $\Omega(p) = \{p\}$.

Consider a running example in Fig. 2(a)–(b). For Y as RESEARCH, suppose there are six pages $p_1, \ldots, p_6$ in the

(a) Example pages (Y is RESEARCH)

	Content	$Y(p_i)$
p_1	He conducts research on parallel and hpc systems.	1
p_2	He published many papers on parallel & hpc research.	1
p_3	His research on complexity of parallel algorithms is valuable.	1
p_4	He also studies computational complexity at U Illinois.	1
p_5	Visit him at Siebel Center, U Illinois.	0
p_6	He was a senior manager at IBM before joining U Illinois.	0

(b) Example queries

Query	Retrievable pages
q_1: `parallel research`	$p_1: 1,\ p_2: 1,\ p_3: 1$
q_2: `hpc research`	$p_1: 1,\ p_2: 1$
q_3: `complexity`	$p_3: 1,\ p_4: 1$
q_4: `u illinois`	$p_4: 1,\ p_5: 0,\ p_6: 0$
q_5: `ibm`	$p_6: 0$

(c) Reinforcement graph

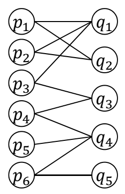


Fig. 2: Running illustration for Marc Snir.

universe. Among them, $p_1, \dots, p_4$ are relevant, *i.e.*, $\Omega(Y) = \{p_1, \dots, p_4\}$. Meanwhile, as an example, query q_2 can retrieve $\Omega(q_2) = \{p_1, p_2\}$.

Given the sets $\Omega(*)$, $\forall v \in P \cup Q$, we can compute v's precision or recall w.r.t. Y as its utility.

$$\text{precision}^{(Y)}(v) = |\Omega(Y) \cap \Omega(v)|/|\Omega(v)| \tag{2}$$

$$\text{recall}^{(Y)}(v) = |\Omega(Y) \cap \Omega(v)|/|\Omega(Y)| \tag{3}$$

In reality, we do not observe the entire universe of P, as there exist additional pages beyond the six pages. Therefore, precision and recall cannot be computed exactly by counting. Thus, parallel to Eq. 2–3, we develop the probabilistic counterparts of precision and recall as our two utility measures, denoted $\mathcal{P}$ and $\mathcal{R}$ respectively. That is, $\mathcal{U}$ can be instantiated as either $\mathcal{P}$ or $\mathcal{R}$.

$$\mathcal{P}(v) \triangleq P(\omega \in \Omega(Y) | \omega \in \Omega(v)), \tag{4}$$

$$\mathcal{R}(v) \triangleq P(\omega \in \Omega(v) | \omega \in \Omega(Y)), \tag{5}$$

where ω is a random page from $\Omega(*)$. For brevity, we have omitted $^{(Y)}$ from the notations and only write $\mathcal{P}$ or $\mathcal{R}$, to implicitly mean that we are measuring the utilities w.r.t. the target aspect Y. Interestingly, these definitions apply uniformly to pages and queries, enabling us to capture their mutual reinforcement in a unified way.

Mutual reinforcement. As our intuition, useful pages can lead to useful queries, and vice versa. For the running example in Fig. 2(a)–(b), p_1 is useful and contains a useful query q_1, while q_1 can further discover useful pages p_2 and p_3.

Such mutual reinforcement can be modeled by a reinforcement graph $G = (V, E)$, as illustrated in Fig. 2(c). The vertex set of G is $V = P \cup Q$, and the edge set E can be 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. More generally, W_{pq} can also encode the connection strength in $[0, \infty)$. For instance, we can use a retrieval model to quantify the strength between page p and query q, as a higher retrieval score implies that q is more likely to retrieve p.

Subsequently, mutual reinforcement exists between neighboring vertices on G. 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 (say 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.*, $N(p_1) = \{q_1, q_2\}$.

Precision of page. We start with rewriting the utility of precision $\mathcal{P}(q)$ in Eq. 6. In line 1, we expand to joint distributions with pages in $\Omega(q)$ which are also q's neighboring pages. Line 2 follows from Bayes' rule. We obtain line 3 as ω's relevance to Y only depends on p given that $\omega = p$, and $\omega = p$ is equivalent to $\omega \in \{p\} = \Omega(p)$. Observe that the first term is the probability that a page retrieved by q turns out to be p, among all pages in $N(q)$. We can therefore estimate it as $W_{pq}/\sum_{p' \in N(q)} W_{p'q}$. Moreover, the second term is simply $\mathcal{P}(p)$ by the definition in Eq. 4. Thus, we arrive at line 4. Intuitively, 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.*, $P(\omega = p | \omega \in \Omega(q))$.

$$\begin{aligned}
&\mathcal{P}(q) \triangleq P(\omega \in \Omega(Y) | \omega \in \Omega(q)) \\
&\overset{1}{=} \textstyle\sum_{p \in N(q)} P(\omega \in \Omega(Y), \omega = p | \omega \in \Omega(q)) \\
&\overset{2}{=} \textstyle\sum_{p \in N(q)} P(\omega = p | \omega \in \Omega(q)) P(\omega \in \Omega(Y) | \omega = p, \omega \in \Omega(q)) \\
&\overset{3}{=} \textstyle\sum_{p \in N(q)} P(\omega = p | \omega \in \Omega(q)) P(\omega \in \Omega(Y) | \omega \in \Omega(p)) \\
&\overset{4}{=} \textstyle\sum_{p \in N(q)} \frac{W_{pq}}{\sum_{p' \in N(q)} W_{p'q}} \mathcal{P}(p)
\end{aligned} \tag{6}$$

Recall of page. Likewise, we derive a counterpart for recall in Eq. 7. Intuitively, 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.*, $P(\omega \in \Omega(q) | \omega = p)$.

$$\begin{aligned}
&\mathcal{R}(q) \triangleq P(\omega \in \Omega(q) | \omega \in \Omega(Y)) \\
&= \textstyle\sum_{p \in N(q)} P(\omega \in \Omega(q), \omega = p | \omega \in \Omega(Y)) \\
&= \textstyle\sum_{p \in N(q)} P(\omega \in \Omega(q) | \omega = p, \omega \in \Omega(Y)) P(\omega = p | \omega \in \Omega(Y)) \\
&= \textstyle\sum_{p \in N(q)} P(\omega \in \Omega(q) | \omega = p) P(\omega \in \Omega(p) | \omega \in \Omega(Y)) \\
&= \textstyle\sum_{p \in N(q)} \frac{W_{pq}}{\sum_{q' \in N(p)} W_{pq'}} \mathcal{R}(p)
\end{aligned} \tag{7}$$

Precision and recall of query. In symmetry, $\mathcal{P}(p)$ or $\mathcal{R}(p)$ can be expressed in terms of $\mathcal{P}(q)$ or $\mathcal{R}(q)$, $\forall q \in N(p)$. Their derivations are similar to Eq. 6–7.

$$\mathcal{P}(p) = \textstyle\sum_{q \in N(p)} \frac{W_{pq}}{\sum_{q' \in N(p)} W_{pq'}} \mathcal{P}(q) \tag{8}$$

$$\mathcal{R}(p) = \textstyle\sum_{q \in N(p)} \frac{W_{pq}}{\sum_{p' \in N(q)} W_{p'q}} \mathcal{R}(q) \tag{9}$$

Unification. Interestingly, regardless of precision or recall and whether for page or query, we can unify the above reinforcement rules (Eq. 6–9) into one equation below. $\forall v \in V$,

$$\mathcal{U}(v) = F(\{\mathcal{U}(v') | v' \in N(v)\}), \tag{10}$$

where F represents an aggregation function over the neighbors' utilities $\{\mathcal{U}(v')|v' \in N(v)\}$, which instantiates differently for precision or recall. As a result, pages (or queries) that connect to similar neighboring queries (or pages) tend to have similar utilities.

Regularization. Apart from the mutual reinforcement of utilities between neighboring vertices on the graph, the target aspect itself entails crucial clues for utility inference.

Remember that the target aspect Y is modeled by a relevance function for pages (to be materialized by a classifier in Sect. VI). For a page p, we can inject its relevance w.r.t. the target aspect—given by $Y(p)$—into the reinforcement rules to further regularize the utilities. Ideally, every relevant page should have a precision of 1, whereas all of them should share a total recall of 1. Consider Marc Snir in Fig. 2. As p_1, p_2, p_3, p_4 are the only relevant pages (which we know of), we should guide each of their precision towards 1, and each of their recall towards $\frac{1}{4}$ in the absence of other evidence so that their recalls sum up towards 1.

We name such guidance *utility regularization*, denoted by $\widehat{\mathcal{P}}$ for precision and $\widehat{\mathcal{R}}$ for recall. In the above example, we would have $\widehat{\mathcal{P}}(p_i) = 1$ and $\widehat{\mathcal{R}}(p_i) = \frac{1}{4}$ with $i \in \{1,2,3,4\}$, to ultimately regularize the estimation of utilities, $\mathcal{P}(p_i)$ and $\mathcal{R}(p_i)$, respectively. More formally, $\forall p \in P$,

$$\widehat{\mathcal{P}}(p) = Y(p), \tag{11}$$

$$\widehat{\mathcal{R}}(p) = Y(p)/\textstyle\sum_{p' \in P} Y(p'). \tag{12}$$

$\forall v \notin P$, we set $\widehat{\mathcal{P}}(v) = \widehat{\mathcal{R}}(v) = 0$ to mean that there is no regularization for v.

Finally, we incorporate utility regularization into the reinforcement rules (as summarized by Eq. 10), as follows:

$$\mathcal{U}(v) = (1-\alpha)F(\{\mathcal{U}(v')|v' \in N(v)\}) + \alpha\, \widehat{\mathcal{U}}(v), \tag{13}$$

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 $\widehat{\mathcal{R}}(v)$.

Solution. The next step is to solve the inference model in Eq. 13. For a given graph G and regularization $\widehat{\mathcal{U}}$, the goal is to find $\mathcal{U}(v)$, $\forall v \in V$, that satisfy Eq. 13.

Our inference model is equivalent to random walks with restarts [24]. In particular, it can be verified that $\mathcal{P}$ is the stationary distribution of the *backward* random walk on G, whereas $\mathcal{R}$ is the stationary distribution of the *forward* random walk on G [16], [15], [25]. In both walks, the restarting probability equals to our regularization parameter α, and the preference vector corresponds to our utility regularization $\widehat{\mathcal{U}}$. We refer readers to the given literature on the details of the random walks. Subject to graph irreducibility, each random walk has a unique stationary distribution.

The stationary distributions can be found using standard iterative updating. On a graph $G = (V, E)$, the running time is $O(|V| + |E|)$ for each iteration, and it typically converges in 50 iterations. Thus, the time cost is linear in the scale of the graph. Alternatively, there also exist numerous algorithms [26], [25], [27] to improve the efficiency, which are beyond the scope of this paper.

Entity	Example page content	Example query
Marc Snir	...many HPC papers in IJHPCA ...	`hpc ijhpca`
Philip Yu	...his data mining papers in TKDE ...	`data mining tkde`
Andrew Ng	...his recent AI paper in JMLR ...	`ai jmlr`

Fig. 3: Examples of entity variation.

IV. Domain-aware L2Q

As Sect. I discussed, an entity does not exist in isolation. There often exist many peer entities in the same domain (we call them *domain entities*). We must learn what kind of queries are useful from these domain entities, not only to bootstrap from the beginning when we know little about the target entity, but also to enhance the entire querying process.

More specifically, for a given domain we can prepare some domain entities, and further collect their pages (*domain pages*) in advance. Building upon the utility inference model in Sect. III, we address the subproblem of domain-aware L2Q, which naturally consists of two phases below.

- *Domain phase (D).* Learn what kind of queries are useful from domain pages P_D. This phase is only executed once.
- *Entity phase (E).* Choose the best query for the target aspect of the given entity, based on both current result pages P_E and "knowledge" learnt from P_D in the domain phase. This phase is executed once for every query selection (*i.e.*, for each iteration in Fig. 1).

Note that we use subscripts $_D$ and $_E$ to differentiate the corresponding notions in the two phases. Accordingly, we concretize Eq. 1 to formalize domain-aware L2Q, as follows.

$$q^* = \arg\max\nolimits_{q \in Q_E} \mathcal{U}_E(q|P_E, P_D), \tag{14}$$

where $\mathcal{U}_E(q|P_E, P_D)$ denotes that P_E, P_D are given as input, which we will henceforth omit to only write $\mathcal{U}_E(q)$ for brevity.

However, realizing the two phases is non-trivial due to inherent differences between them. We will first "bridge their gap" in Sect. IV-A, before discussing the two phases in Sect. IV-B and IV-C, respectively.

A. Bridging domain and entity phases

The major challenge of domain-aware L2Q arises from *entity variations*. Even in the same domain, different entities are often described by varying words. As Fig. 3 illustrates, `hpc ijhpca` is a useful query for Snir, but not for Yu or Ng. Thus, directly learning useful queries from domain entities is likely to fail, since the learnt queries may not align well with the target entity.

Templates. To address such variations, we observe that queries for different entities often match similar abstractions. For instance, all the three queries in Fig. 3 can be abstracted as "⟨topic⟩ ⟨journal⟩," where ⟨∗⟩ represents a common *type* in the domain. A type is just a set of words, such as ⟨topic⟩ = {`hpc`, `data mining`, `ai`, ...} and ⟨journal⟩ = {`ijhpca`, `tkde`, `jmlr`, ...}, which are often available in existing knowledge bases like Freebase. Alternatively, certain types like ⟨person⟩ and ⟨location⟩ can be recognized by NLP tools, and others like ⟨phonenum⟩ and ⟨email⟩ by regular expressions.

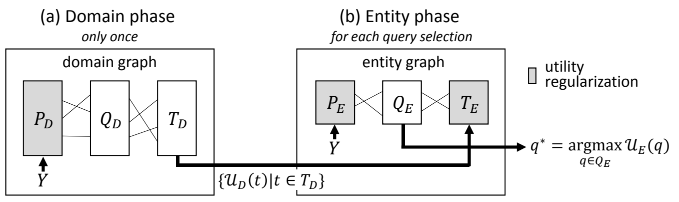


Fig. 4: Pipeline of domain-aware L2Q.

(a) Example templates

Query	Template
q_1: `parallel research`	t_1: ⟨topic⟩ `research`
q_2: `hpc research`	t_1: ⟨topic⟩ `research`
q_3: `complexity`	t_2: ⟨topic⟩
q_4: `u illinois`	t_3: ⟨institute⟩
q_5: `ibm`	t_3: ⟨institute⟩

(b) Reinforcement graph

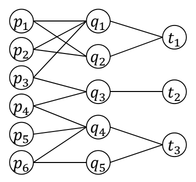


Fig. 5: Running illustration extended with templates.

We name such query abstractions *templates*, and use them to bridge different entities in the same domain. For instance, from Ng we learn that `ai jmlr` is useful, so is the template "⟨topic⟩ ⟨journal⟩" given that `ai` $\in$ ⟨topic⟩ and `jmlr` $\in$ ⟨journal⟩. Since Snir's query `hpc ijphca` can be abstracted by the same template, it is also likely to be useful, enabling us to generalize from Ng. Different from previous concepts of templates or patterns [16], [13], [28], [15], in this work we devise templates for a novel purpose—through templates as the medium, we can capture domain "knowledge" across entities consistently, to ultimately benefit future target entities. Formally, we define a template as follows.

DEFINITION 1 (TEMPLATE): Consider a universe of words $\mathcal{W}$, and a set of types $\mathcal{C} = \{C_1, C_2, \ldots, C_{|\mathcal{C}|}\}$ where each type contains a subset of words, *i.e.*, $C_i \subset \mathcal{W}$. Then, for a given *maximum query length* L, and some $\ell \leq L$,

- a *query* is a sequence of words $q = (w_1, w_2, \ldots, w_\ell)$ such that each word $w_i \in \mathcal{W}$;
- a *template* is a sequence of units $t = (u_1, u_2, \ldots, u_\ell)$ such that each unit u_i is either a word $w \in \mathcal{W}$ or a type $C \in \mathcal{C}$;
- We say that t can abstract q if $w_i = u_i$ when u_i is a word, or $w_i \in u_i$ when u_i is a type, $\forall i \in \{1, \ldots, \ell\}$. □

Bridging two phases. Given templates as the medium to generalize domain entities, how exactly do we use them to bridge the two phases? Note that templates cannot be directly used by a search engine given their abstraction. Ultimately, we must still "translate" templates to concrete queries.

As discussed earlier, due to entity variations we cannot directly take the utilities of the domain queries to the target entity. Thus, the utility inference model must incorporate templates—we infer the utilities for the templates in the domain phase, and then use them to help us understand what queries are likely to be useful in the entity phase. Technically, mutual reinforcement also exists between templates and queries—useful templates often lead to useful queries and vice versa. In other words, utilities can be extended to templates, and we need to estimate $\mathcal{U}(t)$ for each template t too.

The mutual reinforcement between pages, queries and templates exists in both phases, as summarized in Fig. 4. In the domain phase, we induce useful queries Q_D from domain pages P_D, and further generalize them into templates T_D. In the entity phase, we adopt those templates T_E that also appear with the target entity to match target queries Q_E, and further regularize them with target pages P_E. In each phase, there is utility regularization for pages w.r.t. Y, as discussed in Sect. III. Moreover, the utilities of templates from the domain phase, $\{\mathcal{U}_D(t) | t \in T_D\}$, provide additional regularization for the target entity, as we will elaborate in the entity phase.

Utilities of templates. Consider a template universe T, which can be enumerated from queries with a given set of types. Based on the running example in Fig. 2, we obtain $T = \{t_1, t_2, t_3\}$ in Fig. 5(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. 5(b).

Following Sect. III, let $\Omega(t)$ denote 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 $\Omega(t_1) = \{p_1, p_2, p_3\}$. The utilities of t, $\mathcal{P}(t)$ and $\mathcal{R}(t)$, can then be defined by Eq. 4 and 5 respectively, in a uniform manner as those of pages and queries. Similar to the mutual reinforcement between P and Q, we can model that between Q and T. We summarize the rules between Q and T below, whose derivations mirror those in Sect. III. As each query is now connected to both pages and templates, we denote its page neighbors by $NP(q)$ and template neighbors by $NT(q)$.

$$\mathcal{P}(t) = \textstyle\sum_{q \in N(t)} \frac{W_{qt}}{\sum_{q' \in N(t)} W_{q't}} \mathcal{P}(q) \tag{15}$$

$$\mathcal{R}(t) = \textstyle\sum_{q \in N(t)} \frac{W_{qt}}{\sum_{t' \in NT(q)} W_{qt'}} \mathcal{R}(q) \tag{16}$$

$$\mathcal{P}(q) = \textstyle\sum_{t \in N(q)} \frac{W_{qt}}{\sum_{t' \in NT(q)} W_{qt'}} \mathcal{P}(t) \tag{17}$$

$$\mathcal{R}(q) = \textstyle\sum_{t \in N(q)} \frac{W_{qt}}{\sum_{q' \in N(t)} W_{q't}} \mathcal{R}(t) \tag{18}$$

As $\mathcal{U}(q)$ can be expressed using either its neighboring pages (Eq. 6–7) or templates (Eq. 17–18), we combine both sides by taking their average as the final utility of q. Essentially, here we only consider a balanced influence from pages and from templates; investigating other combinations is left to future work. The resulting inference model with templates can still be solved using the same random walks discussed in Sect. III.

B. Domain phase

In the domain phase, we generalize from domain entities, to learn templates for future target entities, as illustrated in Fig. 4(a). Concretely, we need to specify a reinforcement graph G_D for the domain entities (*domain graph*), as well as utility regularization $\widehat{\mathcal{U}}_D$ on this graph. As output, we infer the utilities $\mathcal{U}_D$, in particular those of templates. That is, we apply the inference rules in Eq. 13 on G_D. More explicitly, for any vertex v on G_D, and letting N_D denote the neighbor function of G_D, we need to solve

$$\mathcal{U}_D(v) = (1-\alpha) F(\{\mathcal{U}_D(v') | v' \in N_D(v)\}) + \alpha\, \widehat{\mathcal{U}}_D(v). \tag{19}$$

(a) Example pages $P_D = \{p_7, p_8, p_9\}$ (Y is RESEARCH)

	Content	$Y(p_i)$	$\widehat{\mathcal{P}}_D(p_i)$	$\widehat{\mathcal{R}}_D(p_i)$
p_7	...his AI research projects at Baidu ...	1	1	1/2
p_8	...published many AI papers at Stanford ...	1	1	1/2
p_9	...taught at Stanford ...	0	0	0

(b) Domain graph

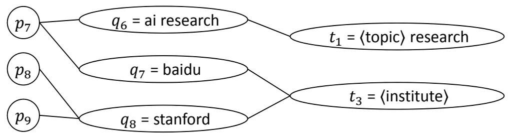


Fig. 6: Illustration of Andrew Ng as a domain entity.

Domain graph. Take Andrew Ng as an example for domain entity, with three pages $P_D = \{p_7, p_8, p_9\}$ shown in Fig. 6(a). Queries $Q_D = \{q_6, q_7, q_8\}$ and templates $T_D = \{t_1, t_3\}$ can be enumerated from the pages, to form a domain graph in Fig. 6(b). In general, given some domain pages P_D, we can enumerate the set of queries Q_D contained in them. (More details of enumerating the queries will be covered in Sect. VI.) Furthermore, from these queries we can obtain templates T_D based on a given set of types. Although the domain queries (from Ng) look different from target queries (from Snir in Fig. 5), they share some templates (t_1 and t_3).

Utility regularization. For each domain page $p_i \in P_D$, we derive its utility regularization $\widehat{\mathcal{U}}_D(p_i)$ as discussed in Sect. III (Eq. 11–12). Using Ng as an example, we can obtain $\widehat{\mathcal{P}}_D(p_i)$ and $\widehat{\mathcal{R}}_D(p_i)$ based on $Y(p_i)$, as presented in Fig. 6(a).

Output: templates. By solving the inference rules on the domain graph (Eq. 19), we can output the templates and their utilities $\{\mathcal{U}_D(t) | t \in T_D\}$. In our example for Ng, the precision model will give $\mathcal{P}_D(t_1) > \mathcal{P}_D(t_3)$ since t_3 covers an irrelevant page p_9, whereas the recall model will give $\mathcal{R}_D(t_1) < \mathcal{R}_D(t_3)$ as t_1 fails to cover a relevant page p_8.

C. Entity phase

In the entity phase, we leverage templates learnt from the domain phase, as summarized in Fig. 4(b). Parallel to Sect. IV-B, we need a reinforcement graph G_E for the target entity (*entity graph*) and utility regularization $\widehat{\mathcal{U}}_E$ on this graph. Unlike the domain phase, templates will provide additional utility regularization, together with current pages of the target entity. We then select the best query according to the inferred utilities $\mathcal{U}_E$. Essentially, we apply the inference rules in Eq. 13 on G_E. For any vertex v on G_E, and letting N_E denote the neighbor function of G_E, we need to solve

$$\mathcal{U}_E(v) = (1 - \alpha) F(\{\mathcal{U}_E(v') | v' \in N_E(v)\}) + \alpha\, \widehat{\mathcal{U}}_E(v). \quad (20)$$

Entity graph. Similar to the domain phase, from the current result pages P_E, we can enumerate a set of queries as candidate queries. However, it is rather limited to only consider queries from current pages as the candidates—many potentially useful queries are unforeseeable due to the incompleteness of P_E, as the gathering process is still ongoing. To overcome this limitation, we also consider queries from the domain entities. (For efficiency consideration, we restrict to queries that occur with at least 50 domain entities.) That is, the candidate query set Q_E contains queries enumerated from both P_E and P_D. Subsequently, we enumerate the templates T_E from Q_E, and obtain the corresponding entity graph.

Utility regularization. As shown in Fig. 4, in addition to utility regularization on pages, we can leverage templates learnt in the domain phase to further regularize the target entity. On the one hand, regularization on current pages P_E only applies to the target entity itself, since other entities have different pages. On the other hand, regularization on templates T_E can be applied to other entities in the same domain, since templates are consistent across them. By using both forms of regularization, we can identify useful queries not only specific to the target entity, but also general in the domain. As with the domain phase, we encode regularization $\widehat{\mathcal{U}}_E(p)$ for each entity page $p \in P_E$ based on $Y(p)$. For each template, we use its utility learnt from the domain phase as our regularization. Using Snir as the target and Ng as the domain entity, Snir will have utility regularization for t_1 and t_3, since their utilities are learnt from Ng in the domain phase (Sect. IV-B). Formally, $\forall t \in T_E \cap T_D$,

$$\widehat{\mathcal{P}}_E(t) = \lambda\ \mathcal{P}_D(t), \quad (21)$$

$$\widehat{\mathcal{R}}_E(t) = \lambda\ \mathcal{R}_D(t), \quad (22)$$

where $\lambda > 0$ is an *adaptation* parameter to control how much we adapt from the domain entities. In other words, λ adjusts the trade-off between matching the domain templates and the entity pages.

Output: best query. By solving the inference rules on the entity graph (Eq. 20), we obtain the utility $\mathcal{U}_E(q)$ for every candidate query $q \in Q_E$. Finally, we output the query with maximum utility.

V. CONTEXT-AWARE L2Q

As Sect. I motivated, a query does not exist in isolation. When assessing the candidate queries in iteration-i, there is a context of past queries that were already fired, which we denote as $\Phi \triangleq \{q^{(0)}, q^{(1)}, \ldots, q^{(i-1)}\}$.

Due to the redundancy between queries, measuring the utility of each candidate query without accounting for the context queries is hardly promising. Consider Marc Snir in Fig. 2 as the target entity, and suppose the context queries are $\Phi = \{q_1, q_5\}$. We illustrate in Fig. 7 the effectiveness of each candidate query q on its own, as well as the overall effectiveness of q and the context Φ together. For instance, given $\Omega(Y) = \{p_1, p_2, p_3, p_4\}$, q_2 (retrieving p_1, p_2) and q_3 (retrieving p_3, p_4) are equally effective on their own. However, together with Φ, q_3 (retrieving p_1, p_2, p_3, p_4, p_6) is superior to q_2 (retrieving p_1, p_2, p_3, p_6), since q_2 fails to retrieve any page beyond those of Φ. In other words, an individual good choice is not necessarily an overall good choice, when we are given the context of past queries.

Thus, we address the subproblem of context-aware L2Q: selecting the best query based on the utility of each candidate query in conjunction with the context. We can further extend

candidate q	own prec.	own recall	overall prec. w/ Φ	overall recall w/ Φ
if $q = q_2$	$2/2 = 1$	$2/4 = 0.5$	$3/4 = 0.75$	$3/4 = 0.75$
if $q = q_3$	$2/2 = 1$	$2/4 = 0.5$	$4/5 = 0.8$	$4/4 = 1$
if $q = q_4$	$1/3 = 0.33$	$1/4 = 0.25$	$4/6 = 0.67$	$4/4 = 1$

Fig. 7: Effectiveness of Marc Snir's queries with $\Phi = \{q_1, q_5\}$.

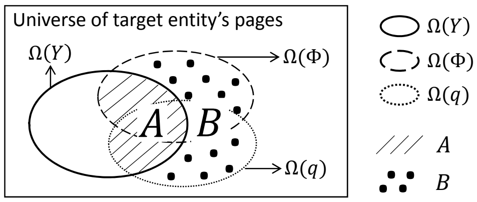


Fig. 8: Venn diagram to illustrate collective utilities.

Eq. 14 as follows. (As usual, we will omit P_E, P_D from the utility notation for brevity.)

$$q^* = \arg\max_{q \in Q_E} \mathcal{U}_E(\Phi \cup \{q\}|P_E, P_D). \tag{23}$$

In other words, we measure the *collective* utility for a set of queries consisting of both the candidate query q and the context queries Φ. In the above example, we should choose $q^* = q_3$ to maximize collective precision, and $q^* = q_3$ or q_4 to maximize collective recall.

Parallel to the probabilistic utilities of one query (Eq. 4–5), we define below the collective utilities of a set of queries probabilistically, where $\Omega(Q) \equiv \bigcup_{q \in Q} \Omega(q)$.

$$\mathcal{P}_E(\Phi \cup \{q\}) \triangleq P(\omega \in \Omega(Y) \mid \omega \in \Omega(\Phi \cup \{q\})), \tag{24}$$
$$\mathcal{R}_E(\Phi \cup \{q\}) \triangleq P(\omega \in \Omega(\Phi \cup \{q\}) \mid \omega \in \Omega(Y)). \tag{25}$$

The main challenge now is to realize the collective utilities. As it will turn out, we can estimate collective precision based on collective recall. Thus, we begin with collective recall.

A. Collective recall

As our major insight, $\mathcal{R}_E(\Phi \cup \{q\})$ can be decomposed into simpler components, including $\mathcal{R}_E(q)$ and $\mathcal{R}_E(\Phi)$, as well as the redundancy between q and Φ. Such a decomposition is desirable, since we have already established the inference of $\mathcal{R}_E(q)$ in Sect. IV, and we can compute $\mathcal{R}_E(\Phi)$ recursively using the same decomposition. Thus, to compute collective recall, we only have to focus on solving the redundancy between q and Φ.

Consider the Venn diagram in Fig. 8. We focus on three subsets that define collective utilities: i) $\Omega(\Phi)$, pages retrieved by any query in the context Φ; ii) $\Omega(q)$, pages retrieved by a candidate query q; iii) $\Omega(Y)$, the relevant pages. These subsets interact with each other, forming subset A consisting of relevant pages by Φ and q collectively, and subset B consisting of irrelevant pages by them collectively. In particular, A can be aggregated from relevant pages of Φ and those of q, if we only count their overlap (relevant pages of both Φ and q) once. Correspondingly, $\mathcal{R}_E(\Phi \cup \{q\})$ is the sum of $\mathcal{R}_E(\Phi)$ and $\mathcal{R}_E(q)$ if we further subtract their "overlap."

Rewriting. The above intuition can be formalized below. Line 2 is due to the inclusion–exclusion principle, to separate q and Φ so that we can reuse their recalls. The expression matches our insight, since the first term is simply $\mathcal{R}_E(\Phi)$, the second term is $\mathcal{R}_E(q)$, and the last term captures the redundancy between q and Φ—the probability that a relevant page can be retrieved by both of them, which we denote by $\Delta(\Phi, q)$.

$$\begin{aligned}
&\mathcal{R}_E(\Phi \cup \{q\}) \triangleq P(\omega \in \Omega(\Phi \cup \{q\}) \mid \omega \in \Omega(Y)) \\
&\overset{1}{=} P(\omega \in \Omega(\Phi) \vee \omega \in \Omega(q) \mid \omega \in \Omega(Y)) \\
&\overset{2}{=} P(\omega \in \Omega(\Phi) \mid \omega \in \Omega(Y)) + P(\omega \in \Omega(q) \mid \omega \in \Omega(Y)) \\
&\quad - P(\omega \in \Omega(\Phi), \omega \in \Omega(q) \mid \omega \in \Omega(Y)) \\
&\overset{3}{=} \mathcal{R}_E(\Phi) + \mathcal{R}_E(q) - \Delta(\Phi, q).
\end{aligned} \tag{26}$$

Estimation. We only need to estimate the three terms in Eq. 26. Among them, $\mathcal{R}_E(q)$ is already available from Sect. IV.

Moreover, $\mathcal{R}_E(\Phi)$ can be recursively computed by further 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 $r_0 \in (0, 1)$, which we call *seed query* parameter, to be chosen by cross validation in our experiments.

Lastly, we estimate $\Delta(\Phi, q)$, the redundancy between Φ and q, in the following. Line 1 is a result of Bayes' rule. Since $\Omega(\Phi) \equiv P_E$, we have $\Omega(\Phi) \cap \Omega(Y) \equiv \{p | Y(p) = 1, p \in P_E\}$, *i.e.*, the set of relevant pages among the current result pages. This set can be equivalently rewritten as $\Omega(\widetilde{Y})$ such that $\widetilde{Y}(p) = 1$ iff $Y(p) = 1$ and $p \in P_E$. Subsequently, we arrive at line 2. By the definitions of recall and collective recall, we can further rewrite it into line 3. While $\mathcal{R}_E(\Phi)$ can be computed recursively as discussed earlier, $\mathcal{R}_E^{(\widetilde{Y})}(q)$ can be found using the inference rules for recall, with regularization $\widehat{\mathcal{R}}_E^{(\widetilde{Y})}(p) = \widetilde{Y}(p) / \sum_{p' \in P_E} \widetilde{Y}(p'), \forall p \in P_E$.

$$\begin{aligned}
&\Delta(\Phi, q) = P(\omega \in \Omega(\Phi), \omega \in \Omega(q) \mid \omega \in \Omega(Y)) \\
&\overset{1}{=} P(\omega \in \Omega(q) \mid \omega \in \Omega(\Phi) \cap \Omega(Y)) P(\omega \in \Omega(\Phi) | \omega \in \Omega(Y)) \\
&\overset{2}{=} P(\omega \in \Omega(q) \mid \omega \in \Omega(\widetilde{Y})) P(\omega \in \Omega(\Phi) | \omega \in \Omega(Y)) \\
&\overset{3}{=} \mathcal{R}_E^{(\widetilde{Y})}(q) \cdot \mathcal{R}_E(\Phi)
\end{aligned}$$

B. Collective precision

To optimize collective precision, Φ and q should collectively retrieve as many relevant pages (*i.e.*, A in the Venn diagram), but at the same time as few total pages (*i.e.*, $A \cup B$). As our major insight, collective precision can be decomposed into two components that correspond to A and $A \cup B$, respectively. In particular, $|A|$ is the coverage of relevant pages by Φ and q collectively, and $|A \cup B|$ is their coverage of all pages collectively, both of which intuitively correspond to some form of collective recall. Thus, we can compute collective precision by way of collective recall.

Rewriting. The above intuition can be formalized below. Bayes' rule is applied to obtain line 1. Note that $P(\omega \in \Omega(Y))$, the prior of relevant pages, is a constant for any query and thus does not affect query selection. Furthermore, $P(\omega \in \Omega(\Phi \cup \{q\})) \equiv P(\omega \in \Omega(\Phi \cup \{q\}) \mid \omega \in \Omega(Y^*))$ such that $Y^*(p) = 1$ for any page. Therefore, we arrive at line 2, which matches our insight. Specifically, the numerator is simply $\mathcal{R}_E(\Phi \cup \{q\})$ by definition, which is proportional to $|A|$; the denominator is just $\mathcal{R}_E^{(Y^*)}(\Phi \cup \{q\})$ by definition, which is proportional to $|A \cup B|$.

$$\begin{aligned}
&\mathcal{P}_E(\Phi \cup \{q\}) \triangleq P(\omega \in \Omega(Y) \mid \omega \in \Omega(\Phi \cup \{q\})) \\
&\overset{1}{=} \frac{P(\omega \in \Omega(\Phi \cup \{q\}) \mid \omega \in \Omega(Y))}{P(\omega \in \Omega(\Phi \cup \{q\}))} P(\omega \in \Omega(Y)) \\
&\overset{2}{\propto} \frac{P(\omega \in \Omega(\Phi \cup \{q\}) \mid \omega \in \Omega(Y))}{P(\omega \in \Omega(\Phi \cup \{q\})) | \omega \in \Omega(Y^*))} \\
&\overset{3}{=} \frac{\mathcal{R}_E(\Phi \cup \{q\})}{\mathcal{R}_E^{(Y^*)}(\Phi \cup \{q\})}
\end{aligned} \tag{27}$$

Estimation. To estimate Eq. 27, we only need to compute the collective recalls w.r.t. Y and Y^*. For Y, we follow the discussion in Sect. V-A. For Y^*, the only difference is that every page is treated as "relevant," and we simply derive utility regularization based on $Y^*(p)$ instead of $Y(p)$, without any other change.

VI. Experiments

We empirically evaluate the proposed L2Q approach. We adopt the overall workflow in Fig. 1 to harvest pages iteratively. In each iteration, the task boils down to the core problem of L2Q, where we select the best query to gather pages focused on the target entity aspect via a search engine. After each iteration, we evaluate the cumulative quality of pages gathered thus far.

We start with experimental setup in Sect. VI-A. Next, we showcase the significance of the two subproblems, domain and context-aware L2Q, in Sect. VI-B. Lastly, we compare the performance of our "full" L2Q approach with various baselines in Sect. VI-C.

A. Experimental setup

Corpora. 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 two domains: *researchers* and *cars*. In total, we have 996 researchers randomly chosen from DBLP's most prolific authors[1] and 143 consumer car models released in 2009. For each entity, we attempted to collect 50 pages from the Web to construct the corpora. This number is generally adequate to cover pages of diverse aspects for each entity, since the goal of this paper is to achieved focused harvesting of entity aspects. To retrieve pages from the corpora, we used a language model with Dirichlet smoothing [29] as the search engine. For each query, pages in the corpus are ranked and the top 5 are returned. As such, we can measure the performance of our approach against an ideal but otherwise infeasible solution (to be explained in evaluation methodology).

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

Researchers			Cars		
Aspects	Frequency	Accuracy	Aspects	Frequency	Accuracy
BIOGRAPHY	8K	0.99	VERDICT	7K	0.91
PRESENTATION	10K	0.99	INTERIOR	7K	0.96
AWARD	11K	0.97	EXTERIOR	5K	0.97
RESEARCH	107K	0.85	PRICE	8K	0.92
EDUCATION	11K	0.99	RELIABILITY	2K	0.97
EMPLOYMENT	3K	0.94	SAFETY	2K	0.99
CONTACT	7K	0.97	DRIVING	16K	0.89

Fig. 9: Tested entity aspects and accuracy of aspect classifiers.

Templates. To construct the types for enumerating templates, we resorted to three options. First, we constructed a dictionary to map a keyword or phrase to a type based on Freebase[2] and Microsoft Academic Search (MAS)[3], such as mapping `data mining` to ⟨topic⟩. Freebase is an open and general database containing a huge number of types across many different domains. Thus, we can rely on it to devise templates for many real-world applications. In addition, we used MAS to supplement types for the researcher domain. Second, we relied on Standford CoreNLP [30] to recognize the named entities as types, such as ⟨organization⟩, ⟨person⟩ and ⟨location⟩. Third, we devised regular expressions to tag well-formed texts, such as ⟨phonenum⟩, ⟨url⟩ and ⟨email⟩.

Entity aspects. We tested seven target aspects (Y's) on each domain as shown in Fig. 9. To enable a finer granularity of evaluation, we segmented each page into paragraphs[4], and we looked at the relevance of each paragraph w.r.t. each given Y. (Note that query selection is orthogonal to the retrieval units used.) Specifically, we trained one classifier for each Y based on conditional random fields, which can classify a paragraph as relevant to Y or not. Our aspect classifiers can achieve a high level of accuracy on both domains as shown in Fig. 9, and thus their output is taken as the ground truth.

Candidate query enumeration. To enumerate candidate queries from a page, we first tokenize the page into words. Each word is either a single keyword, or a phrase (*e.g.*, `data mining`) that can be mapped to a type. Subsequently, we applied a sliding window of ℓ words over the page for each $\ell \in \{1, 2, \ldots, L\}$. Note that L is the maximum query length in Definition 1. The ℓ words in each window are taken as a candidate query. In particular, we set $L = 3$, *i.e.*, only queries containing 1, 2 or 3 words are enumerated from pages. This maximum length is generally sufficient to capture various semantic units, as choosing a larger value for L results in no notable improvement in effectiveness, but increases the query space and thus computational cost.

Evaluation methodology. In each domain, we randomly reserved half of the entities as domain entities, and the remaining as target entities. One of our experiments also varied the number of domain entities used to showcase the effect of domain size. Target entities were further divided into two equal splits, such that one of the split is reserved for parameter validation, and the other for testing. We repeated the split randomly for 10 times.

[2]https://www.freebase.com/

[3]http://academic.research.microsoft.com/

[4]Segmentation was done using jsoup from http://www.jsoup.org/.

On the testing set, we evaluate the retrieved pages in terms of their actual precision and recall (and eventually F-score) for every target entity and aspect. We then normalize the results against an *ideal* solution, for two reasons. Firstly, as different entities entail varying degrees of difficulty, normalization makes results comparable across entities. Secondly, the ideal solution, which supposedly achieves a performance upper bound, enables us to perceive the actual gap with the best "achievable" performance. Note that, for each entity, the same normalization factor is applied to all methods (*i.e.*, our approach and the baselines) without biasing towards any method—a better method is still better after normalization.

We thus design an ideal solution as the upper bound to measure against. Ideally, we would select queries that cover as many relevant pages as possible, and achieve a high precision over the covered pages. We then select queries to maximize the product of their *actual* coverage and precision, which can be obtained by feeding each candidate query to the search engine. Thus, it is clearly infeasible in real applications, and only acts as a performance upper bound for normalization.

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.

Settings. In most experiments, we varied the number of queries (*i.e.*, number of iterations in Fig. 1) between 2 and 5, excluding the initial seed query. When we do not mention the number of queries used, the default is 3. We selected the seed query parameter r_0 in Sect. V-A by cross validating on the validation set. We set the regularization parameter to $\alpha = 0.15$ in Eq. 13, which is a typical value robust to random walks on most graphs. Lastly, we set the adaption parameter to $\lambda = 10$ (Sect. IV-C), which turns out to be generally effective and stable in our experiments.

B. Effect of domain and context awareness

We first validate the significance of the two subproblems, namely, domain-aware and context-aware L2Q. Thus, we devised and compared several strategies below, in order to highlight the usefulness of the domain entities and the context queries. As we proposed two measures of utility, namely precision $\mathcal{P}$ and recall $\mathcal{R}$, each strategy (except the random method RND) has two variants: one optimizes $\mathcal{P}$ and the other optimizes $\mathcal{R}$. We then evaluate each variant using the metric that it intends to optimize.

- RND, which randomly selects a query from all the candidates, to establish a reference point for other strategies.
- P or R, which optimizes precision or recall as Sect. III discussed, *i.e.*, without domain and context-awareness.
- P+q or R+q, which directly uses *queries* (+q) of best precision or recall learnt from the domain phase, to show the problem of entity variations.
- P+t or R+t, which optimizes precision or recall with *template*-based learning (+t), but without context-awareness.
- L2QP or L2QR, which optimizes precision or recall with both domain and context-awareness (*i.e.*, the full approach).

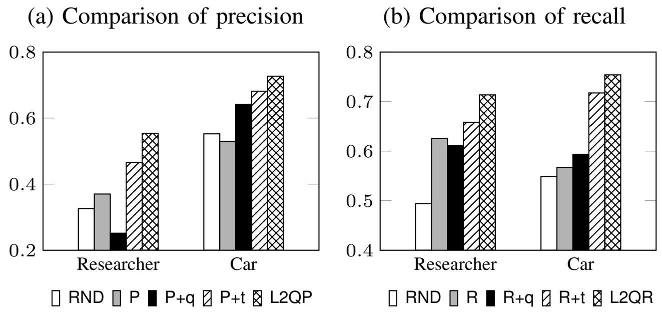


Fig. 10: Validation of domain and context awareness.

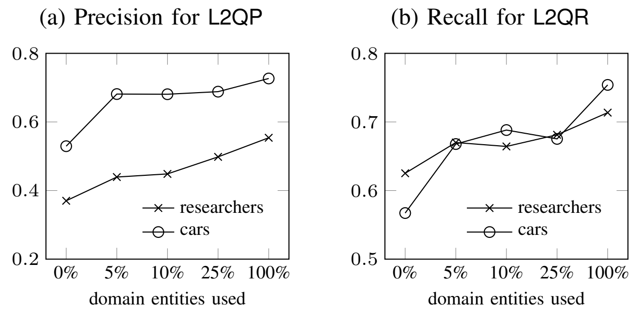


Fig. 11: Effect of domain size on full approaches.

Note that, in Sect. VI-C, we will eventually combine the two variants and evaluate the combined method.

Domain awareness. The results in Fig. 10 demonstrate that, while domain entities can be beneficial to L2Q, we must employ templates due to entity variations. Evidently, P+t is superior to P in precision, given that P+t learn from domain entities with templates and P does not utilize the domain at all. Similarly, R+t outperforms R in recall for the same reason. However, if we do not abstract queries into templates, we cannot leverage the domain to the fullest extent. In particular, P+q directly use queries with best precision among domain entities, but its performance is worse than the template-based P+t. We observe a similar outcome between R+q and R+t in recall. That means, entity variations do exist and it is necessary to adopt templates. Naturally, when entity variations are more severe, P+q and R+q could even become worse than P and R, as in the case of researchers.

We also investigate the effect of domain size. As reported in Fig. 11, we use between 0% and 100% domain entities. In general, using more domain entities results in better precision for L2QP and better recall for L2QR. The improvement is especially substantial when we increase from 0% to 5%, meaning that even a small number of domain entities can be quite useful.

Context awareness. The results also show that it is important to account for the past queries through collective utilities. In Fig. 10, our full approach L2QP outperforms P+t in precision. Note that both L2QP and P+t learn from domain entities through templates, but L2QP is aware of the context of past queries whereas P+t is not. Likewise, L2QR is superior to R+t in terms of recall.

C. Comparison with baselines

We further compare our approaches L2QP, L2QR, and their combination (to obtain a balance between precision and recall) with four independent baselines below. 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.

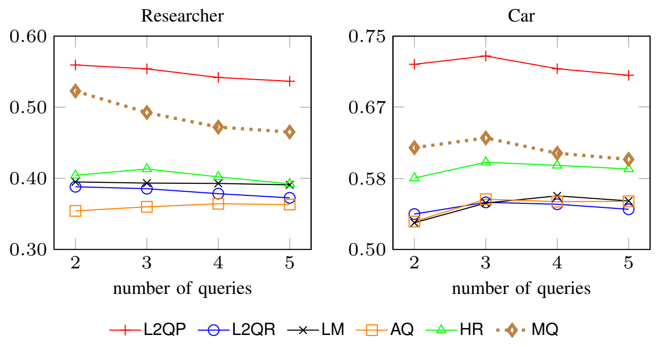
- *Language Model* (LM), based on the language feedback model [22]. In each iteration, it chooses the query with maximum likelihood on the k most relevant current pages. In particular, we use $k = 1$, which results in the best performance on our corpora.
- *Adaptive Querying* (AQ), based on the adaptive query selection policy [5]. It was designed to crawl text databases, using query statistics adaptive to the current results. As it lacks the notion of relevance, to adopt it for our purpose, the query statistics are only computed over relevant pages instead of all pages.
- *Harvest Rate* (HR), based on the harvest rate heuristic [2]. It was originally meant for crawling structured databases, using query statistics from current results and domain data. We first modify its query and record model as a bag of words, and incorporate the notion of relevance just as we did for AQ. We then apply templates: the statistics of each query is computed as the average over its templates. (We only use templates in HR but not the others, since only HR exploits domain data.)
- *Manual Querying* (MQ), based on human designed queries. For each domain and aspect, we asked nine graduate students to provide five queries that they would use to search for the target entity aspect. For instance, for researchers' AWARD aspect, sample queries given by the user study include `award`, `distinguished`, `award won`, and so on. We observed that there is generally good inter-user agreement, and thus only report the average performance of the users. Note that manually designing queries that are specific to each entity does not scale up, and thus is not a viable baseline in real-world applications.

Precision and recall. We first compare L2QP and L2QR to the baselines in terms of precision and recall, with varying number of queries (*i.e.*, iterations). The results are reported in Fig. 12.

In terms of precision as Fig. 12(a) shows, L2QP achieves the best performance, surpassing not only the baselines, but also L2QR, since L2QP is designed to optimize precision. This observation is consistent in both domains across different number of queries. On average, L2QP outperforms the best algorithmic baselines by 28%, and beats the manual baseline by 14%. Note that, as we use more queries, L2QP and MQ tend to suffer a minor decrease in precision since there only exist a limited number of pages relevant to the target aspect of each entity.

In terms of recall as Fig. 12(b) shows, L2QR likewise outperforms all the other methods, as it is designed to optimize recall. On average, L2QR outperforms the best algorithmic baseline by 11%, and beats the manual baseline by 14%.

(a) Comparison of precision



(b) Comparison of recall

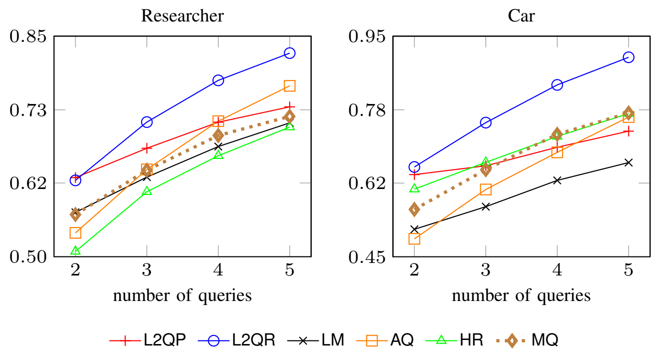


Fig. 12: Comparison of precision and recall.

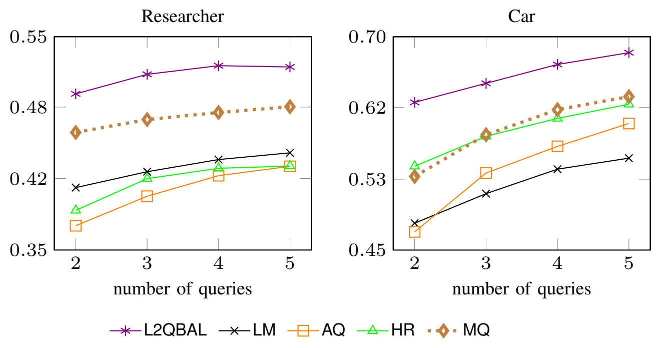


Fig. 13: Comparison of F-scores with balanced strategy.

Combining L2QP and L2QR. Although L2QP and L2QR perform the best in their corresponding utilities, we often seek a trade-off in the real world, which is usually measured by F-score. Therefore, we combined L2QP and L2QR, resulting in a new query selection method L2QBAL that balances precision and recall. Specifically, we select queries based on the geometric mean of the collective precision and recall. We do not optimize the harmonic mean given that our utilities are probabilistic in nature, which means the estimated precision and recall have incomparable scales.

We verified that L2QBAL indeed outperforms L2QP and L2QR in F-score. Subsequently, we compare L2QBAL with the baselines in terms of F-score. As shown in Fig. 13, L2QBAL is consistently better than all the baselines for various number of queries. On average, it beats the best algorithmic baseline

Domain	Selection			Fetch
	L2QP	L2QR	L2QBAL	
Researcher	2.1	1.5	2.4	~ 18
Car	1.9	1.4	2.2	~ 8

Fig. 14: Average time cost per query (seconds).

by 16%, and exceeds the manual baseline by 10%. In other words, it is feasible to combine L2QP and L2QR. As L2QBAL is only a trivial combination, we expect that a more thorough and principled approach would give even better results, which is left to future work.

Efficiency. Finally, we investigate the efficiency of the entity phase. Note that the efficiency of the domain phase is not of primary concern, as it is only executed once. We consider the selection time (for deciding which query) and fetch time (for downloading pages from remote servers), in order to comprehend the former in the overall workflow of page gathering. Selection is CPU bound and varies for different methods, whereas fetch is I/O bound and is consistent for different methods.

For each of our methods, we report in Fig. 14 the time spent per query on a machine with a 2.2GHz CPU, utilizing only a single thread. In general, our methods only require 1–2 seconds to select a query, and the entity phase is dominated by the fetch operation. That is, our methods only impose a minor overhead over the fetch time. In real-world applications, our selections are fast enough for online processing, and they can be further improved by various techniques, such as parallelizing over entities, and interleaving the selection (CPU) and fetch (I/O) operations between different entities.

VII. Conclusion

In this paper, we studied the task of learning to query for gathering Web pages focused on an entity aspect of interest. To intelligently select queries, we quantify the usefulness of each candidate query based on an utility inference model. Furthermore, we recognize the importance of two subproblems. Firstly, we must learn from domain entities through templates, which enable domain-aware L2Q. Secondly, we must consider the context of past queries through collective utilities, which enable context-aware L2Q. We empirically evaluated our approach on two domains, and showed that it outperformed both algorithmic and manual baselines significantly.

References

[1] R. Qian, "Understand your world with Bing," Accessed: 9 Feb 2015. http://blogs.bing.com/search/2013/03/21/understand-your-world-with-bing/.

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

[3] W. W. Cohen and Y. Singer, "Learning to query the Web," in *AAAI Workshop on Internet-Based Information Systems*, 1996, pp. 16–25.

[4] E. Agichtein and L. Gravano, "Querying text databases for efficient information extraction," in *ICDE*, 2003, pp. 113–124.

[5] P. Zerfos, J. Cho, and A. Ntoulas, "Downloading textual hidden web content through keyword queries," in *Digital Libraries, 2005*, 2005, pp. 100–109.

[6] J. Hoffart, M. A. Yosef, I. Bordino, H. Fürstenau, M. Pinkal, M. Spaniol, B. Taneva, S. Thater, and G. Weikum, "Robust disambiguation of named entities in text," in *EMNLP*, 2011, pp. 782–792.

[7] J. Cho, H. Garcia-Molina, and L. Page, "Efficient crawling through URL ordering," in *WWW*, 1998, pp. 161–172.

[8] M. Diligenti, F. Coetzee, S. Lawrence, C. L. Giles, and M. Gori, "Focused crawling using context graphs," in *VLDB*, 2000, pp. 527–534.

[9] S. Pandey and C. Olston, "Crawl ordering by search impact," in *WSDM*, 2008, pp. 3–14.

[10] C. Sheng, N. Zhang, Y. Tao, and X. Jin, "Optimal algorithms for crawling a hidden database in the Web," *PVLDB*, vol. 5, no. 11, pp. 1112–1123, 2012.

[11] M. P. Kato, T. Sakai, and K. Tanaka, "When do people use query suggestion? A query suggestion log analysis," *Inf. Retr.*, vol. 16, no. 6, pp. 725–746, 2013.

[12] M. Shokouhi, "Learning to personalize query auto-completion," in *SIGIR*, 2013, pp. 103–112.

[13] E. Agichtein and L. Gravano, "Snowball: Extracting relations from large plain-text collections," in *Digital Libraries*, 2000, pp. 85–94.

[14] O. Etzioni, M. Cafarella, D. Downey, S. Kok, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates, "Web-scale information extraction in KnowItAll: (preliminary results)," in *WWW*, 2004, pp. 100–110.

[15] Y. Fang and K. C.-C. Chang, "Searching patterns for relation extraction over the web: Rediscovering the pattern-relation duality," in *WSDM*, 2011, pp. 825–834.

[16] G. Agarwal, G. Kabra, and K. C.-C. Chang, "Towards rich query interpretation: Walking back and forth for mining query templates," in *WWW*, 2010, pp. 1–10.

[17] M. Chen, K. Q. Weinberger, and J. Blitzer, "Co-training for domain adaptation," in *NIPS*, 2011, pp. 2456–2464.

[18] W. Dai, Y. Chen, G. rong Xue, Q. Yang, and Y. Yu, "Translated learning: Transfer learning across different feature spaces," in *NIPS*, 2008, pp. 353–360.

[19] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng, "Self-taught learning: Transfer learning from unlabeled data," in *ICML*, 2007, pp. 759–766.

[20] S. Tong and D. Koller, "Support vector machine active learning with applications to text classification," *JMLR*, vol. 2, no. Mar, pp. 45–66, 2002.

[21] G. Cao, J.-Y. Nie, J. Gao, and S. Robertson, "Selecting good expansion terms for pseudo-relevance feedback," in *SIGIR*, 2008, pp. 243–250.

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

[23] Y. Xu, G. J. Jones, and B. Wang, "Query dependent pseudo-relevance feedback based on Wikipedia," in *SIGIR*, 2009, pp. 59–66.

[24] H. Tong, C. Faloutsos, and J. Y. Pan, "Fast random walk with restart and its applications," in *ICDM*, 2006, pp. 613–622.

[25] Y. Fang, K. C. Chang, and H. W. Lauw, "RoundTripRank: Graph-based proximity with importance and specificity." in *ICDE*, 2013, pp. 613–624.

[26] F. Zhu, Y. Fang, K. C. Chang, and J. Ying, "Incremental and accuracy-aware personalized PageRank through scheduled approximation," *PVLDB*, vol. 6, no. 6, pp. 481–492, 2013.

[27] S. Chakrabarti, A. Pathak, and M. Gupta, "Index design and query processing for graph conductance search," *VLDBJ*, vol. 20, pp. 445–470, 2010.

[28] S. Brin, "Extracting patterns and relations from the world wide web," in *WebDB*, 1999, pp. 172–183.

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

[30] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky, "The Stanford CoreNLP natural language processing toolkit," in *ACL: System Demonstrations*, 2014, pp. 55–60.