Search engines: from queries to discovery
A learning-theoretic perspective

Arun Sharma

National ICT Australia
Sydney, Australia

Joint work with Eric Martin and Patrick Caldon

Consider the following queries.

- Who was the president of China in 1998?
- How old is the universe?

The answer to the first query is definite.

The answer to the second query is provisional at any given time.

Search engines can assist in answering queries of the first kind.

However, queries of the second kind are mostly unexplored.

The mechanisms for answering the two kinds of queries are likely to be distinct.

Information queries / Discovery queries

Information queries ask for information that can be deduced.

Discovery queries ask for knowledge that can only be induced; at any given time, the answer is only a conjecture.

Data and background knowledge are two ingredients of the discovery process.

- Additional data may cause the current conjecture to be abandoned and replaced by a new one.
- If the discovery process is to be successful, then the correct answer will be found and never abandoned.

We will see that data and background knowledge can also assist in answering complex information queries.

Learning theory is uniquely positioned to provide a perspective on both information retrieval and discovery.

RichProlog offers a canonical implementation of the fundamental concepts of Learning theory in a logical setting.

We want to examine the following issues.

- If we use the web for discovery,
  - what constitutes data?
  - what is background knowledge?
  - what are the queries?
- What could be the role of search engines in the discovery process?

We will see how Learning theory and RichProlog provide a framework for examining these issues.

Σ_1 queries

In the terminology of Logic programming, current search engines answer Σ_1 queries of the form

$$\exists\, \text{page}\ \varphi(\text{page})$$

where $\varphi(\text{page})$ is a boolean combination of atoms of the form

$$\mathit{contains}(\text{page}, \mathit{keyword}).$$

An example is:

$$\exists\ \text{page}[\mathit{contains}(\text{page}, \mathit{indian}) \wedge \mathit{contains}(\text{page}, \mathit{flora}) \wedge \neg\, \mathit{contains}(\text{page}, \mathit{american})]$$

in a first attempt to find information on the flora of India.

Syntactic classification of formulas

The syntactic complexity of a formula in prenex form (with all quantifiers at the front) is given by:

- the leading quantifier;
- the number of alternations of distinct quantifiers.

In particular, with φ denoting a formula without quantifiers:

- $\exists x_1 \ldots \exists x_i \varphi$ is a Σ_1 formula;
- $\forall x_1 \ldots \forall x_i \varphi$ is a Π_1 formula;
- $\exists x_1 \ldots \exists x_i \forall y_1 \ldots \forall y_j \varphi$ is a Σ_2 formula;
- $\forall x_1 \ldots \forall x_i \exists y_1 \ldots \exists y_j \varphi$ is a Π_2 formula;
- $\exists x_1 \ldots \exists x_i \forall y_1 \ldots \forall y_j \exists z_1 \ldots \exists z_k \varphi$ is a Σ_3 formula;
- ...

Σ_2 queries

Discovery requires solving more complex Σ_2 queries. Examples:

$$\exists x[\mathit{vaccine}(x) \wedge \forall y(\mathit{virus\_instance}(y) \rightarrow \mathit{disables}(x, y))]$$

$$\exists x[\mathit{law}(x) \wedge \forall y(\mathit{observed\_datum}(y) \rightarrow \mathit{predicts}(x, y))]$$

Does there exist a book B such that B is a book on Indian flora and for all reviews R, if R is a review on B then R is positive?

$$\exists\ \text{page}[\mathit{contains}(\text{page}, \mathit{indian}) \wedge \exists x(\mathit{contains}(\text{page}, x) \wedge \mathit{book}(x) \wedge \mathit{contains}(\text{page}, \mathit{flora}) \wedge \neg \mathit{contains}(\text{page}, \mathit{american})] \wedge \forall y(\mathit{review\_on}(y, x) \rightarrow \mathit{positive}(y)))]$$

Intended queries

With search engines, it is important to make the distinction between input queries like query and intended queries.

Users will sometimes ask:

Does there exist a web page P such that P contains keyword1 and P does not contain keyword2

when they expect the answer to:

Does there exist a web page P such that

- *P contains keyword1 and P does not contain keyword2;*
- *for all pages Q, if Q contains keyword1 and does not contain keyword2, then Q is no 'better' than P.*

Introduction
Outline
Queries
Learning theory and RichProlog
Discovery from the web
Syntactic classification of queries
Query answering and optimization
Web, data, and background knowledge

More formally

Users will sometimes ask the Σ_1 query:

$$\exists\, \text{page}[\mathit{contains}(\text{page}, \mathit{keyword1}) \wedge \neg \mathit{contains}(\text{page}, \mathit{keyword2}))]$$

when they expect the answer to the Σ_2 query:

$$\exists\, \text{page}[\mathit{contains}(\text{page}, \mathit{keyword1}) \wedge \neg\, \mathit{contains}(\text{page}, \mathit{keyword2}) \wedge \forall\, \text{page'}(\mathit{contains}(\text{page'}, \mathit{keyword1}) \wedge \neg\, \mathit{contains}(\text{page'}, \mathit{keyword2}) \rightarrow \neg\, \mathit{better}(\text{page'}, \text{page}))]$$

Input queries are Σ_1 whereas intended queries are sometimes Σ_2.

Introduction
Outline
Queries
Learning theory and RichProlog
Discovery from the web
Syntactic classification of queries
Query answering and optimization
Web, data, and background knowledge

Computing solutions to intended queries

Existing search engines:

- have a definition of 'better,' based on numbers of hints, numbers of links, number of references, etc.;
- compute all solutions to the input (Σ_1) query but rank them according to their definition of 'better;'
- the user is expected to consider one the first few solutions output as the solution to the intended query.

When 10,000 hits are returned, the user ignores all except the first few ones; the other links are not considered to be a valuable source of information.

Introduction
Outline
Queries
Learning theory and RichProlog
Discovery from the web
Syntactic classification of queries
Query answering and optimization
Web, data, and background knowledge

General Σ_2 queries / optimization (1)

The user has little control over the definition of 'better' employed by the search engine.

The definition of 'better' should not be independent of the keywords input by a particular user.

For instance, 'better' might mean 'cheaper,' for a user who wants to buy a bargain manual car and would like to get an answer to:

$$\exists\, \text{page}[\mathit{contains}(\text{page}, \mathit{car}) \wedge \exists x(\mathit{contains}(\text{page}, x) \wedge \mathit{is\_a\_car}(x) \wedge \mathit{manual}(x) \wedge \forall\, \text{page'}\, \forall y(\mathit{contains}(\text{page'}, y) \wedge \mathit{is\_a\_car}(y) \wedge \mathit{manual}(y) \rightarrow \neg\, \mathit{cheaper}(y, x)))]$$

General Σ_2 queries / optimization (2)

Some search engine applications do address the Σ_2 queries that represent an optimization problem.

Solving a Σ_2 query is not always reducible to computing an optimal solution.

We are interested in Σ_2 queries that go beyond mere optimization.

A query on poetry (1)

Suppose you have heard of a poet who has only written sonnets, but do not remember his name.

> *A sonnet is a 14 line poem divided into $4 + 4 + 4 + 2$ with the rhyme pattern abab cdcd efef gg (plus other syntactic constraints)*

Typing the keyword "sonnet only" provides irrelevant information (103 hits on Google).

A sonnet: *To Science* by Edgar Allan Poe

Science! True daughter of Old Time thou art!
Who alterest all things with thy peering eyes.
Why preyest thou thus upon the poet's heart,
Vulture, whose wings are dull realities?

How should he love thee? or how deem thee wise,
Who wouldst not leave him in his wandering
To seek for treasure in the jewelled skies,
Albeit he soared with an undaunted wing?

Hast thou not dragged Diana from her car?
And driven the Hamadryad from the wood
To seek a shelter in some happier star?
Hast thou not torn the Naiad from her flood,

The Elfin from the green grass, and from me
The summer dream beneath the tamarind tree?

A query on poetry (2)

What you want is an answer to the Σ_2 query:

$$\exists\,\text{page}[\mathit{contains}(\text{page}, \mathit{poem}) \wedge \exists x(\mathit{contains}(\text{page}, x) \wedge \mathit{poet}(x) \wedge \forall\,\text{page}'\,\forall y(\mathit{contains}(\text{page}', y) \wedge \mathit{is\_a\_poem}(y) \wedge \mathit{written\_by}(y, x) \rightarrow \mathit{sonnet}(y)))]$$

A more careful user might rather opt for the query:

$$\exists\,\text{page}[\mathit{contains}(\text{page}, \mathit{poem}) \wedge \exists x(\mathit{contains}(\text{page}, x) \wedge \mathit{poet}(x) \wedge \forall\,\text{page}'\,\forall y \forall t(\mathit{contains}(\text{page}', y) \wedge \mathit{contains}(\text{page}', t) \wedge \mathit{has\_title}(y, t) \wedge \mathit{is\_a\_poem}(y) \wedge \mathit{written\_by}(t, x) \rightarrow \mathit{sonnet}(y)))]$$

Data and background knowledge

When a Σ_1 query is solved by the search engine, the (usually) large number of links returned can be conceived of as a stream of data, useful for answering information or discovery queries.

- For information queries, the answer will be deduced and known to be correct.
- For discovery queries, the answer will be induced, believed, and subject to changes.

In either case, background knowledge is helpful or even necessary.

- The background knowledge will be useful to extract information from the web page.
- Users should be able to input not only queries, but also knowledge.

Decomposing the query

The (first version of the) query suggests a three step approach.

- Solve $\exists$ page *contains*(page, *poem*).

  A standard search will do it and return a large number of links that will play the role of data.
- Solve $\exists \text{page}[\textit{contains}(\text{page}, \textit{poem}) \wedge \exists x(\textit{contains}(\text{page}, x) \wedge \textit{poet}(x))]$

  Poets have to be extracted from the data using some background knowledge.
- Solve the whole query. Sonnets have to be extracted from the data using (another) background knowledge.

General pattern

The general pattern of a discovery query for the web is:

$$\exists \text{page}[\textit{contains}(\text{page}, \textit{keywords}) \wedge \exists \bar{x}(\varphi(\bar{x}) \wedge \forall \bar{y} \chi(\bar{x}, \bar{y}))]$$

Learning theory and RichProlog provide a framework for answering such queries.

Introductory example

A guessing game [Gold, 1967]:

Suppose I have a set of sets of numbers:
$\{\{1, 2, 3, \ldots\}, \{2, 3, 4, \ldots\}, \{3, 4, 5, \ldots\}, \{4, 5, 6 \ldots\}, \ldots\}$.

Suppose I pick $\{3, 4, 5, 6, 7, \ldots\}$.

I'll tell you a number in the set, and you have to guess the set.

- I tell you 10 - you guess $\{10, 11, 12, \ldots\}$
- I tell you 5 - you guess $\{5, 6, 7, \ldots\}$
- I tell you 7 - you still guess $\{5, 6, 7, \ldots\}$

  $\vdots$
- I tell you 3 - you still guess $\{3, 4, 5, \ldots\}$

  $\vdots$

Correctness in the limit

A good strategy:

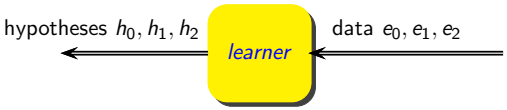
> *Guess the set whose least element is the lowest number thus far presented*

There's no finite set which allows you to draw this conclusion with certainty, unless the smallest number is 1.

On a more technical note:

- In classical logic, compactness allows to know with certainty that a conclusion is correct from finite data.
- In Learning theory, we can only hypothesize that a conclusion is correct from finite data, without knowing with certainty. Correctness comes in the limit.

Identification in the Limit



The sequence of hypotheses $h_0, h_1, h_2, \ldots$ output by the learner must eventually converge to a correct description of the world that generates the sequence of data $e_0, e_1, e_2, \ldots$.

Logical connection

The scenario of learning in the limit can be cast in a logical setting for answering Σ_2 queries.

Learning in the limit corresponds to:

- guessing a witness to existentially quantified variables in a Σ_2 query
- before going through a limiting (noncompact) refutation stage involving the universally quantified variables.

RichProlog is to this logical setting what Prolog is to classical logic.

Solving queries: the Prolog approach

Axiomatize the problem: Write down a set of clauses A and a sentence ψ, that represent knowledge about the reality being modeled.

Solve the problem: Prove ψ from A.

This approach has many limitations. In particular:

- A represents axiomatic knowledge only, not data.
- ψ has to be a Σ_1 sentence; but we also want to be able to answer Σ_2 queries.

RichProlog programs

Predicates are partitioned into theoretical and observational.

For instance, *father*(x,y) and *price*(x,y) can be observational, whereas *ancestor*(x,y) and *cheap*(x) can be theoretical.

A RichProlog program consists of:

- a (possibly infinite) stream of data, built from the observational predicates—data can be positive only, negative only, or of either kind;
- some finite background knowledge, consisting of rules whose body is built from either observational or theoretical predicates, and whose head is built from a theoretical predicate.

A toy example of RichProlog program

Take an observational predicate *parent_of*(x,y) and theoretical predicates including *ancestor*(x,y) and *ideal_wife*(x).

Data are positive only. They come from genealogy books and are facts of the form:

parent_of(*adam*,*seth*).
parent_of(*seth*,*enos*).
...

The background knowledge consists of:

ancestor(X,Y) :- *parent_of*(X,Y).
ancestor(X,Z) :- *parent_of*(X,Y), *ancestor*(Y,Z).

plus a (long) list of rules defining *ideal_wife*(x).

RichProlog queries

The most general RichProlog queries are sentences of the form

$$\exists \bar{x}(\varphi(\bar{x}) \wedge \neg \exists \bar{y} \chi(\bar{x}, \bar{y}))$$

where $\varphi(\bar{x})$ and $\chi(\bar{x}, \bar{y})$ are restricted boolean combinations of atomic formulas built from either theoretical or observational predicates.

The syntax of RichProlog queries is not *ad hoc* but is imposed by results from Learning theory: any solvable discovery query can be put in this form.

A toy example of RichProlog query

Suppose Frank is looking for a wife.

- Frank has a long list of qualities that make an ideal wife.
- Franks wants to make sure that his ideal wife is not related to him by blood.

The following RichProlog query captures those requirements:

$$\exists x(\mathit{ideal\_wife}(x) \wedge \neg \exists y(\mathit{ancestor}(y, x) \wedge \mathit{ancestor}(y, \mathit{frank})))$$

Strategy

To solve $\exists \bar{x}(\varphi(\bar{x}) \wedge \neg \exists \bar{y} \chi(\bar{x}, \bar{y}))$, RichProlog applies a generate and test strategy:

- generate a possible witness $\bar{t}$ such that $\varphi(\bar{t})$ can be deduced from the background knowledge plus the (finite) set of available data;
- test whether $\bar{t}$ is correct, by trying to prove $\exists \bar{y} \chi(\bar{t}, \bar{y})$ from the background knowledge plus the (finite) set of available data.

Testing whether $\bar{t}$ is correct is a refutation procedure, that will discard any incorrect witness in the limit, when enough data are available.

Example

When trying to find an ideal wife, data and background are used to find a woman who has all the attributes of an ideal woman, solving the Σ_1 query $Q_D = \exists x\, ideal\_wife(x)$.

If Q_D succeeds, the result is a woman whose name is *name1*.

Using background knowledge, RichProlog tries and solve the Σ_1 query $Q_I = \exists y(ancestor(y, name1) \wedge ancestor(y, frank))$

The name *name1* is retained as long are Q_I does not succeed.

If Q_I succeeds, the computation backtracks to find alternative solutions to Q_D.

Generating possible poets

When Google answers the first part of the query, which is $\exists$ page $contains(\text{page}, poem)$, it returns 7,950,000 hits.

Using some background knowledge:

- author names start with capital lettters;
- author names are preceded with by;
- ...

possible witnesses to x for the query

$$\exists \text{page}[contains(\text{page}, poem) \wedge \exists x(contains(\text{page}, x) \wedge poet(x))]$$

will be generated.

Refuting with poems

More background knowledge can be used to solve the remaining part of the query, by refuting the current witness.

Indeed, sonnets can easily be characterized using syntactic rules.

William Shakespeare wrote about 154 sonnets, and will be one of the first witnesses being generated. The query will find numerous Shakespearian sonnets, initially confirming this hypothesis.

Though we might think that he is a solution to the query, the poem *A Lover's Complaint* eventually refutes that guess.

Sir Philip Sidney is one of the answers that can be discovered in the limit.

Building queries

Users are not experts in logic. They input keywords from the which a Σ_1 query is built; they do not input Σ_1 queries.

How will users express their discovery queries?

Σ_2 queries for optimization problems can be built automatically. A user who wants to buy a bargain manual car can input *car* and *manual*, ask to optimize according to first keyword and one of a few predefined keywords like *cheap*, and get an answer to:

$$\exists\,\text{page}[\mathit{contains}(\text{page}, \mathit{car}) \wedge \exists x(\mathit{contains}(\text{page}, x) \wedge \mathit{is\_a\_car}(x) \wedge \mathit{manual}(x) \wedge \forall\,\text{page'}\,\forall y(\mathit{contains}(\text{page'}, y) \wedge \mathit{is\_a\_car}(y) \wedge \mathit{manual}(y) \rightarrow \neg\,\mathit{cheaper}(y, x)))]$$

What about more general Σ_2 queries?

Variety of domains

Σ_2 queries contain formulas of the form $\chi(\bar{x}, \bar{y})$ for existentially quantified $\bar{x}$ and universally quantified $\bar{y}$.

This raises two main issues:

The first issue is that variables in $\bar{x}$ and variables in $\bar{y}$ vary over different domains.

With standard queries the domain is the set of web pages. To exploit information and get answers to more complex queries, arbitrary domains are needed.

In full generality, variables will have to vary over the set of web pages as well as over sets of concepts that make up the content of these pages.

Information extraction

The second issue is that $\chi(\bar{x}, \bar{y})$ is a relation between at least two pieces of information, not a property. Working with predicates of arity 2 at least is much more difficult than working with predicates of arity 1.

For example, the main difficulty to answer the query

does there exist a web page P and a book on Indian flora B such that:

- *B has occurrences in P, and*
- *all reviews R on B are positive.*

is to extract the information that R is a review of B.

Background knowledge

Data are the result of a basic search. But the background knowledge will usually be given by the user.

Searching is an activity that can update the background knowledge.

- How will users input their background knowledge?
- How will search engines interact with the system that exploits the background knowledge?
- Searching is an activity that can update knowledge. How can we convert the result of searching into useful background knowledge?
- How can we improve the quality of the background knowledge?

Conclusion (1)

- We exploit very little of what is returned as the result of a search: we consider the first 10 results and ignore the remaining 10,000 links that have also been returned.
- We would like to use these links as data to help us answer discovery queries.
- We would like to ask more complex queries, whose solutions could only be computed in the limit.
- Learning theory and RichProlog provide a framework to reflect on these issues.

Conclusion (2)

- Users should be able to input not only queries, but also knowledge.
- Knowledge is as essential as data to answer discovery queries, but even information queries can benefit from knowledge.
- Knowledge can be shared, traded.
- Information extraction is one of the main challenges.