Searching Patterns for Relation Extraction over the Web: Rediscovering the Pattern-Relation Duality

Yuan Fang [†,*]
fang2@illinois.edu

Kevin Chen-Chuan Chang [†,*]
kcchang@illinois.edu

[†] Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL, USA
[*] Advanced Digital Sciences Center, Illinois at Singapore, Singapore

ABSTRACT

While tuple extraction for a given relation has been an active research area, its dual problem of pattern search– to find and rank patterns in a principled way– has not been studied explicitly. In this paper, we propose and address the problem of pattern search, in addition to tuple extraction. As our objectives, we stress *reusability* for pattern search and *scalability* of tuple extraction, such that our approach can be applied to very large corpora like the Web. As the key foundation, we propose a conceptual model PRDualRank to capture the notion of precision and recall for both tuples and patterns in a principled way, leading to the "rediscovery" of the Pattern-Relation Duality– the formal quantification of the reinforcement between patterns and tuples with the metrics of precision and recall. We also develop a concrete framework for PRDualRank, guided by the principles of a perfect sampling process over a complete corpus. Finally, we evaluated our framework over the real Web. Experiments show that on all three target relations our principled approach greatly outperforms the previous state-of-the-art system in both effectiveness and efficiency. In particular, we improved optimal F-score by up to 64%.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Search process*

General Terms

Design, Algorithms, Performance, Experimentation

1. INTRODUCTION

While the Web has evolved into our ultimate information repository, with most online contents being HTML text, to unleash "data inside," we are facing the immense challenge of information extraction (IE), to convert unstructured contents to structured information. While the barrier is daunting, there also come novel opportunities. The massive contents on the Internet offer both *redundancy* and *diversity*, both of which inspire new techniques. In particular, to scale up to the Web, IE has recently moved toward search-based, which is consistent with the emergence of Web-based Query Answering (QA), by sending keyword queries (*e.g.*, "capital city") to a general search engine, retrieving top pages, and processing them to extract desired information (*e.g.*, the relation capital-city-of).

This search-and-extract approach has been recently studied in several Web-based IE [9, 4] and QA [12, 7, 8] systems, which reveal consistent interesting observations:

- *Size is good– Redundancy has fundamentally changed the problem* from document-centered to corpus-centered. The size of the Web ensures that, when the corpus is considered as a whole, the "voting" effect beyond each individual document will contribute significantly to surface good results.
- *Diversity is good– Simple patterns can outperform* sophisticated NLP-based techniques. The diversity of the Web ensures that simple patterns, while inexpensive to execute, will match *some* documents to extract good results.

The insights amount to the approach of *pattern-based relation extraction*, to extract a relation R of tuples t by matching some textual patterns p. *E.g.*, for the relation capital-city-of, to extract tuples of the form (#city, #country), we can use pattern $p_1 = \langle$#city is the capital ... of #country$\rangle$ or $p_2 = \langle$#city is ... city of #country$\rangle$, both of which will match the text "Paris is the capital city of France" and extract the tuple (Paris, France). (Note that we use #E to denote an entity type, which will be explained in Sect. 3.)

Unfortunately, to date, while *pattern* is at the heart of pattern-based IE, it remains an open issue as how to find good patterns– in a principled and systematic manner. How do we find patterns like p_1 and p_2? Which one is better? Why? (Presumably, p_1 is more accurate to indicate the desired capital relationship; however, p_2 can likely return more complete answers.) For finding patterns to use, the existing works fall into two categories: On the one hand, patterns may be manually specified (by inspecting the corpus), which is the approach from early pioneering work [11] to recent ones [9, 17]. On the other hand, patterns can be learned implicitly in an iterative process, as in DIPRE [6] and Snowball [3]. As Sect. 2 will contrast, in these iterative learning approaches, patterns are lacking principled quality metrics, and they are often hidden in the iterative process of relation extraction, as only a by-product.

In this paper, as our first objective, we propose and address the problem of *pattern search*– to find and rank reusable patterns in a principled way. As Fig. 1(a) illustrates, given a small number of seed examples, such as (Ottawa, Canada) and (Beijing, China), we wish to search systematically in the space of candidates to return good patterns in an order ranked by their "quality." Such explicit and principled discovery of patterns not only enhances extraction effectiveness– because patterns are now of high quality– but also

WSDM'11, February 9–12, 2011, Hong Kong, China.

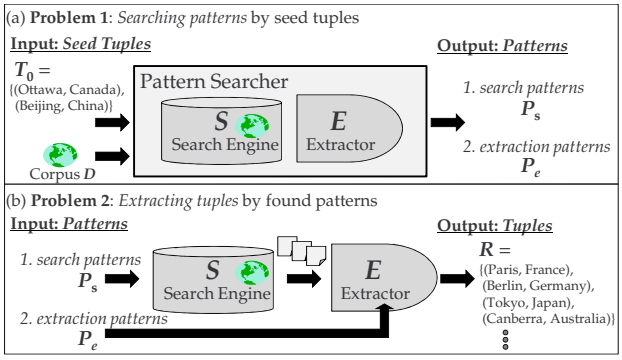


Figure 1: Our focus– the dual problems

enables new scenarios. First, with reusable patterns, we can execute and re-execute these patterns on demand to extract new tuples over an evolving corpus like the Web, without relearning the patterns as a part of the extraction process. Second, with such patterns, we can also execute focused retrieval by combining them with ad-hoc keywords (*e.g.*, use "ancient history ⟨#city is the capital ... of #country⟩" to find capital cities that are also historical), as recently explored in ad-hoc content querying [17].

Further, as our second objective, upon the patterns found we study the subsequent problem of scalable *tuple extraction*, to form a complete framework for relation extraction, as Fig. 1(b) illustrates. Our proposed solution relies on the ranked patterns of various granularities to match different scopes of the corpus, enabling the extraction of new tuples in a scalable manner. It is also important to return tuples of high "quality" in a principled way, parallel to the problem of pattern search. Factoring in both requirements, the ultimate goal of the patterns can be fulfilled– to efficiently and effectively find more tuples for a given relation.

As it turns out, both problems boil down to developing effective *ranking* of patterns and tuples. Our goal is thus to study the principles for such ranking. While no formal principles exist, the informal insight of *Pattern-Relation Duality* (or *PR Duality*) has long been observed since DIPRE [6]. It is essentially stated as follows:

PR Duality (Original): *Given a good set of patterns, we can build a good set of tuples. Given a good set of tuples, we can build a good set of patterns.*

While the insight has been widely used in existing works, it falls short of providing a formal principle. Subsequent works [3, 2] continue to leverage the insight, but mostly resort to heuristics for modeling the iterative reinforcement. First, what does "good" mean? We need to formalize a set of metrics– precision and recall– for capturing the quality of patterns as well as tuples. Second, how exactly do patterns and tuples interrelate? We need to develop a probabilistic inference framework that captures the semantic "propagation" in between. As the foundation of our approach, we address these questions and propose the conceptual model PRDualRank, and thus "rediscover" *PR Duality* as the underlying principle for ranking both patterns and tuples.

PR Duality (Rediscovered): *Patterns and tuples reinforce each other under the quality metrics of precision and recall. That is, the quality of a pattern– both precision and recall– can be determined by the tuples it extracts. Likewise, the quality of a tuple can be determined by the patterns it matches.*

Upon the conceptual model PRDualRank, we further develop a concrete framework that applies the principles in PRDualRank to solve the dual problems of pattern search and tuple extraction over the Web. Conceptually, PRDualRank assumes complete knowledge of all possible patterns and tuples, which is infeasible on a large corpus. Hence we introduce a sampling process for PRDualRank, which follows two guiding principles based on the *perfect corpus* and *perfect sampling* assertions.

In summary, this paper makes the following contributions:

- We propose the novel problem of *pattern search*, to systematically rank both search and extraction patterns.
- We present the formal inference mechanism of *PR Duality*, the key ranking principle of our approach PRDualRank;
- We develop the concrete framework of PRDualRank, guided by the principles of a perfect sampling process over a complete corpus.
- We performed extensive experiments over the real Web on three relations and demonstrated the effectiveness. In particular, we improved optimal F-score by a factor up to 1.64.

2. RELATED WORK

In terms of problem. Extracting tuples of a given relation from a text corpus has long been studied [6, 3, 9]. However, its dual problem of searching textual patterns only exists implicitly as an intermediate step of tuple extraction.

In contrast, this paper treats patterns as first class citizens. We first stress their *reusability*. We explicitly search and rank patterns in a principled way, such that they can be reused for future extraction (*e.g.*, over the evolving Web at a later point), or to be used by other systems like DoCQS [17]. Second, we stress their *scalability*, which hinges on a principled framework for finding out "searchable" patterns to narrow the scope of extraction. Most previous works but QXtract [4] have not discussed the automatic discovery of searchable patterns. QXtract generates queries (*i.e.*, searchable patterns) to retrieve relevant documents that may contain tuples in a preprocessing step, after which existing methods like Snowball [3] is used on the retrieved documents separately. We instead identify the principles behind the ranking of patterns, and thus integrate the ranking of such searchable patterns into the same framework PRDualRank as other kinds of patterns, further enhancing *PR Duality*.

For quality metrics, none of the previous works explores any principled framework to derive the quality of the tuples and patterns. In particular, none of the works develop a complete set of metrics covering both precision and recall in IR. Instead, most emphasize "confidence" or "probability assessment," which captures precision only in an informal and heuristic way. *E.g.*, in Snowball [3], confidence estimation is based on the assumption of independent patterns, resulting in excessively high confidence for all tuples. In QA [13], the relation must have a key attribute. *E.g.*, for capital-city-of, #country is a key attribute as it cannot associate with more than one #city as its capital. In KnowItAll [9, 10], it is assumed that tuples of one relation do not satisfy other relations. These assumptions are not always valid. Some other works such as [14, 15, 5] estimate the confidence based on some simple statistics.

For problem setting, we rely on bootstrapping using seed tuples, similar to [6, 3].

In terms of techniques. We propose a semi-supervised approach based on iterative inference on tripartite graphs constructed from tuples, patterns and their contexts. Such graph-based inference propagation is widely reported in the literature [2, 1, 16]. Specifically, our inference involves rewriting of a node's precision and recall in terms of its neighbors on the graph, which maps to random walks as inspired by [1]. In [1], a tripartite graph can be constructed from queries, templates, and sites. The precision and recall of queries can be inferred from its neighboring sites through click-throughs and its neighboring templates through instantiation. We

(a) Example Snippets from Corpus D = {d_1, d_2, d_3, d_4, d_5, d_6}

Id	Doc	Snippet	Freq.
s_1	d_1, d_3, d_4, d_5	**Paris** is the capital city of *France*…	4
s_2	d_1	**Paris** is the largest city of *France*…	1
s_3	d_2, d_3	**Ottawa** is the national capital city of *Canada*…	2
s_4	d_4, d_6	**Toronto** is the largest city of *Canada*…	2

(b) Tuples T

t_1 = (Paris, France) , t_2 = (Ottawa, Canada), t_3 = (Toronto, Canada)

(c) Patterns

1. Search Pattern P_s

p_a = <tourist | government>
p_b = <capital city | congress>

2. Extraction Patterns P_e

p_1 = <#city is the national capital … of #country>
p_2 = <#city is the … capital city of #country>
p_3 = <#city is the largest city of #country>
p_4 = <#city is the … city of #country>

Figure 2: Running example

share the insight that our approach is similar in inferring the quality of tuples and patterns from their neighboring nodes. However, two major differences exist.

First, in [1] all queries and click-throughs are available from a query log. All possible templates can also be enumerated from this log. But in our problem, we are only given a few seeds as input, without the complete spaces of tuples and patterns. Thus, we must resort to partial materialization of the graph, which is a form of sampling. We establish two assertions, *perfect corpus* and *perfect sampling*, as guiding principles for the sampling process.

Second, although we also construct a tripartite graph, it is essentially a bipartite graph in which the contexts only bridge tuples and patterns to model the abstract notion of binding between them. However, contexts are still necessary as a fundamental concept, upon which precision and recall can be defined.

3. PROBLEM: PATTERN SEARCH

Towards scalable and reusable pattern-based relation extraction, we must address two dual problems, in which the concept of *pattern* is central. As Fig. 1 shows: with respect to a *corpus* D of documents (say, the Web), we are interested in developing a mechanism that can extract a target relation R from D, by way of searchable and reusable patterns.

3.1 Data Model

We start with defining our data model. Consider a running example in Fig. 2 over a toy corpus D (say, some web pages), for a target relation R = capital-city-of.

A *corpus* D, such as the web, is a collection of documents, where some "snippets" of text may contain relations of interest. Fig. 2 shows some example snippets. Each snippet may appear multiple times, as its *frequency*; *e.g.*, s_1 occurs 2 times. Since our extraction is to recognize a span of text (such as "Paris" in s_1) as a certain semantic role (*e.g.*, a capital city), we denote the *vocabulary*, *i.e.*, all the tokens (which can be words, phrases, and symbols, depending on applications) in D, by $\Omega(D)$.

As our ultimate target, a *relation* contains a set of *tuples* (which we aim to extract) of the form $(e_1, \ldots, e_n)$, where e_i is an attribute or *entity*. Each entity e of a type #E is an instance of its type (note that a type is prefixed by # for clarity), which draws value from its domain Ω(#E) to form an *instance*; *i.e.*, if e is of type #E, then $e \in \Omega$(#E). *E.g.*, e = "France" is of type #country, where Ω(#country) = {"France", "USA", "China", ...}. Note that the recognition of entities from text– *e.g.*, that "France" in snippet s_1 is a #country– is called *entity tagging*, and is itself an active research area. As this paper focuses on relation extraction, we assume that the Extractor E (Fig. 1) is equipped with an entity tagger; many off-the-shelf tools are available today. In Fig. 2, we thus recognize three tuples t_1, t_2, and t_3. While our discussion in this paper assumes a binary relation R of the form (#E$_1$, #E$_2$), the framework is general with respect to the arity.

At the core of pattern-based extraction lies the notion of "patterns." Generally, a *pattern* is a syntactic structure with tokens (from $\Omega(D)$) and entities (from R) that describe a "presentation" of a desired tuple– *i.e.*, how it may appear in D. Over a large corpus, to speed up extraction, we will need patterns of different "granularities" to match various scopes like documents, passages, and sentences. Our framework generally supports such variously-scoped patterns in an integrated mechanism (Sect. 4).

For our discussion, in this paper, we assume two classes of patterns, each with a rather simple form. However, we stress that the framework is general to handle any number of various classes of patterns, which can be defined in arbitrarily complex forms. Our *first* class of patterns aims at retrieving relevant documents from the corpus by a search engine: Thus, intuitively, a *search pattern* is a set of searchable keyword phrases matching a relevant document d. Fig. 2 shows some example search patterns, *e.g.*, p_b = ⟨capital city | congress⟩ will match documents with two phrases "capital city" and "congress" (in no particular order). Since a search pattern is to be executed by a search engine, its expressiveness is limited by the chosen engine. Our *second* class of patterns will locate actual slots where desired entities occur to form a tuple. Thus, an *extraction pattern* further matches a text snippet s within relevant documents d. In Fig. 2, p_1 = ⟨#city is the national capital ... of #country⟩ specifies that the entities and the keywords appear in the order, with some optional text "..." in between. It will match s_1– by aligning the words and recognizing "Paris" as a #city and "France" as a #country– and extract (Paris, France) as a matching tuple. Formally, we adopt the following simple forms of patterns:

Definition 1 (Search Pattern): A *search pattern* is a set of phrases, written as $\langle g_1 | \ldots | g_m \rangle$. Each *phrase* g_i is an n-gram of words $(w_1 \ldots w_n)$, where $w_i \in \Omega(D)$, and $|g_i| = n \leq L_{max}$, for some choice of phrase length L_{max}. ■

Definition 2 (Extraction Pattern): An *extraction pattern* for (#E$_1$, #E$_2$) is a list of two phrases, $\langle g_1 \ldots g_2 \rangle$ or $\langle g_2 \ldots g_1 \rangle$, where "…" is an optional wildcard. Each g_i is a phrase containing a reference to #E$_i$, *i.e.*, $g_i = l_1, \ldots, l_{n_1}$, #E$_i$, $r_1, \ldots, r_{n_2}$ where $l_i, r_j \in \Omega(D)$ and $|g_i| \leq L_{max}$, for some choice of phrase length L_{max}. ■

3.2 Problems

Our approach consists of two dual stages, as Fig. 1 illustrates. As input, like many existing pattern-based extraction efforts [6, 3], we assume a small number of seed tuples (*e.g.*, {(Ottawa, Canada), (Beijing, China)}), and our ultimate goal is to find the matching relation (*e.g.*, tuples for capital-city-of). However, unlike existing works, we emphasize the role of patterns, and stress the needs for scalability and reusability. For *scalability*, as we just explained, we support two (or more) classes of patterns, where search patterns can quickly retrieve relevant documents, and extraction patterns can accurately identify tuples from text. For *reusability*, we systematically search for good patterns as a first-class citizen (and not a by-product hidden in the extraction process), and rank them with principled quality metrics. Overall, the framework consists of two stages, and thus two problems we must develop effective techniques for.

Pattern Search. Given a small number of seed tuples from relation R, search the corpus to find and rank patterns for R.

Tuple Extraction. Given discovered patterns for R, execute them over the corpus to find and rank tuples for R.

4. PATTERN-RELATION DUALITY

Our key challenge lies in *ranking*– to find good patterns (for pattern search) and good tuples (for tuple extraction). To develop the principles of ranking, our study attempts to formalize *PR Duality*, leading to the PRDualRank framework– which ranks patterns and tuples by the principles. While our discussion mentions only extraction patterns, the same framework can be used for search patterns (and, in principle, any patterns that match certain scopes for extraction), as our experiments also demonstrate.

Pattern/Tuple Space. To begin with, we examine our search space. That is, to search for patterns, what is the set of candidate patterns we should consider?

Consider extraction patterns. By Def. 2, a pattern is a combination of two phrases g_1 and g_2 (or, in general, n such g_i for n-nary relations), each of which can be at most L_{max} in length. Each occurrence e_i of an entity type #E$_i$ (say, "Paris" as a #city in snippet s_1) can form such a phrase with its surrounding tokens (say "#city is", "#city is the", *etc.*). With respect to corpus D, we thus consider the space of extraction patterns as $P = \{\ p \mid p = \langle g_1 \ldots g_2 \rangle, g_i$ is a phrase containing a reference to #E$_i$, g_i occurs in D, $|g_i| \leq L_{max}\}$. For our example, in Fig. 2, the space would include p_1, p_2, p_3, p_4, and many that are not listed, such as ⟨#city is ... of #country⟩.

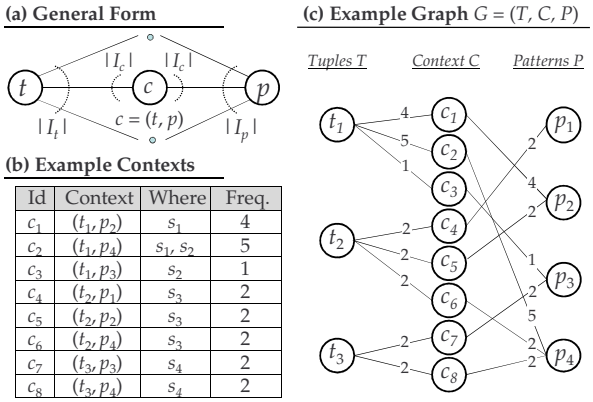
Similarly, for ranking tuples, there is also a potentially large space of candidates. For a relation R of the form (#E$_1$, #E$_2$), such as (#city, #country), the complete space is the Cartesian product of the two entity domains Ω(#E$_1$)$\times\Omega$(#E$_2$), such as any combinations of city and country. As our goal is to extract such tuples from corpus D, we should only consider those pairs that actually appear in D– in fact, since entities that are far apart are unlikely to be semantically associated, we need only consider those #E$_1$ and #E$_2$ that are close within some proximity window W_{max}. Thus, the space of tuple to consider, with respect to corpus D and relation R, is $T = \{\ t \mid t = (e_1, e_2), e_i \in \Omega(\#\mathsf{E}_i), e_1, e_2$ occur in D within W_{max} words $\}$. For our example (Fig. 2, assuming $W_{max} = 10$), if the entities only occur in the listed snippets, then the space is $T = \{t_1, t_2, t_3\}$.

Pattern-Tuple Association. Towards an essential objective of this paper– to formally develop *PR Duality*– we further capture the associations of patterns and tuples, upon their dual spaces. The modeling of such associations enables us to systematically relate patterns and tuples, which will effectively form the "bridge" to "induce" the duality between them, as we will see.

Intuitively, each co-occurrence of a tuple t and pattern p in some snippet in corpus D forms an association that is "intended" (by the author of the document), and thus it bears some semantic relatedness. We refer such an association (t, p) as a *context* consisting of t and p. For $c = (t, p)$, we say c *associates* (or binds) t and p; on the other hand, we also say t and p *instantiate* c. Note that since each context (t, p) can occur multiple times in different snippets, it has a *frequency* as the total occurrences. *E.g.*, Fig. 3(b) lists the contexts occurring in the snippets of Fig. 2. t_1 co-occurs with p_2 and thus forms a context $c_1 = (t_1, p_2)$, with a frequency = 4 (since snippet s_1 occurs 4 times). As another example, t_1 and p_4 co-occur at both s_1 and s_2, forming $c_2 = (t_1, p_4)$, with a total frequency = 5.

The space of contexts includes any combinations of tuples and patterns that actually occur in corpus D, *i.e.*, $C = \{\ c \mid c = (t, p), t \in T, p \in P, t$ and p co-occur in $D\}$. Since a context c can occur at different snippets, when necessary, we will write $c@s$ to indicate the specific occurrences of c at s; *e.g.*, c_2 occurs at s_1 and s_2, or $c_2@s_1$ and $c_2@s_2$. Also note that C is a multiset, since a snippet can appear multiple times (say, s_1 has a frequency = 4; Fig. 2). For the running example, as Fig. 3(b) lists, we have

$$C = \{c_1@s_1 : 4, c_2@s_1 : 4, c_2@s_2 : 1, \ldots, c_8@s_4 : 2\}. \quad (1)$$



(b) Example Contexts

Id	Context	Where	Freq.
c_1	(t_1, p_2)	s_1	4
c_2	(t_1, p_4)	s_1, s_2	5
c_3	(t_1, p_3)	s_2	1
c_4	(t_2, p_1)	s_3	2
c_5	(t_2, p_2)	s_3	2
c_6	(t_2, p_4)	s_3	2
c_7	(t_3, p_3)	s_4	2
c_8	(t_3, p_4)	s_4	2

Figure 3: Context Graph– bridging T and P

While a context associates a specific pair of (t, p), a pattern p can instantiate multiple contexts, and so can a tuple t. Let us denote the set of contexts that p instantiates by I_p, and similarly that of t by I_t. *E.g.*, $I_{p_1} = \{c_4\}$ and $I_{t_1} = \{c_1, c_2, c_3\}$, according to Fig. 3(b). Formally it is defined by the following, where $*$ represent "any."

$$I_p \equiv \{c \mid c \in C, c = (*, p)\};\ \ I_t \equiv \{c \mid c \in C, c = (t, *)\}$$

Furthermore, let $I_{tp} \equiv I_t \cap I_p$, *i.e.*, $\{c \mid c \in C, c = (t, p)\}$.

Given the spaces P, T and C, we can construct a Context Graph $G = (T, C, P)$ as a convenient way to visualize associations. G is a tripartite graph where T, C, P are the disjoint set of nodes, *i.e.*, $t_i \in T, c_i \in C, p_i \in P$ as in Fig. 3(c). C also models the set of edges, such that a context $c = (t, p)$ connects t and p from the two sides. *E.g.*, $c_1 = (t_1, p_2)$ connects t_1 and p_2. In general, a context c only connects to exactly one tuple and one pattern by definition, as illustrated in Fig. 3(a). Furthermore, a context node in the graph has the same in-degree and out-degree, since it is a pair of tuple and pattern that appear together. Intuitively, a context bridges a tuple and a pattern, allowing "passing-though" between them.

4.1 Metrics: Precision and Recall

Given the space of possible patterns P, we need formal metrics to measure their quality in order to find good ones. As our search and ranking spaces encompass interrelated T, C and P, we aim to develop metrics that will uniformly apply to all these notions, in order to unify and integrate their "mutual reinforcement," which is the intuition of *PR Duality*.

To determine the right metrics, we investigate the purpose of patterns– which is to "interpret" a snippet s, to determine if we should extract the tuple embedded in s for target relation R. A pattern p, when associated with a tuple t, forms a context $c = (t, p)$, as a particular *interpretation* for a specific snippet s– *i.e.*, when c occurs at s, or $c@s$, it forms an interpretation of s. Consider s_1 "Paris is the capital city of France." It has two interpretations: One is by context $c_1@s_1$: "Paris is the capital city of France," and the other is by $c_2@s_1$: "Paris is the ... city of France". With an *oracle* (such as a human expert), we would be able to judge each interpretation, to decide if the embedded tuple is relevant to R. *E.g.*, for the interpretations of s_1 as just observed, our oracle would likely determine that $c_1@s_1$ is relevant to capital-city-of, while $c_2@s_1$ is not. By checking each context occurrence, our oracle can determine the

1) QuestP: *Quest Backward for Precision Inference*	2) QuestR: *Quest Forward for Recall Inference*
$\underline{\boldsymbol{P1}}$: $\mathcal{P}(p) = \sum_{t_i \in \tau(p)} \mathcal{P}(t_i) \cdot \frac{\lvert I_{t_i p} \rvert}{\lvert I_p \rvert}$	$\underline{\boldsymbol{R1}}$: $\mathcal{R}(p) = \sum_{t_i \in \tau(p)} \frac{\lvert I_{t_i p} \rvert}{\lvert I_{t_i} \rvert} \mathcal{R}(t_i)$
$\underline{\boldsymbol{P2}}$: $\mathcal{P}(t) = \begin{cases} \mathcal{P}_0(t) & \text{if } t \in T_0; \\ \mathcal{P}(t) = \sum_{p_i \in \pi(t)} \mathcal{P}(p_i) \cdot \frac{\lvert I_{t p_i} \rvert}{\lvert I_t \rvert} & \text{otherwise.} \end{cases}$	$\underline{\boldsymbol{R2}}$: $\mathcal{R}(t) = \sum_{p_i \in \pi(t)} \frac{\lvert I_{t p_i} \rvert}{\lvert I_{p_i} \rvert} \mathcal{R}(p_i)$

Figure 4: The dual inference framework: QuestP and QuestR

set of all *relevant contexts*– *i.e.*, the occurrences that form a relevant interpretation. For our example, examining the entire context space C (Eq. 1), our oracle can determine

$$C_R = \{c_1@s_1 : 4, c_4@s_3 : 2, c_5@s_3 : 2\}. \quad (2)$$

Since patterns match contexts– *i.e.*, by combining with tuples to form contexts– a good pattern should match more relevant contexts, and less irrelevant ones. In other words, the purpose of a pattern is, thus, to retrieve those relevant context in C_R. We can thus measure the quality of pattern t by its "retrieval effectiveness" for finding C_R (*e.g.*, Eq. 2) from the entire space C (*e.g.*, Eq. 1).

The objectives of set retrieval (from C_R) naturally translates to the need of both *precision* $\mathcal{P}$ and *recall* $\mathcal{R}$ metrics, as in standard IR evaluation. Consider pattern p_1, which retrieves $I_{p_1} = \{c_4@s_3{:}2\}$. With respect to C_R, we thus have precision $\mathcal{P}(p_1) = \frac{|C_R \cap I_{p_1}|}{|I_{p_1}|} = \frac{2}{2} = 1.0$, and recall $\mathcal{R}(p_1) = \frac{|C_R \cap I_{p_1}|}{|C_R|} = \frac{2}{4+2+2} = 0.25$. As another example, pattern p_2 that retrieves $I_{p_2} = \{c_1@s_1 : 4, c_5@s_3 : 2, c_9@s_5 : 4\}$– suppose there are also snippets s_5 reading "Milan is the fashion capital city of Italy" in addition to the running example– will have $\mathcal{P}(p_2) = \frac{4+2}{4+2+4} = 0.6$ and $\mathcal{R}(p_2) = \frac{4+2}{4+2+2} = 0.75$. Apparently, while some patterns like p_1 are more accurate with a higher precision but a lower recall, other patterns like p_2 are broader with a higher recall but a lower precision. Hence the need for both metrics is clear. To capture such intuitions, we formally define the quality metrics of pattern p, $\forall\, p \in P$:

$$\mathcal{P}(p) = |C_R \cap I_p|/|I_p| \quad (3)$$
$$\mathcal{R}(p) = |C_R \cap I_p|/|C_R| \quad (4)$$

We can similarly define the quality metrics for tuple t, which also instantiate a set of context I_t. By comparing I_t with C_R, we state the metrics as, $\forall t \in T$:

$$\mathcal{P}(t) = |C_R \cap I_t|/|I_t| \quad (5)$$
$$\mathcal{R}(t) = |C_R \cap I_t|/|C_R| \quad (6)$$

Finally, these metrics also apply to each context c, $\forall c \in C$. Simply let $I_c = \{\mathrm{c}\}$; *i.e.*, we view each c as retrieving itself:

$$\mathcal{P}(c) = |C_R \cap I_c|/|I_c| \quad (7)$$
$$\mathcal{R}(c) = |C_R \cap I_c|/|C_R| \quad (8)$$

4.2 Inference: Random Walks

While we have the right metrics defined, we must realize them in a probabilistic sense. For pattern search, as our problem setting (Fig. 1), we are only given as input a few seed tuples– instead of the complete C_R. In other words, the deterministic definitions of $\mathcal{P}$ and $\mathcal{R}$ in Eq. 3–8 require the ground-truth of relevant contexts C_R. Since we lack such a ground-truth, it is only feasible to redefine precision and recall probabilistically.

Probabilistic Modeling. As the foundation, we must generalize the deterministic senses of precision and recall to probabilistic definitions. Let's start with patterns, and consider Eq. 3 in a statistical way. Since the numerator is the count, or *frequency*, of $C_R \cap I_p$, it represents the probability, when we draw a random context c, that $c \in C_R$ and $c \in I_p$ both hold, *i.e.*, $\Pr(c \in C_R, c \in I_p)$. Similarly, the denominator represents $\Pr c \in I_p$. Substituting them into Eq. 3, we obtain the *probabilistic precision* of p as the conditional probability of $c \in C_R$ given $c \in I_p$– *i.e.*, the likelihood that c is relevant, given that it is instantiated by p. We can similarly generalize Eq. 4 to the *probabilistic recall* of p:

$$\mathcal{P}(p) = \Pr(c \in C_R | c \in I_p) \quad (9)$$
$$\mathcal{R}(p) = \Pr(c \in I_p | c \in C_R) \quad (10)$$

For tuples $t \in T$, we can generalize quite similarly:

$$\mathcal{P}(t) = \Pr(c \in C_R | c \in I_t) \quad (11)$$
$$\mathcal{R}(t) = \Pr(c \in I_t | c \in C_R) \quad (12)$$

For contexts $c \in C$, we can also rewrite Eq. 7 as $\mathcal{P}(c) = \Pr(x \in C_R | x \in I_c)$. Further, since I_c is simply $\{c\}$ itself, the condition $x \in I_c$ means $x = c$, which simplifies the expression to just $\Pr(c \in C_R)$. In words, the precision of c measures how likely c is relevant– which is quite intuitive. We can thus generalize its $\mathcal{P}$ and $\mathcal{R}$:

$$\mathcal{P}(c) = \Pr(c \in C_R) \quad (13)$$
$$\mathcal{R}(c) = \Pr(x = c | x \in C_R) \quad (14)$$

Probabilistic Inference. The generalized probabilistic senses of precision and recall, while simple, forms an elegant system of inference, as we will next establish. To highlight, we summarize the results in Fig. 4, which describes the mutual inference rules $P1$ and $P2$ for precision, as well as $R1$ and $R2$ for recall, between tuples and patterns. The mutual inference can be derived quite simply, through only mechanical derivation using elementary probability theorems. We note that our general approach of probability rewriting is inspired by the QueST framework [1]. While we are addressing a distinct problem with significant differences, as Sect. 2 discussed, the probabilistic inference parallels that in [1]. We thus name the updating framework (Fig. 4) the same as in QueST– *i.e.*, QuestR for precision, and QuestP for recall.

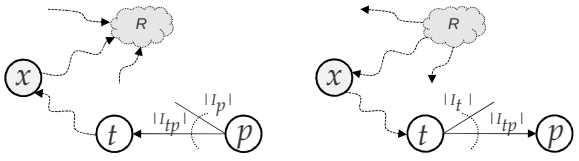
While our derivations do not explicitly require the Context Graph, our results can be interpreted as random walks on the graph as we will see. We start from a tuple (or pattern), and perform random walks to reach a pattern (or tuple). In the context graph, a context only serves as a bridge that connects a tuple and a pattern, which simply allows pass-throughs to reach the other "side" of the graph.

(1) Precision: The probabilistic precisions of patterns p (Eq. 9) can be inferred from those of tuples t (Eq. 11), and vice versa, by way of the "bridging" of contexts c (Eq. 13). We derive below.

Establish C as Bridge: To begin with, for context c, we can rewrite $\mathcal{P}(c)$ as $\mathcal{P}(c.\mathrm{t})$, where $c.\mathrm{t}$ denote the tuple component of c, *i.e.*, if $c = (t, p)$, then $c.\mathrm{t} = t$. Intuitively, the precision of c is "passed through" to that of its tuple t.

$$\begin{aligned} \mathcal{P}(c) \equiv^1 \Pr(c \in C_R) &=^2 \textstyle\sum_{t_i \in T} \Pr(c \in C_R, c \in I_{t_i}) \\ &=^3 \textstyle\sum_{t_i \in T} \Pr(c \in C_R | c \in I_{t_i}) \Pr(c \in I_{t_i}) \\ &=^4 \textstyle\sum_{t_i = c.\mathrm{t}} \Pr(c \in C_R | c \in I_{t_i}) \Pr(c \in I_{t_i}) \\ &=^5 \mathcal{P}(c.\mathrm{t}) \end{aligned} \quad (15)$$

After step 1 starting with Eq. 13, step 2 breaks it into joint probability with $c \in I_{t_i}$ for all t_i. Step 3 expands it by Bayes' theorem.



(a) *Precision* by backward random walk. (b) *Recall* by forward random walk.

Figure 5: Interpreting QuestP and QuestR as random walks

Algorithm PRDualRank (G, T_0)

Input:	$G = (T, C, P)$, the context graph.
	$T_0 \subset T$, seed tuples.
Output:	P and T ranked by precision/recall.

1) **for each** $t \in T_0$ **do** $\mathcal{P}(t) \leftarrow 1; \mathcal{R}(t) \leftarrow \frac{1}{|T_0|}$; **endfor**
2) QuestP: update $\mathcal{P}$ till convergence by rules $P1, P2$;
3) QuestR: update $\mathcal{R}$ till convergence by rules $R1, R2$;
4) **return** P, T ranked with $\mathcal{P}$ and $\mathcal{R}$ scores;

Figure 6: Ranking patterns and tuples by *PR Duality*

Step 4 simplifies the summation to only one term, since we recognize that $\Pr(c \in I_{t_i})$, for a particular c, is non-zero only when $t_i = c.\text{t}$, since each context c has exactly one tuple t. In step 5, since $\Pr(c \in I_{t_i}) = 1$ when $t_i = c.\text{t}$, it is removed, and the remaining term is $\mathcal{P}(t_i)$ (by Eq. 11), *i.e.*, $\mathcal{P}(c.\text{t})$.

Infer P by T: We rewrite $\mathcal{P}(p)$ with $\mathcal{P}(t)$, for those t that it associates with, which we denote by $\tau(p) \equiv \{t \mid (t, p) \in C\}$. The derivation leads to rule $P1$ in Fig. 4.

$$\begin{aligned}
\mathcal{P}(p) &\equiv^1 \Pr(c \in C_R | c \in I_p) \\
&=^2 \textstyle\sum_{c_i \in C} \Pr(c \in C_R, c = c_i | c \in I_p) \\
&=^3 \textstyle\sum_{c_i \in I_p} \Pr(c \in C_R | c = c_i, c \in I_p) \cdot \Pr(c = c_i | c \in I_p) \\
&=^4 \textstyle\sum_{c_i \in I_p} \Pr(c \in C_R | c = c_i) \cdot \Pr(c = c_i | c \in I_p) \\
&=^5 \textstyle\sum_{c_i \in I_p} \Pr(c_i \in C_R) \cdot \Pr(c = c_i | c \in I_p) \\
&=^6 \textstyle\sum_{c_i \in I_p} \mathcal{P}(c_i) \cdot \frac{|I_{c_i}|}{|I_p|} =^7 \sum_{t_i \in \tau(p)} \mathcal{P}(t_i) \cdot \frac{|I_{t_i p}|}{|I_p|} \quad (16)
\end{aligned}$$

Step 2 expands with joint distributions with every c_i, and step 3 uses Bayes' theorem. Step 4 removes condition $c \in I_p$, of which $c \in C_R$ is conditionally independent, given the more specific condition $c = c_i$. Step 5 then simply substitute c_i for c in the first term, resulting in the removal of the condition. Then, in step 6, we recognize the first term as simply $\mathcal{P}(c_i)$, while the second term can be calculated as $\frac{|I_{c_i}|}{|I_p|}$– *i.e.*, the proportion of c_i among the contexts that p instantiates. Finally, step 7 "passes through" Eq. 15 as a "bridge" to T; it substitutes t_i for c_i– since these t_i instantiate c_i, and c_i is also instantiated by p, it means $t_i \in \tau(p)$.

Infer T by P: Conversely, by symmetry with the above process, we can rewrite $\mathcal{P}(t)$ in terms of those p that it associates with, denoted by $\pi(t)$, *i.e.*, rule $P2$ in Fig. 4. Note that the update does not apply to seed tuples $t \in T_0$. Since they are "labeled" examples, they maintain their given values $\mathcal{P}_0(t)$ assigned initially.

(2) Recall: In parallel to the derivation for precision, the probabilistic recall of patterns p (Eq. 10) can be inferred from those of tuples t (Eq. 12), and vice versa, again through the "bridging" of contexts c (Eq. 14). For brevity, we omit the derivation, which leads to rules $R1$ and $R2$ in Fig. 4.

Interpretation. The resulting inference framework, QuestP for precision and QuestR for recall, exhibits rather interesting duality– they are symmetric and opposite to each other. Similar to what the original QueST framework [1] has observed, the inference can be interpreted as random walks on the Context Graph. However, unlike the interpretation in [1], the random walk exists over the network between T and P– and C is only a "pass-through."

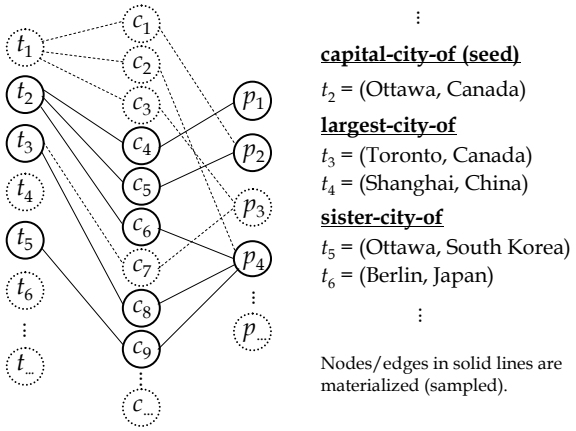


Figure 7: A partially materialized Context Graph

QuestP is a random walk backward as depicted in Fig. 5(a), *i.e.*, starting from p, what is the probability that we reach the hidden origin R? It is consistent with probabilistic precision of p, which is the proportion of tuples associated with p that will reach R.

QuestR is a random walk forward as depicted in Fig. 5(b), *i.e.*, starting from some tuple hidden in R, what is the probability that we reach p? It is consistent with probabilistic recall of p, which is the proportion of tuples in the hidden R that will reach p.

Overall Framework: PRDualRank. We now introduce the overall framework called PRDualRank to rank both tuples and patterns by precision as well as recall. We outline the framework in Fig. 6. Given the seed tuples and the Context Graph, we first initialize precision and recall for tuples (line 1). Next, we invoke QuestP to update the precision of tuples and patterns iteratively until convergence (line 2). QuestR is similarly invoked to update the recall (line 3). The convergence property has been discussed elsewhere (*e.g.*, in [1]). Finally, ranked tuples and patterns are outputted (line 4).

Key Principle: PR Duality. As a concluding remark, the conceptual model PRDualRank not only exemplifies the original *PR Duality* in [6], but also formally quantifies and thus "rediscovers" it (as first stated in Sect. 1): both precision and recall of a pattern can be expressed in terms of its associated tuples, by rule $P1$ and $R1$ in Fig. 4. In duality, the quality of a tuple can be expressed in terms of its associated patterns, by rule $P2$ and $R2$.

5. CONCRETE FRAMEWORK

Suppose the conceptual model can be represented by the tripartite graph in Fig. 7, which is extended from the example graph in Fig. 3. Tuples $t_1, \ldots, t_3$, patterns $p_1, \ldots, p_4$ and context $c_1, \ldots, c_8$ are from the running example. For now ignore the solid/dashed lines. Starting only with a few seed tuples (*e.g.*, t_2), we lack the complete space of tuples, patterns, as well as the contexts that bridge them. Thus it is difficult to use the conceptual model directly. Instead, we resort to partial materialization. *E.g.*, in Fig. 7 a partial graph is constructed which only includes solid nodes and edges. In this example only the extraction patterns are shown; a similar graph can be constructed for the search patterns. Thus the challenge lies in devising a good sampling process that partially materializes the conceptual model.

We next describe the guiding principles for the sampling process. Subsequently we present the solutions to the dual problems of pattern search and tuple extraction.

5.1 Guiding Principles

Before establishing any guidelines, it is worth noting that a tuple may have multiple senses. *E.g.*, (Paris, France) satisfies both capital-city-of and largest-city-of. Consider a tuple t that satisfies relations $R_1, \ldots, R_n$. Assign each context $c \in I_t$ to multisets $I_{t,R_1}, \ldots, I_{t,R_n}$, such that $c \in I_{t,R_i}$ iff c is relevant to R_i. In other words, the contexts in the same multiset reflect the same relation.

First of all, we must capture all "relevant" patterns that instantiate relevant context(s). *E.g.*, in Fig. 7 it is necessary to sample relevant patterns p_1, p_2, while it is acceptable to miss the irrelevant p_3. We assert that this can be achieved on a *perfect corpus*.

Assertion 1 (Perfect Corpus): A perfect corpus is complete and balanced, such that contexts relevant to R would be identically distributed for every tuple in R. Specifically, given any tuple t and t' that are in R, there exists a one-one correspondence on $I_{t,R}$ and $I_{t',R}$ (we denote $I_{t,R} \Leftrightarrow I_{t',R}$), such that $c \in I_{t,R}$ maps to $c' \in I_{t',R}$ if c and c' are both instantiated by the same pattern. ■

Intuitively, for a relation R, a perfect corpus would contain snippets that express each tuple of R in every possible way. Thus, all relevant patterns can be sampled by examining the contexts instantiated by any tuple of R. This intuition is formally stated as Lemma 1. Its proof is trivial and thus omitted.

Lemma 1: Assume a perfect corpus. Given a pattern p, if $I_p \cap C_R \neq \varnothing$, then for any tuple t in R, there exists a context (t, p). ■

After sampling all relevant patterns, it is also important to capture tuples of different relations. As illustrated in Fig. 7, $t_2, \ldots, t_6$ satisfy three different relations on Ω(#city) and Ω(#country). Intuitively, if all of these relations are sampled, it helps to better distinguish target tuples from the rest, similar to the building of a classifier which requires both positive and negative (or at least unlabeled) examples. Ideally at least one from t_3, t_4 and one of t_5, t_6 should be sampled, given the seed t_2. The sampled tuples are our unlabeled examples. Here, in terms of their relation, t_3 is equivalent to t_4, and so is t_5 to t_6. Formally, two tuples t, t' are equivalent iff they satisfy exactly the same relations (denoted $t \sim t'$). The tuple space T can be partitioned into equivalence classes consisting of equivalent tuples. We denote the equivalence classes $T_1, \ldots, T_m$, i.e., $T = \cup_{k=1}^{m} T_k$. We assert a *perfect sampling* process which in theory ranks patterns by precision and recall the same as full materialization does (shown in Lemma 2).

Assertion 2 (Perfect sampling): In perfect sampling, the same proportion (say ρ) of tuples are sampled from each equivalence class. Let $T^S = \cup_{k=1}^{m} T_k^S \subseteq T$ denote the sampled subspace of tuples, where $T_k^S \subseteq T_k$ and $|T_k^S| = \rho |T_k|$. Similarly, let $I_p^S = \{(t^S, p) \in I_p | t^S \in T^S\} \subseteq I_p$. Note that $|I_p^S| = \rho |I_p|$. ■

Lemma 2: With perfect sampling, $\mathcal{P}(p) = \sum_{t_i \in \tau(p)} \mathcal{P}(t_i) \cdot \frac{|I_{t_i p}|}{|I_p|}$ (Eq. 16) $= \sum_{t_i \in \tau(p) \cap T^S} \mathcal{P}(t_i) \cdot \frac{|I_{t_i p}|}{|I_p^S|}$. $\mathcal{R}(p)$ has a similar result.

PROOF: Suppose $t \sim t'$, then $I_t \Leftrightarrow I_{t'} \rightarrow \mathcal{P}(t) = \mathcal{P}(t')$. Thus, $\sum_{t_i \in \tau(p)} \mathcal{P}(t_i) \cdot \frac{|I_{t_i p}|}{|I_p|} = \sum_k \left(\sum_{t_i \in \tau(p) \cap T_k} \mathcal{P}(t_i) \cdot \frac{|I_{t_i p}|}{|I_p|} \right) = \sum_k \left(\rho \sum_{t_i \in \tau(p) \cap T_k} \mathcal{P}(t_i) \cdot \frac{|I_{t_i p}|}{\rho |I_p|} \right) = \sum_k \left(\sum_{t_i \in \tau(p) \cap T_k^S} \mathcal{P}(t_i) \cdot \frac{|I_{t_i p}|}{|I_p^S|} \right)$ $= \sum_{t_i \in \tau(p) \cap T^S} \mathcal{P}(t_i) \cdot \frac{|I_{t_i p}|}{|I_p^S|}$. A similar proof for $\mathcal{R}(p)$ exists. ■

Lemma 2 implies that it is important to sample tuples of different relations in a similar distribution as all tuples, whereas the specific attributes of the sample tuples does not matter (*i.e.*, for two tuples $t \sim t'$, it makes no difference to sample either of them).

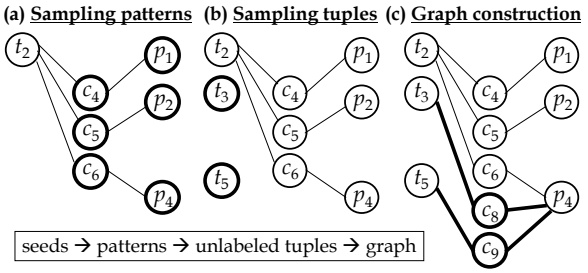


Figure 8: Snapshots of dual partial materialization

For Assertion 1, the Web is well suited as it is a reasonable approximation of the perfect corpus. Thus the remaining challenge is to design a partial materialization approach which fairly approximates the perfect sampling in Assertion 2, as we shall see next.

5.2 Pattern Search

As discussed, we need to partially construct the inference graphs starting only with the seeds– we introduce the dual partial materialization approach. In addition to the seed tuples, this approach requires a search engine S that interfaces with D. For the Web, general purpose keyword-based search engines are sufficient.

PR Duality implies that a tuple can be used to discover more patterns and vice versa. We illustrate key steps of this approach using snapshots in Fig. 8. Starting with the seed t_2 = (Ottawa, Canada), First, we query S with both attributes of t_2 (*i.e.*, Ottawa+Canada) to retrieve K_{seed} documents containing t_2 in D. From the retrieved documents, patterns associated with t_2 are found, namely p_1, p_2, p_4 (bolded in Fig. 8(a)). This conforms to Lemma 1, as all relevant patterns are sampled (p_3 is irrelevant).

Next, we query S with each attribute of the seeds (*i.e.*, query with e_1 and e_2 separately), and apply the newly found extraction patterns to sample unlabeled tuples, namely t_3 and t_5 (bolded in Fig. 8(b)). Querying with attributes of the seeds allows efficient discovery of unlabeled tuples, as it is unscalable to scan through the entire Web. Note that search patterns are not used for this purpose, as they are not yet ranked, thus it would be inefficient to use all of them than to use only the attributes of the seeds. Although t_3 and t_5 both share an attribute with the seed t_2, we stress that the sampling is fair in the sense that different relations are still captured. In particular, t_3 satisfies the relation largest-city-of, so does t_4. Thus it makes no difference in sampling either of them. Similarly, t_5, t_6 satisfy the same relation sister-city-of. This is in line with Lemma 2, which suggests that it only matters to sample different relations, whereas the specific attributes of the sampled tuples are unimportant.

Finally, after sampling patterns and tuples, the tripartite graph can be partially constructed by connecting tuples and patterns via contexts, as shown in Fig. 8(c). The constructed graphs, one for search patterns and one for extraction patterns, subsequently enable PRDualRank to rank the patterns found.

The above description is formally outlined by the algorithm shown in Fig. 9, where line 1–5 correspond to snapshot (a) in Fig. 8, line 6–10 correspond to (b), and line 11–13 correspond to (c).

The outputs are a list of search patterns and a second list of extraction patterns. Each list is ranked with precision as well as recall, both normalized to [0,1], which enables us to rank the patterns by F-scores, the harmonic mean of precision and recall. Given Assertion 1, such learnt patterns are fairly complete, and thus can be reused on other corpora, in particular, the constantly evolving Web.

5.3 Tuple Extraction

The ultimate goal of the patterns is to find more tuples of R from D. *E.g.*, t_1 = (Paris, France) in Fig. 7 is not materialized, and we

Algorithm PatternSearch (T_0, S, E)

Input: T_0, the set of seed tuples.
S, the search engine for D.
E, the extractor.
Output: ranked search & extraction patterns.

1) $P_s \leftarrow \varnothing$; $P_e \leftarrow \varnothing$; $U \leftarrow \varnothing$;
2) **for each** $t = (e_1, e_2) \in T_0$ **do**
3) $D' \leftarrow S$.Query$(e_1 + e_2, K_{seed})$;
4) $P_s \leftarrow P_s \cup${search patterns found in D'};
5) $P_e \leftarrow P_e \cup${extraction patterns found in D'};
endfor
6) **for each** $t = (e_1, e_2) \in T_0$ **do**
7) **for each** e_i **do**
8) $D' \leftarrow S$.Query(e_i, K_{seed});
9) $U \leftarrow U \cup${tuples with attribute $e_i \in E$.Extract(D', P_e)};
endfor
endfor
10) $T \leftarrow$ {top frequent $K_{nolabel}$ tuples $\in U$} $\cup T_0$;
11) $D' \leftarrow \varnothing$;
12) **for each** $t = (e_1, e_2) \in T$ **do** $D' \leftarrow D' \cup S$.Query$(e_1$+$e_2, K_{seed})$;
13) build graphs $G_s = (T, C_s, P_s)$ and $G_e = (T, C_e, P_e)$ based on D';
14) $P_s \leftarrow$ PRDualRank(G_s, T_0); $P_e \leftarrow$ PRDualRank(G_e, T_0);
15) **return** P_s, P_e;

Figure 9: Sketch of pattern search

aim to find it out by utilizing the learnt patterns (p_1, p_2, p_4). By the perfect sampling process described in Assertion 1 and 2, all tuples of R can be found using the patterns learnt.

The learnt patterns consist of two classes, the search patterns and the extraction patterns, for different purposes. For now, we only select top patterns in F-score to achieve a balance between precision and recall. We will study the effect of using precision or recall based patterns by experiments in Sect. 6.3.

Each of the top K_{width} search patterns (search width) is used to retrieve K_{depth} documents (search depth) that may contain instance tuples. Based on our experiments, any single word is unlikely to be focused enough on the Web, thus we additionally require the selected search patterns to contain at least two words.

Next, top K_{ext} extraction patterns are applied on the retrieved documents to extract candidate tuples. We also ignore extraction patterns of precision lower than 0.5, which tend to be too noisy.

Finally, we rank the candidate tuples using top K_{cand} documents containing each candidate. A candidate t is more likely to satisfy R if t instantiates more relevant contexts (captured by its recall), and instantiates a smaller fraction of irrelevant contexts in I_t (captured by its precision). Thus, we consider the F-score in ranking candidates. The score can be computed from search patterns, extraction patterns, or combined from both. We will discuss the different ranking schemes in the experiments.

The above description is sketched by the algorithm in Fig. 10. It has three major components: document retrieval by search patterns (line 1–4), tuple extraction by extraction patterns (line 5–6), and candidate ranking (line 7–9).

6. EXPERIMENTS

We showcase our experimental evaluation of PRDualRank on the real Web across different relations. Overall, the experiments demonstrate that PRDualRank significantly outperforms the baseline in both effectiveness and efficiency.

6.1 Experiment Setting

Corpus. We used the real World Wide Web (D), coupled with Yahoo API[1] as the search engine (S).

[1]http://developer.yahoo.com/search/boss/

Algorithm TupleExtraction (P_s, P_e, S, E)

Input: P_s, the set of ranked search patterns.
P_e, the set of ranked extraction patterns.
S, the search engine for D.
E, the extractor.
Output: a list of tuples ranked by F-score.

1) $P_s \leftarrow$ {top K_{width} patterns by $F_1 \in P_s$};
2) $D_{search} \leftarrow \varnothing$;
3) **for each** $p \in P_s$ **do**
4) $D_{sch} \leftarrow D_{sch} \cup S$.Query$(p, K_{depth})$;
endfor
5) $P_e \leftarrow$ {top K_{ext} patterns by $F_1 \in P_e$};
6) $Cand \leftarrow E$.Extract(D_{sch}, P_e);
7) **for each** $t = (e_1, e_2) \in Cand$ **do**
8) $D' \leftarrow S$.Query$(e_1$+$e_2, K_{cand})$;
9) compute $\mathcal{P}(t)$ and $\mathcal{R}(t)$ and derive its F-score based on D';
endfor
10) **return** $Cand$;

Figure 10: Sketch of tuple extraction

Table 1: Target relations

Relation	Description	Relationship
birth	Birth year of Nobel prize laureates in physics	one-many
capital	Capital city of countries	one-one
area-code	Area code(s) of largest 100 U.S. cities	many-many

Target Relations. Table 1 shows the target relations we used, which are chosen for their different relationship types (*i.e.*, one-one, one-many and many-many). We report the overall performance for all the relations. For finer grained experiments, we only report the results for the capital relation, as we observe similar trends on other relations.

Entity Tagging. We used a dictionary-based approach to recognize entities involved in the target relations. *E.g.*, for the relation capital, we manually prepared a dictionary of all countries in the world, as well as a dictionary of cities using sources like Wikipedia. The same tagging approach is used in PRDualRank and the baseline.

Baseline (QXtract&Snowball, or Q&S). This is the state-of-the-art scalable tuple extraction system for our problem scenario using only a few seed tuples, which lack a principled framework of quality metrics. It is a two-phased system. In the first phase, QXtract [4] finds search queries based on seeds and an extraction system like Snowball [3]. The search queries are then used to retrieve documents as a preprocessing step to allow scalable tuple extraction. In the second phase, Snowball extracts tuples from the documents retrieved. For fairness, we used the same number of documents in learning the queries by QXtract and in ranking the patterns by PRDualRank as well as the same number of queries in QXtract as search patterns in PRDualRank to retrieve the same number of documents from the Web. For other parameters in Q&S, we follow the settings outlined in [4, 3].

Schemes. We also compare the performance of three different schemes of PRDualRank. These schemes only differ in ranking candidate tuples. They are: (i) Dual-Sch, ranking candidates by using search patterns only; (ii) Dual-Ext, ranking by extraction patterns only; (iii) Dual-Combine, which simply takes the average score from both classes of patterns.

Evaluation methodology. Since directly judging the quality of patterns is highly subjective, we resorted to an indirect way which judges the patterns based on the extracted tuples. PRDualRank outputs a ranked list of n tuples. We compare top N tuples from the output with the ground-truth tuples of the target relation. The ground-truth tuples are manually prepared and verified. For each

Table 2: Comparison of optimal F-scores

Relation	Dual-Combine	Dual-Sch	Dual-Ext	Q&S
birth	**0.799**	0.748	0.796	0.485
capital	0.552	0.537	**0.571**	0.398
area-code	**0.859**	0.824	0.853	0.651

$N = 1, \ldots, n$, we evaluate the precision and recall of the set of top N tuples with respect to the ground-truth, and present the result in a precision against recall plot. The different schemes of PRDualRank and the baseline are evaluated in the same manner.

Additionally, there are web pages with a list of ground-truth tuples on the Web. *E.g.*, for capital, there is a list of capitals for every country on about.com[2]. However, we cannot assume such a list exists for any arbitrary relation, thus we should not take advantage of them. For fairness, we treat a web page as a list if it contains more than 10% of all the ground-truth tuples, which is simply disregarded during the tuple extraction phase of every method.

Parameter settings. We discuss the parameters used in dual partial materialization. The larger K_{seed} (number of documents retrieved for each tuple) and $K_{nolabel}$ (number of unlabeled tuples), the more effective pattern learning as it is closer to the perfect corpus in Assertion 1. In practice, as long as they are sufficiently large, the results will be fairly stable and insensitive to their values. Thus we set $K_{seed} = 500$ and $K_{nolabel}$ ten times the number of seeds. On the other hand K_{cand} (number of documents retrieved for each candidate) can be smaller. We set $K_{cand} = 50$, which is large enough to achieve stable ranking of the candidates. Finally, for all experiments, we only used three seeds as user input. Despite the small number of seeds, good performance can be achieved.

For parameters K_{width}, K_{depth} and K_{ext}, we varied them to study their effects on tuple extraction. Otherwise, they are set to default values $K_{width} = K_{ext} = 120$, and $K_{depth} = 700$.

6.2 Overall Performance

Effectiveness. Following the evaluation methodology outlined in Sect. 6.1, the effectiveness of PRDualRank and the baseline is presented as precision against recall plots in Fig. 11.

Observe that all schemes of PRDualRank significantly outperform Q&S (brown color or "4") on all relations. In particular, Dual-Sch (blue color or "3") which only uses search patterns to rank the tuple, is clearly better than Q&S. This implies that our search patterns are good as it not only retrieves the relevant documents, but also ranks tuples, outperforming the combined effort of Q&S. On the other hand, Dual-Ext (green color or "2") is even better, simply because extraction patterns are more tightly coupled with tuples than search patterns. This indicates that the extraction patterns have also performed well in extracting the tuples. Lastly, Dual-Combine (red color or "1") makes use of scores from both classes of patterns. Not surprisingly, this combined approach achieves comparable results as Dual-Ext (and marginally better than Dual-Ext on birth). This implies that we can potentially exploit different classes of patterns for *optimal ranking*, in case that we have no prior information on which class of patterns is the best choice. However a discussion on this issue is beyond the scope of this paper, and we plan to study it as part of our future work.

We also report the optimal F-score achieved by each method in Table 2, which reconfirms the performance comparison. In particular, we improved the optimal F-score by a factor up to 1.64.

It is worth noting that the performance of all methods on capital is worse than the other two relations. The reason is that there can be many different relations on Ω(#city) and Ω(#country). Even the seeds themselves are inherently ambiguous, *e.g.*, they also satisfy the city-of relation.

[2]http://geography.about.com/od/countryinformation/a/capitals.htm

Table 3: Comparison of execution time (in sec)

Relation	Dual-Combine-I	Dual-Combine-II	(total)	Q&S -I	Q&S -II	(total)
birth	247	2474	2721	6124	25208	31332
capital	2580	2639	5219	13664	10020	23684
area-code	384	3243	3627	11093	8708	19801

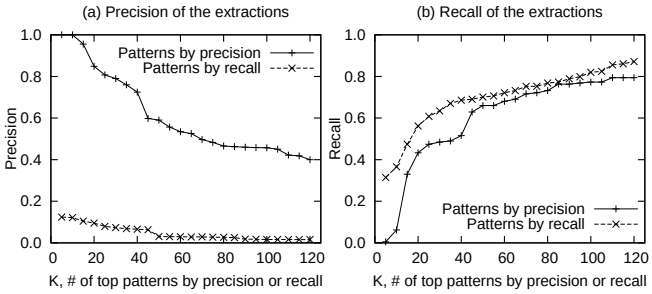


Figure 12: Using top K patterns by precision or by recall

Efficiency. To study the scalability of our approach, we present the execution time required for PRDualRank and Q&S. For different ranking schemes of PRDualRank, Dual-Combine is the slowest. The other two schemes are slightly faster, so we only show the time of Dual-Combine. Both methods are two-phased but in different senses. For PRDualRank Phase I is pattern search and Phase II is tuple extraction. For Q&S, Phase I is the preprocessing step of retrieving relevant documents, and Phase II is tuple extraction. The time for each method is presented in Table 3. They do not include the time needed to download web pages, but include the I/O time in accessing the downloaded local copies. The experiments show that PRDualRank is one order of magnitude faster than Q&S on all three relations. In particular, the pattern search phase of PRDualRank is fast enough to make offline suggestions to DoCQS [17] for content queries. On the other hand, Q&S is slow attributed to the repeated scanning of the documents in each iteration of tuple extraction.

6.3 Precision and Recall

In this experiment, we demonstrate the validity of our metrics–precision and recall. Previously we used a number of top patterns (of both classes) by F-score to achieve a balance between the two metrics. We now compare two other cases on the capital relation: (i) using top K patterns by precision; (ii) using top K patterns by recall. No other changes are involved. With respect to the ground-truth tuples, we compute the precision and recall of the tuples that are extracted using top K patterns by precision for case (i), or by recall for case (ii). The results are presented in Fig. 12 for different values of K. Specifically, Fig. 12(a) compares the precision of the extracted tuples in both cases, whereas Fig. 12(b) compares the recall of the extracted tuples in both cases.

It is not surprising that using top patterns by precision results in better precision of the extracted tuples, as shown in Fig. 12(a). Similarly, it is expected that using top patterns by recall captures more correct tuples, *i.e.*, achieves better recall, as illustrated in Fig. 12(b). This experiment shows that our proposed precision and recall are valid metrics for patterns that "work as intended."

6.4 Effects of Parameters

Lastly, we experiment on the effects of the parameters K_{width}, K_{depth} and K_{ext} for the capital relation, which are shown Fig. 13.

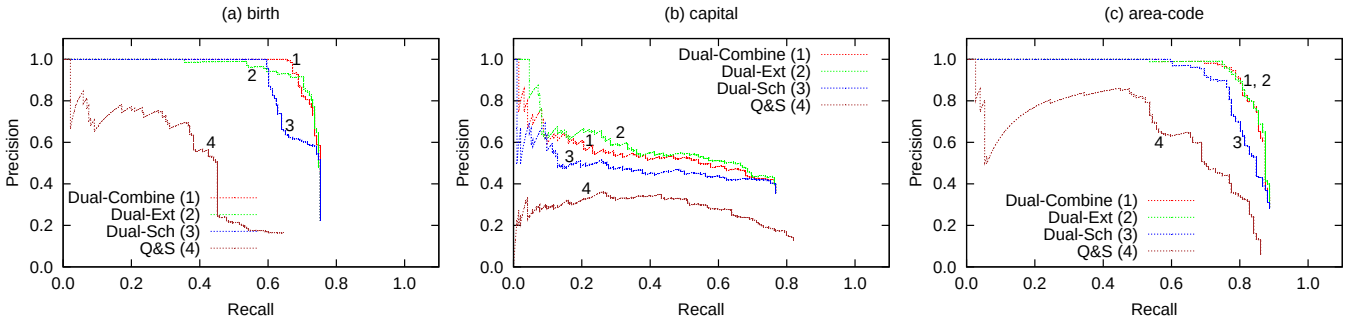


Figure 11: Comparison of PRDualRank and the baseline

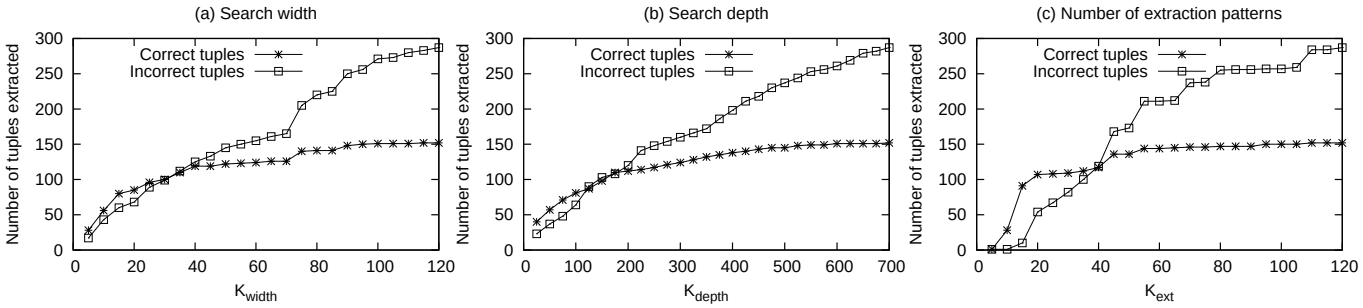


Figure 13: Effects of parameters

We vary each parameter while assigning the other two their respective default values (as in Sect. 6.1). Specifically, the number of correct and incorrect tuples are counted when tuning each parameter. The results clearly demonstrate that the number of correct tuples extracted would converge when the parameters are sufficiently large. Hence, our approach is not sensitive to these parameters as long as they are reasonably large (*e.g.*, the default values).

7. CONCLUSION

In this paper, we discussed the dual problems of pattern search and tuple extraction, while stressing reusable patterns and scalable tuple extraction. To solve the problems in a principled way, we formally "rediscovered" *PR Duality* through the conceptual model PRDualRank with the metrics of precision and recall for both tuples and patterns, as well as developed a concrete framework to achieve a fairly good approximation of the conceptual model. Last but not the least, we evaluated the framework over the Web, which shows that such a principled approach greatly outperforms the previous state-of-the-art system on all three target relations.

8. REFERENCES

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

[2] E. Agichtein. Confidence estimation methods for partially supervised information extraction. In *SDM*, 2006.

[3] E. Agichtein and L. Gravano. *Snowball*: extracting relations from large plain-text collections. In *ACM DL*, pages 85–94, 2000.

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

[5] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. Open information extraction from the web. In *IJCAI*, pages 2670–2676, 2007.

[6] S. Brin. Extracting patterns and relations from the World Wide Web. In *WebDB*, pages 172–183, 1998.

[7] C. L. A. Clarke, G. V. Cormack, and T. R. Lynam. Exploiting redundancy in question answering. In *SIGIR*, pages 358–365, 2001.

[8] S. Dumais, M. Banko, E. Brill, J. Lin, and A. Ng. Web question answering: is more always better? In *SIGIR*, pages 291–298, 2002.

[9] O. Etzioni, M. J. 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*, pages 100–110, 2004.

[10] O. Etzioni, M. J. Cafarella, D. Downey, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates. Methods for domain-independent information extraction from the Web: An experimental comparison. In *AAAI*, pages 391–398, 2004.

[11] M. A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *COLING*, pages 539–545, 1992.

[12] C. C. T. Kwok, O. Etzioni, and D. S. Weld. Scaling question answering to the web. In *WWW*, pages 150–161, 2001.

[13] D. Ravichandran and E. H. Hovy. Learning surface text patterns for a question answering system. In *ACL*, pages 41–47, 2002.

[14] S. Sekine. On-demand information extraction. In *ACL*, 2006.

[15] Y. Shinyama and S. Sekine. Preemptive information extraction using unrestricted relation discovery. In *HLT-NAACL*, 2006.

[16] X. Yin, J. Han, and P. S. Yu. Truth discovery with multiple conflicting information providers on the web. In *SIGKDD*, pages 1048–1052, 2007.

[17] M. Zhou, T. Cheng, and K. C.-C. Chang. Data-oriented content query system: searching for data into text on the web. In *WSDM*, pages 121–130, 2010.