Deep web search: an overview and roadmap

K. Tjin-Kam-Jet

University of Twente,
Enschede, the Netherlands
E-mail: tjinkamj@ewi.utwente.nl

D. Trieschnigg

University of Twente,
Enschede, the Netherlands
E-mail: trieschn@ewi.utwente.nl

D. Hiemstra

University of Twente,
Enschede, the Netherlands
E-mail: hiemstra@ewi.utwente.nl

Abstract: We review the state-of-the-art in deep web search and propose a novel classification scheme to better compare deep web search systems. The current binary classification (surfacing versus virtual integration) hides a number of implicit decisions that must be made by a developer. We make these decisions explicit by distinguishing 7 system aspects that describe a system in terms of its functionality (what it can, and what it cannot do) and in terms of its solution to a specific problem. We then motivate the need for a search system which has a single-field free-text query interface that supports real-time structured search over multiple sources. To this end, we discuss two possible federated architectures and state the scientific challenges. Finally, we present the findings of our ongoing project and briefly outline related work to free-text interfaces over structured data.

Keywords: review, survey, deep web, deep web search, interfaces, OneBox, natural language, free text, surfacing

1 Introduction

Most internet users are used to finding Web information with search engines such as Google, Bing and Yahoo. These search engines provide a single point of entry to the *surface web*, the part of the internet which can be discovered by following hyperlinks (Bergman 2001). Typically these search engines *crawl* the surface web:

hyperlinks are followed and snapshots of pages are downloaded and indexed. Search results provided by the search engine are based on this local index.

Perhaps an even larger amount of information is available on the *deep web*, the part of the internet which cannot be discovered by simply following hyperlinks. One can think of information stored in structured databases that can only be accessed by submitting a web form (Bergman 2001, Chang et al. 2004), such as product search websites and online library catalogues. A second group of deep web information is provided by web applications which give real-time information based on a particular user request. Examples are online travel planners and different kinds of booking systems. Information from such systems can be very dynamic in nature: the same request issued at different moments in time may result in different information pages. Note that, like Amazon.com, websites can provide a hyperlink structure on all database items to accomodate surface web search engines such that the items may still be crawled. However, this will not guarantee that search engines will always have the current price or the current items in stock.

In this paper, we will explore *deep web search systems*: search engines which provide access to (third-party) deep web content. These systems need to deal with the structured and dynamic nature of deep web content. We make the following contributions. First, we present a high-level overview of existing approaches to deep web search. Second, we present a classification scheme of deep web search systems and discuss the advantages and disadvantages of the different types of systems. Third, we present our vision and ongoing work in the Federated OneBox project: an attempt to overcome some of the limitations of current deep web search systems.

The overview of this papers is as follows. In Section 2 we provide a background of approaches to deep web search. In Section 3 we present a novel classification scheme of deep web search systems and use it to classify a variety of existing systems. In Section 4 we introduce and motivate Federated OneBox, a project to improve the state-of-the-art in deep web search. In Section 5 we describe the current state of the project. In Section 6 we summarize and conclude this paper.

2 Background

There are three reasons why we need a deep web search engine. First, as mentioned in the introduction, the deep web is larger than the surface web, and search engines cannot effectively search the deep web. Second, unless someone knows the name of a deep (source) site, it is difficult to find the deep source itself with current search engines. A user is compelled to use the interface of a source site, not only because he can thereby access the site's deep content, but also because he can pose structured queries, which he cannot do with current search engines. Third, each deep site has its own interface. Therefore, a user who wants to compare items from different sites must repeatedly enter the same query in a different interface, which is tiresome. A deep web search engine with one single interface that provides structured search over all deep sources would be very valuable: it would serve as a *single point of entry* to the deep web.

Two steps are needed to create a deep web search engine. The first step towards a deep web search engine, is understanding the interface to the deep web (which is typically a web form, often consisting of multiple input fields). The

communication (e.g., of web form data) between client browser and web server has been standardized by the W3C[1]; however, there is no standard for interface *design*. Web developers are free to place or not to place labels that explain the kind of data that should be entered in each input field. They may use radio buttons, check boxes, and select menus; there is no pre-defined meaning for these control elements. We rely on common sense of the web developer to design, and the user to understand the web form and interact in a fruitful way. In an attempt to automatically understand query interfaces, the existence of a hidden syntax that guides the creation of a query interface is hypothesized in (Zhang et al. 2004); interface clustering is explored in (He et al. 2004*a*, Zhao et al. 2008, Wu et al. 2004); and methods for integrating interfaces of a similar domain are explored in (He et al. 2004*b*, 2005) (a recent survey of query interface understanding can be found in (Khare et al. 2010)).

For the second step to create a deep web search engine, there are two alternatives: extracting and indexing sample data from data sources, or extracting and matching interface schemas to build mediated interfaces. These steps have adopted the names (deep web) *surfacing* and *virtual integration* and will be discussed in the following sections, respectively.

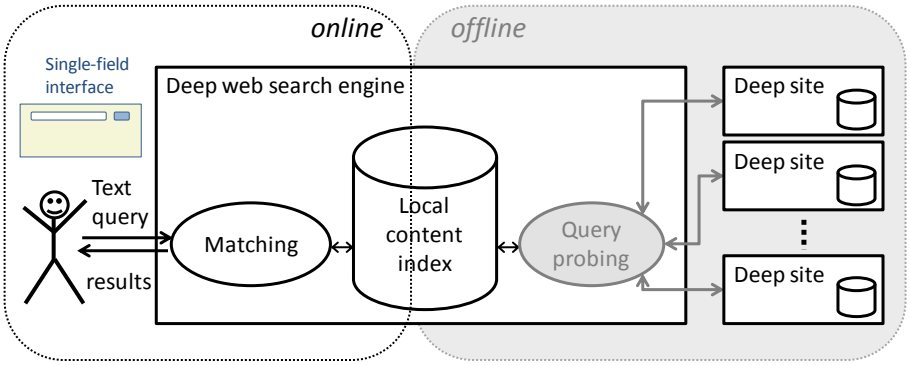
2.1 Surfacing

In surfacing (Raghavan & Garcia-Molina 2001, Álvarez et al. 2007, Barbosa & Freire 2007, Wu et al. 2006, Madhavan et al. 2008), web forms are automatically submitted with "guessed" input field values, and the resulting page is indexed like a normal web page. Surfacing has two important aspects: *i*) the information is duplicated locally at the search engine, and *ii*) the deep sites can be of any domain (e.g., travelling, medical practice, arts). Some disadvantages of surfacing are that: efficient "guessing" is a challenging task (Cafarella et al. 2008*b*), even for plain text data sources (Callan & Connell 2001); maximizing databse coverage while minimizing query traffic is challenging (Wu et al. 2006, Madhavan et al. 2008); it may lose semantics (Madhavan et al. 2009); and, it is effective only for certain kinds of deep content, e.g., not for real-time content. The biggest advantage is that this approach can be coalesced in the existing infrastructure of a search engine, allowing it to scale to web proportions. This also allows for a single point of entry to both traditional web search results and deep web search results.

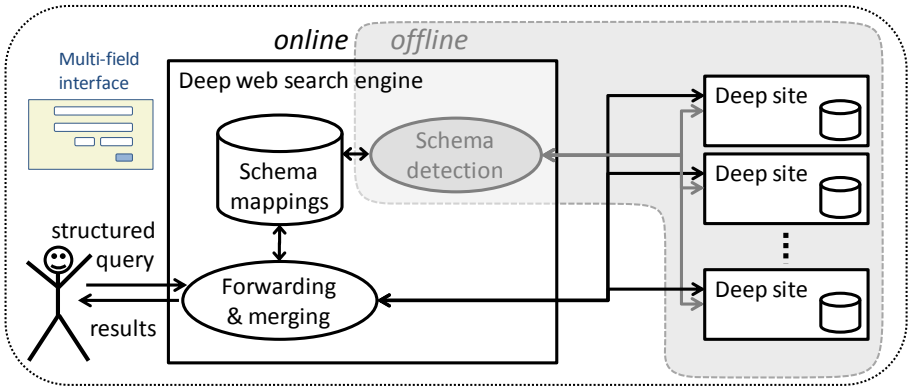
Figure 1a illustrates the surfacing approach and its offline and online processes. The offline process (depicted in grey) probes the deep sites by repeatedly submititng web forms with guessed values for the input fields. The deep sites respond with web pages that possibly contain search results, and those pages are stored in the search engine's index. The online process accepts and matches the user query against the local index and returns results from this index.

2.2 Virtual integration

In virtual integration (Dragut et al. 2009*b*, Madhavan et al. 2009), deep sites are treated as data sources that must be integrated in a larger system. A user queries the system through a web form with multiple input fields, which reflects a mediated schema over related sources. The system selects which sources to query, forwards the query, gathers and merges the results, and presents the results to the user. Virtual



(a) Schematic overview of surfacing. Offline, snapshots of the content of deep sites are taken and stored in a local index. Online, the user queries the system using a single-field interface. The query is matched against the index and local results are returned to the user.



(b) Schematic overview of virtual integration. Offline, the schemas of deep sites are detected and a mediated interface is built. Online, the user queries the system using a multi-field interface. The query is forwarded to the deep sites and the results are returned to the user.

Figure 1: Surfacing versus virtual integration

integration has two important aspects: *i*) the information is kept remotely at the deep sites, and *ii*) the deep sites must be of the same domain. The disadvantages of this approach are that the creation of mediated web forms, source selection, and results merging, are very challenging in a constantly changing environment like the web (e.g., sources update their schema, change their interfaces, or sources join or leave the system). Perhaps the biggest disadvantage is that this approach does not scale to web proportions, despite the advancements in automatic interface extraction and schema mapping. However, when the number of sources remains manageable, the advantages are that the system supports structured queries, and that it is applicable to all kinds of deep content.

Figure 1b illustrates the virtual integration approach and its offline and online processes. The offline process (depicted in grey) detects the schemas of (the query interfaces of) the deep sites and stores them in a database. It then uses this schema index to build a mediated interface (that is, it knows exactly how each field of the

mediated interface maps to some field of a deep site). The online process forwards the user query to the deep sites, downloads and merges the content of the disparate sources, and presents the results to the user.

3 A novel classification of deep web search systems

The existing classification between surfacing and virtual integration only considers whether systems try to retrieve and index deep content (surfacing), or whether they forward the query to deep sources through a mediated interface (virtual integration). However, we have observed many aspects in which systems can vary. The binary classification (surfacing versus virtual integration) does not justify this observation. When a web developer chooses either surfacing or virtual integration, he or she makes a number of implicit decisions. We make these decisions explicit by distinguishing 7 system aspects. Although some aspects are today typically associated with either surfacing or virtual integration, we describe the aspects independently of the two deep web search approaches.

In the next subsections, we describe the aspects and try to explain their (dis)advantages in isolation of each other. Then, we show how existing systems can be classified according to this new classification scheme (see also Table 2 on page 8).

3.1 System aspects

Based on our observation of the differences amongst deep web search systems, e.g., their goals, their research challenges, and their solutions; we propose seven aspects in which these systems can be categorized. The aspects are mainly functional or technical, corresponding with the system's goal or its solution to a specific problem, respectively.

Index location. We distinguish between a *local* index or a *remote* index. A search engine can build a local index of deep content and serve results from this index. Alternatively, it can forward the query to the remote deep site, and (thereby "using the remote index" to) show real-time results. The distinction between local versus remote greatly resembles that of surfacing versus virtual integration, respectively. As such, the (dis)advantages of the methods largely coincide with those of their surfacing or virtual integration counterparts. Further, the choice of local or remote impacts the type of retrievable content.

Type of retrievable content. We distinguish between *static*, *dynamic*, and *real-time* content: *i*) static content is not likely to change over time, e.g., like a news article; *ii*) dynamic content changes over time, e.g., like the price of a product; and, *iii*) real-time content is either computationally derived, e.g., like a web-based calculator that gives the answer to some given equation, or it is sensory information, like travel delay information, information about items in stock, or room availability .

An index contains content that was previously gathered. Therefore, mostly static and dynamic content can be served from an index. An index could in theory contain up-to-date real-time content, if the content was crawled

and indexed just before the query was issued. However, it is not a reliable way to show real-time content: forwarding the query to the deep source and displaying those results is the safest way to show real-time content.

Data acquisition. We distinguish between: *crawling*, *scraping*, *surfacing*, and using an *API*. A search engine uses crawlers to download web pages (these pages are typically not part of the deep web). By using a standard meta-data markup scheme (e.g., such as defined on `http://www.schema.org/`), a web site can aid the crawler in identifying meaningful parts of the page, and even indicate structured data. Alternatively, when a site does not use such a markup scheme, scrapers can be used to extract useful (structured) data from web pages. Deep web data is either: *i*) surfaced (and possibly scraped) by the search engine; *ii*) made publicly available by a deep site through the site's API (such that any search engine can pro-actively download the data); or, *iii*) uploaded to the engine by a deep site, using an API of the search engine.

Domains and sources. We distinguish between *single* or *multiple* topical domains (e.g., travel planning, hotel booking, or car rental), and *single* or *multiple* sources. A search system serves results from one or more sources which can be from the same domain, or they could be from multiple domains. A system that supports querying over multiple domains would be more favorable, since it would provide a single point of entry to all sorts of systems. However, regarding structured search, it will become a daunting task to provide structured search as the number of sources in one domain increases to web-scale. Providing structured search to multiple sources in multiple domains on a web-scale is a big open problem.

Type of search. We distinguish between (also) supporting *structured* queries or only supporting *keyword* queries. Deep content is traditionally accessed by submitting a web form that usually consists of multiple input fields. Such a web form forces us to enter structured queries, thereby allowing focussed retrieval. As a consequence, we expect focussed results. For example, the search results for a notebook costing less than 200 dollars, should only contain notebooks (so no mobile phones, game consoles, or desktop computers) which are less than 200 dollars (so no popular notebook for 299 dollars). Though the example might seem obvious and rather simplistic, a keyword based retrieval system might actually produce such erroneous results.

Type of search refers to whether or not a user can pose structured queries and expect focussed results, as in our example. This is regardless of the type of search interface used.

Search interface. We distinguish between a *single-field* (free-text) interface, or a *multi-field* interface. The single-field interface has many benefits, it is easy to use, users are accustomed to it, and users prefer it over a web form with multiple input fields. The challenge is that, at query-time, the free-text query must be interpreted and translated to a structured query if possible, or, the query should be treated as a keyword query. In a multi-field interface, each field of the search interface maps to a specific field of the deep site's interface. While this significantly reduces the processing steps required at query-time,

automatically creating and maintaining the multi-field interface in the first place is challenging. Also, multi-field interfaces have the disadvantage that separate interfaces must be maintained per domain since each domain has its own set of "generic fields".

Results interface. We distinguish between a *remote* results interface, a *local-static* interface, or a *local-interactive* interface. If, after a query has been submitted, a search system immediately redirects the user to the deep site containing the most likely result, then the system uses a remote results interface. If the system just displays a list of results so that the user may choose the deep site from which he wants to view the results, then the system uses a local-static interface. If the system provides additional capabilities like sorting and filtering based on specific attributes (e.g., size, color, or price), then the system uses a local-interactive interface.

3.2 Systems

In Table 1 we classify various (deep web search) systems according to the aspects mentioned in the previous section. We do not claim that this list of systems is exhaustive; however, it includes enough different systems to give an overview of current solutions to deep web search.

HiWE (Raghavan & Garcia-Molina 2001) and DeepBot (Álvarez et al. 2007), the first two systems in Table 1, are actually just crawlers and are not complete search systems. However, the kind of results we could expect with a hypothetical search system on top of the crawled data would be dynamic in nature; this hypothetical finding is indicated with parenthesis in the table.

The WebTables (Omnivore) project (Cafarella et al. 2008*a*, Cafarella 2009) extracts structured information from tables (i.e., indicated with the `<table>` HTML tag) residing in web pages found in the Google index. It is not entirely clear what kind of structured queries are supported, and what kind of query and results interfaces are used. According to the authors, queries may contain `spatial` operators (e.g, `samecol` and `samerow`, which only return results if the search terms appear in cells in the same column or row of the table) and query-appropriate visialization methods are used to render the results.

Google also surfaces deep content (Madhavan et al. 2008, Cafarella et al. 2008*b*), but, to our knowledge, the system does not support structured (key-value) queries; instead, the standard keyword index is used. Since the results are served from a local index, the system can serve both static and dynamic results, but it cannot serve real-time results.

"Item search"[2] refers to structured (key-value) search that is enabled because: *i*) the search engine supports structured queries; and *ii*) the (surface or deep web) data is published in an open standard, or is delivered via an API. Examples of item search systems include Bing product search, Yahoo shopping, and PriceRunner. These systems require an index since some data may be crawled from the surface web. Also, they support structured search by means of *facets*. For example, in PriceRunner, the search results for the keywords "digital camera" can be narrowed down further by facets like manufacturer, effective pixels, and optical zoom. Structured queries can *only* be specified by making use of the facets, e.g., typing "digital camera, manufacturer: Sony" in the search field does not yield the same

Table 1 A classification of deep web search systems. Items between parenthesis are hypothetical and are not explicitly reported in the original paper(s).

		Index loc.	Data acq.	Sources & dom.	Search type	Search interface	Results interface	Retriev. results
Surfacing	HiWE	L	2,3	B,2	-	-	-	(2)
	DeepBot	L	2	B,2	-	-	-	(2)
	WebTables	L	3	B,2	(1)	(1)	(2)	1,2
	Google surfacing	L	2	B,2	2	1	2	1,2
	"Item search"	L	1b,4a	B,2	(1,2)	1	3	1,2
Virtual integration	Flight planners	R	(3,4b)	A,2	1	2	3	2,3
	MetaQuerier	R	(3,4b)	A,1	1	2	(2)	2,3
	WISE-integrator	R	(3,4b)	A,2	1	2	(2)	2,3
	VisQI	R	(3,4b)	A,2	1	2	(2)	2,3
	OneBox	R	4a,4b	A,1	1	1	1,2	2,3
	Federated OneBox	R	4a,4b	B,2	1	1	1,2	2,3

Table 2 Legend explaining the aspects and values in Table 1.

Aspect	Short description
Index location	
R - Remote	Search engine shows real-time results from remote data source(s)
L - Local	Search engine shows results from local index
Data acquisition	
1a - Crawling	Search engine downloads web pages by following hyper links
1b - Standard	Web pages contain meta-data markup that is known to the crawler
2 - Surfacing	Search engine surfaces web pages by submitting web forms
3 - Scraping	Search engine extracts structured records from web pages
4a - API-1	Web site uploads structured data to a search engine's API
4b - API-2	Search engine downloads structured data from a deep site's API
Sources and domains	
1 - Single	Search engine can only return answers from one source
2 - Multiple	Search engine can return answers from multiple source
A - Single	All sources are from the same domain
B - Multiple	Sources are or can be from different domains
Search type	
1 - Structured	Search engine supports structured (key-value) queries
2 - Non-structured	Search engine supports basic keyword queries
Search interface	
1 - Single field	Search interface consists of single text field
2 - Multiple fields	Search interface has multiple input fields
Results interface	
1 - Remote	Results are accessed and displayed from the original source
2 - Local-static	Result summaries are simply displayed at the search engine
3 - Local-interactive	Results can be filtered or sorted on different attributes
Retrievable results	
1 - Static	Content is not likely to change over time
2 - Dynamic	Content is very likely to (repeatedly) change over time
3 - Real-time	Content is the real-time result of a (proprietary) web application

results as typing "digital camera" and choosing "Sony" for the manufacturer facet. Note that the structured queries are only possible *after* issuing a keyword query, because the facets are only shown in the results interface. The initial query interface only shows a keyword input field.

A flight planner[3] brokers over many airline sites and shows real-time flight results. A multi-field search interface allows users to enter structured queries. The available flights are shown in a local-interactive interface allowing the user to refine the results and easily compare results from different sources.

MetaQuerier (He et al. 2005) translates, on-the-fly, a query expressed in one interface to a set of queries in a target interface. Translation can take place without specifically prepared translation knowledge for a source; so it should be applicable over various domains. However, both source and target interfaces should be from the same domain. The interface in which the results are shown is not defined.

WISE-integrator (He et al. 2004*b*) automatically creates a unified interface for a group of web forms of the same domain. Based on the visual layout of a web form, it extracts attributes which are used to match and integrate multiple interfaces into a unified interface. The unified interface consists of multiple fields, so users can pose structured queries. It is unclear how results are presented to the user.

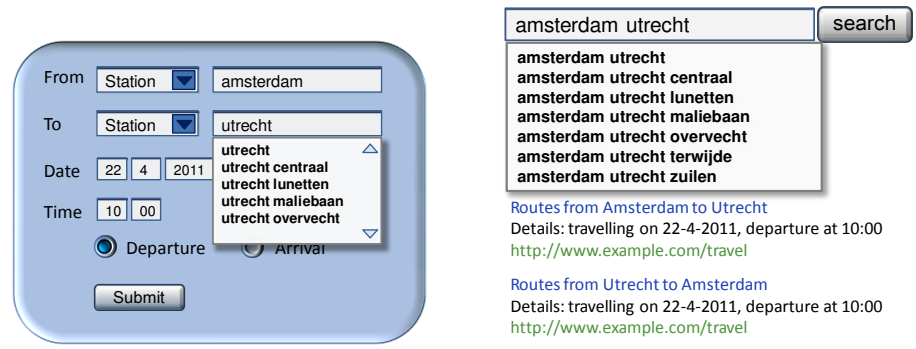
VisQI (Dragut et al. 2009*a*,*b*, Kabisch et al. 2010) also automatically creates unified interfaces for groups of web forms of the same domain. It adopts a hierarchical representation of query interfaces, and it outperforms previous approaches (on extracting query interfaces) with about 6.5%. Users can pose structured queries, but it is unclear how results are presented to the user.

OneBoxwas introduced as a free-text interface on top of web form of a deep site. It consists of a single input field in which queries can be phrased in a natural language fashion, and it support structured search. In the next section, we illustrate the functionality of OneBox by means of an example. We then describe the larger and ongoing Federated OneBox research project.

4 Federated OneBox – a single text search box to search the deep web

The functionality of a OneBox can be illustrated by the following example. Consider the mutli-field web form in Figure 2a, in which a user can enter structured queries. The single-field interface in Figure 2b allows a user to enter free-text queries, which are then interpreted by the system. A *query interpretation* effectively represents a filled out version of the web form shown in Figure 2a, and can be displayed like a standard web search result, as depicted at the bottom of Figure 2b. By clicking on a result, the user "submits the filled out web form" and gains access to the deep content.

OneBox is part of a larger Federated OneBox research project in which we envision a system where users can search both the deep and the surface web using just one single text box interface, e.g., like the interface in Figure 2b. It serves as a single entry point to the complete web, offering both structured and unstructured search. In relation to the classification shown in Table 3, Federated OneBox contains aspects that are normally found in surfacing approaches (e.g., a single text search interface to all data, searching of many sources and domains), but in essence it is a virtual integration approach (e.g., remote indices, dynamic and real-time results); as such, it combines the best of both worlds. A Federated OneBox system would be the only one to offer multi-domain, structured search capabilities through a uniform, single-field search interface. It would alleviate the user from first



(a) A multi-field web form that supports structured search and offers query suggestions.

(b) A single-field interface that also supports structured search and offers query suggestions.

Figure 2: Multi-field and single-field interfaces offering structured search.

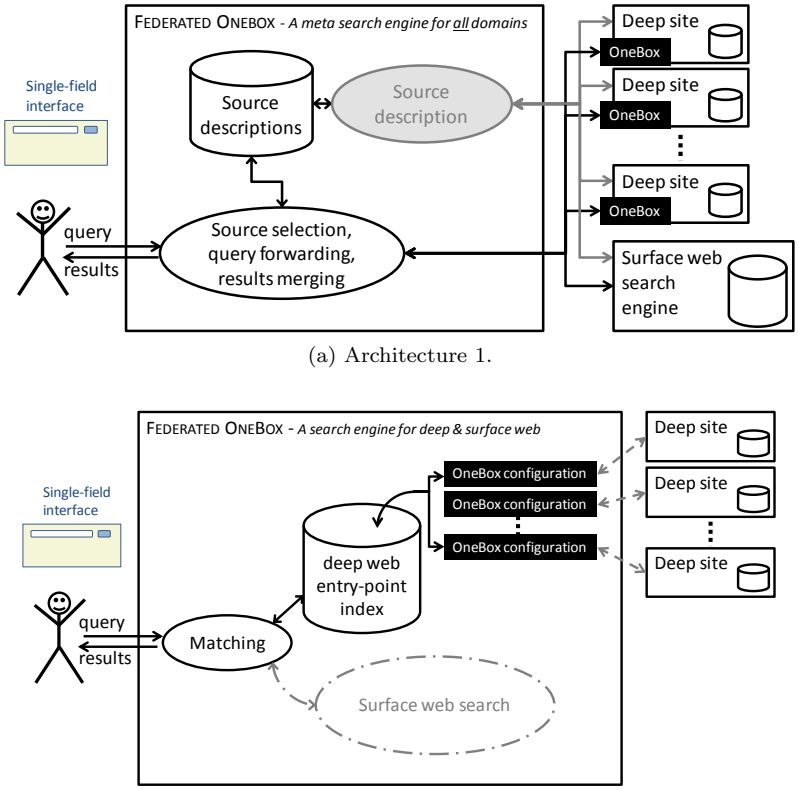
finding relevant (deep) sources, then selecting which sources to use, and finally from having to reformulate the query for each source. When a user enters a structured information need, the system will find relevant sources, rank them, and rephrase the information need in the query format of each selected source.

In this research project, we investigate techniques towards enabling such a Federated OneBox system. Next, we discuss the high-level architectural solutions, and we outline the scientific challenges in this project.

4.1 A federated architecture

Deep sites are autonomous, independent, and offer site-specific search capabilities. We would like to somehow harness the combined (site-specific) search capability over all these sites. A centralized approach which crawls and indexes all data, cannot offer this combined (site-specific) search capability. Therefore, we are motivated to research federated search architectures. First, deep sites can be viewed as participants in a *dataspace* (Franklin et al. 2005, Halevy et al. 2006). Key ideas of a dataspace are that participants co-exist, there is no central coordination, and they all provide support for, at least, text querying. Second, a single entry-point to search over all the participants can be provided by means of a search *broker*. Having a broker and multiple participants, is essentially a federated architecture (Callan 2000, Si & Callan 2005).

There are two main architectures in which a Federated OneBox system can be realized. In the first architecture, shown in Figure 3a, a OneBox is placed as an abstraction layer on top of the local search interface of a deep web site. A OneBox is configured for each deep site to provide a free-text search interface over a multi-field search interface: each site then supports local structured search by means of a free-text query. This in turn simplifies federated search in two ways. First, there is no need for a mediated schema per domain, because all sources of all domains have a single field, free-text, interface. Second, no query-transformation is needed at the broker: a free-text query can be forwarded as is. This architecture procures surface web search by making use of the keyword search capabilities of surface web search engines; and, it procures (structured) deep web search by making use of the free-text



(a) Architecture 1.

(b) Architecture 2.

Figure 3: Alternative architectures for FEDERATED ONEBOX.

search capabilities of the OneBox at each deep site. Note that each deep site can configure its own OneBox, this does not need to be configured by the FEDERATED ONEBOX. Finally, the FEDERATED ONEBOX system provides a single point of entry to the web by brokering over all these search egines like a meta search engine. The FEDERATED ONEBOX system describes all search engines in an offline process, which is indicated in grey in the figure. At query time, the system ranks and selects the most relevant search engines to further process the query, i.e., to forward the query in order to obtain results. This approach introduces some network latency because, once the query is forwarded, the system must wait for the responses of the remote engines before it can merge and show the results to the user. However, the retrieved results are always up-to-date.

In the second architecture, shown in Figure 3b, the FEDERATED ONEBOX system keeps a OneBox configuration for each deep site in a special *deep web entry-point index*. This index does not contain actual deep content. Instead, it contains 'entry-points' that provide acces to deep data, as explained at the beginning of Section 4 (i.e., queries matched against this index result in query interpretations; a query interpretation corresponds to a filled out web form; and, by submitting the web form we can obtain deep web results). As such, query interpretations can be generated as search results without consulting the deep web search engine. As a consequence,

this architecture does not introduce additional network latency for producing deep web search results. In this architecture, there are at least two alternatives for enabling surface web search, these are not detailed further in the figure. The first alternative is to "out source" surface web search, i.e., to forward the query to existing web search engines and merge their results with the deep results. Like in the first acrhitecture, this introduces some additional network latency. The second alternative is to let the system itself crawl and index the surface web; a user query can then be matched against both local indexes (i.e., one for the deep web and one for the surface web). This alternative has two advantages. First, there is no additional network latency which is otherwise introduced by query forwarding. Second, the system has access to the actual scores one which the rankings are based, which can lead to better combined final rankings of the deep and surface results.

4.2 Scientific challenges

We present an adapted list of scientific challenges relating to, on the one hand, providing free-text access to deep content, and on the other hand, relating to the general challenges in federated search.

Query description There is need for a formal syntax in which web administrators can specify the accepted language of the particular resource. How can we keep this intuitive and simple, while allowing enough freedom to specify almost any kind of query, and strict enough to allow easy parsing?

Query translation Due to possible spelling errors, ambiguity, or unknown words to the system, extracting the intended meaning of free-text queries is challenging. A query could be interpreted in different ways. How to devise a feasible approach that achieves reasonable performance over a large percentage of the user queries?

Interpretation ranking As stated in the previous point, a query could be interpreted in many ways. How to rank these interpretations in order to minimize the user's effort to scan through all interpretations, thus quickly finding the right one?

User ignorance How to bridge the gap between the expectations of the user and the capabilities of the system? Is it feasible to automatically suggest available search facets while typing (i.e. the aspects in which the search query can be narrowed further to obtain more specific results)? How to automatically choose suggestions such that the user: 1) is guided while formulating more distinctive queries, and 2) can finish formulating the query faster?

System ignorance How to automatically expand the system's knowledge about valid queries? For example, given a query that contains unknown words, the system presents several annotated interpretations. Then, if the same query is given many times and a particular interpretation is often selected, the system could learn a new rule which includes the unknown word.

Resource description A resource description should facilitate the process of resource selection. A query description (the first challenge) not only describes

how a query should be translated: it effectively describes a resource by its accepted queries. How to adapt and use this description for resource selection?

Resource selection Traditionally, standard IR techniques are applied to rank and select the top(k) resources. Blindly applying these techniques to our resource descriptions will not work, since we describe the accepted queries instead of the resource's contents. How to rank these resources in order to select the top(k), given our resource descriptions? How to determine k?

Results merging A problem of re-ranking a set of results by their relevancy in order to maximize retrieval precision. One resource may return many relevant results, while another may return very few. How to determine the number of results to retrieve from each resource? How to measure their relevancy, in order to rank by relevancy?

5 Ongoing work

We now briefly discuss our preliminary results and other research related specificaly to textual interfaces for querying structured data.

5.1 A free-text interface

We created a novel specification language for configuring OneBox: a free-text interface (FTI) to a multi-field web form. A OneBox configuration specifies: *i*) the dictionary, i.e., the list of accepted tokens for each field; *ii*) an optional set of constraints, e.g., certain input fields may be mandatory and must be filled out by the user; *iii*) a list of patterns that are used to generate suggestions and interpret the query; and, *iv*) the URL of the webform and the formatting rules to display a query interpretation. Using the patterns and the dictionary, OneBox extracts tokens from a query and assigns them to the web form's fields, effectively filling out the web form. OneBox not only interprets the query, it also generates query suggestions on the fly.

5.1.1 Results

We carried out a user study to compare the FTI with an existing travel planner web form. Our results showed that the subjects were significantly faster at finding information when using the FTI instead of the complex form by about 9%. Furthermore, they preferred the FTI over the complex web form. The results also showed that the subjects were highly consistent in their individual query formulations, and that there was considerable query variation between subjects, even in such a relatively simple travel planning scenario.

5.1.2 Related work

(Meng 1999) also researched how to fill out a web form with a given text query. He used statistical disambiguation techniques; however, this requires a substantial amount of (manually annotated) training data from different sources in one domain that is often difficult to obtain. Instead of statistical disambiguation, OneBox

scans for valid pattern combinations and presents a ranked list of alternative interpretations to the user.

Several grammar-based natural language interfaces have been developed (Burton 1976, Hendrix et al. 1978, Carbonell et al. 1983, Carbonell & Hayes 1981); however, the majority of these systems were application-specific which made it difficult to port the systems to different applications (Androutsopoulos et al. 1995). The difficulty of porting a system from one application (domain) to another is also apparent in information extraction systems, i.e. systems that extract all entities from large bodies of texts. The Common Pattern Specification Language (CPSL) was proposed to overcome the difficulty of porting (Appelt & Onyshkevych 1996). At the heart of the CPSL grammar are the *rules*. Each rule has a priority, a pattern and an action. In the spirit of CPSL, we propose a pattern specification language, and use the patterns to scan the input text. However, we generate interactive query suggestions and we produce a ranked list of interpretations instead of a single interpretation.

5.2 When dictionaries are unavailable

The OneBox prototype described in the previous subsection used a dictionary to extract tokens from a query and applied heuristics to rank query interpretations. We extended this prototype by adding token models for identifying possible tokens and applying a probabilistic framework for ranking query interpretations. Therefore, it is able to map out-of-dictionary tokens to fields of a web form, which to our knowledge, is not done by any other system. This approach offers greater flexibility and requires less domain knowledge than existing systems.

5.2.1 Preliminary results

We measured the correctness of the first result (i.e., succ@1) for the following systems: *Upperbound A*: the OneBox prototype described in the previous section that uses a dictionary and ranks by heuristics. *Upperbound B*: the extended OneBox that uses a dictionary and a probabilistic framework for ranking. *System C*: the extended OneBox that has no dictionary (i.e., the dictionary is empty), and uses a probabilistic framework for ranking.

The succ@1-results for the three systems are as follows: 0.9075 for *Upperbound A*, 0.9958 for *Upperbound B*, and 0.7075 for *System C*. These results are promising: even with an empty dictionary, this prototype (system C) can find the right answer in 7 out of 10 times. As the dictionary grows and more tokens are added to the system, the performace will increase towards upperbound-B.

5.2.2 Related work

Similar problems arise in query segmentation (Hagen et al. 2011), and query annotation (Li et al. 2009, Bendersky et al. 2011): a query must be divided into query segments and each segment must be annotated with some label. A label would typically be a name of an input field of a web form, or of the column of a table. This approach is taken in, e.g., (Borkar et al. 2001, Calado et al. 2002, Zhou et al. 2007, Sarkas et al. 2010). Like our work, these system use a probabilistic framework (e.g., a Hidden Markov Model, or Conditional Random Fields). However, these systems do not attempt to identify and label out-of-dictionary tokens.

6 Conclusion and future work

We presented an overview of the state-of-the-art in deep web search and proposed a novel classification scheme to better compare deep web search systems. The current binary classification (surfacing versus virtual integration) hides a number of implicit decisions that must be made by a developer. We make these decisions explicit by distinguishing 7 system aspects that describe a system in terms of its functionality and in terms of its solutions to specific problems.

We motivated the need for Federated OneBox, a search system with one single-field free-text query interface that supports real-time structured search over multiple sources. Federated OneBox contains aspects that are normally found in surfacing approaches (e.g., a single text search interface to all data, searching of many sources and domains), but in essence it is a virtual integration approach (e.g., remote indices, dynamic and real-time results); as such, it combines the best of both worlds. We have layed out a roadmap towards realizing Federated OneBox; we have presented our preliminary findinds; and, we have discussed related work specific to free-text interfaces to search structured data.

Future work will focus more on the challenges relating to federated search (see Section 4.2); up until now, we have focussed on the other challenges relating to free-text interfaces over structured data.

Acknowledgment

This research was supported by the Netherlands Organization for Scientific Research, NWO, grants 639.022.809 and 612.066.513.

References

Álvarez, M., Raposo, J., Pan, A., Cacheda, F., Bellas, F. & Carneiro, V. (2007), Deepbot: a focused crawler for accessing hidden web content, *in* 'DEECS '07: Proceedings of the 3rd international workshop on Data enginering issues in E-commerce and services', ACM, New York, NY, USA, pp. 18–25.

Androutsopoulos, I., Ritchie, G. D. & Thanisch, P. (1995), 'Natural language interfaces to databases – an introduction', *Natural Language Engineering* **1**(01), 29–81.

Appelt, D. E. & Onyshkevych, B. (1996), The common pattern specification language, *in* 'Proceedings of a workshop on held at Baltimore, Maryland', Association for Computational Linguistics, Morristown, NJ, USA, pp. 23–30.

Barbosa, L. & Freire, J. (2007), An adaptive crawler for locating hidden-web entry points, *in* 'Proceedings of the 16th international conference on World Wide Web', WWW '07, ACM, New York, NY, USA, pp. 441–450.

Bendersky, M., Croft, W. B. & Smith, D. A. (2011), Joint annotation of search queries, *in* 'Proceedings of the 49th Annual Meeting of the Association for

Computational Linguistics: Human Language Technologies - Volume 1', HLT '11, Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 102–111.

Bergman, M. K. (2001), 'The deep web: Surfacing hidden value', *Journal of Electronic Publishing*.

Borkar, V., Deshmukh, K. & Sarawagi, S. (2001), Automatic segmentation of text into structured records, *in* 'Proceedings of the 2001 ACM SIGMOD international conference on Management of data', SIGMOD '01, ACM, New York, NY, USA, pp. 175–186.

Burton, R. R. (1976), Semantic grammar: An engineering technique for constructing natural language understanding systems., Technical report, Bolt, Beranek and Newman, Inc., Cambridge, MA.

Cafarella, M. J. (2009), Extracting and querying a comprehensive web database, 4th Biennial Conference on Innovative Data Systems Research (CIDR).

Cafarella, M. J., Halevy, A., Wang, D. Z., Wu, E. & Zhang, Y. (2008*a*), 'Webtables: exploring the power of tables on the web', *Proc. VLDB Endow.* **1**(1), 538–549.

Cafarella, M. J., Madhavan, J. & Halevy, A. (2008*b*), 'Web-scale extraction of structured data', *SIGMOD Rec.* **37**(4), 55–61.

Calado, P., da Silva, A. S., Vieira, R. C., Laender, A. H. F. & Ribeiro-Neto, B. A. (2002), Searching web databases by structuring keyword-based queries, *in* 'Proceedings of the eleventh international conference on Information and knowledge management', CIKM '02, ACM, New York, NY, USA, pp. 26–33.

Callan, J. (2000), Distributed information retrieval, *in* 'Advances in Information Retrieval', Kluwer Academic Publishers, pp. 127–150.

Callan, J. & Connell, M. (2001), 'Query-based sampling of text databases', *ACM Trans. Inf. Syst.* **19**(2), 97–130.

Carbonell, J. G. & Hayes, P. J. (1981), Dynamic strategy selection in flexible parsing, *in* 'Proceedings of the 19th annual meeting on ACL', Association for Computational Linguistics, Morristown, NJ, USA, pp. 143–147.

Carbonell, J. G., Boggs, W. M., Mauldin, M. L. & Anick, P. G. (1983), The XCALIBUR project: a natural language interface to expert systems, *in* 'IJCAI'83', Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 653–656.

Chang, K. C.-C., He, B., Li, C., Patel, M. & Zhang, Z. (2004), 'Structured databases on the web: observations and implications', *SIGMOD Rec.* **33**(3), 61–70.

Dragut, E. C., Fang, F., Yu, C. & Meng, W. (2009*a*), Deriving customized integrated web query interfaces, *in* 'Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology - Volume 01', WI-IAT '09, IEEE Computer Society, Washington, DC, USA, pp. 685–688.

Dragut, E. C., Kabisch, T., Yu, C. & Leser, U. (2009*b*), 'A hierarchical approach to model web query interfaces for web source integration', *Proc. VLDB Endow.* **2**(1), 325–336.

Franklin, M., Halevy, A. & Maier, D. (2005), 'From databases to dataspaces: a new abstraction for information management', *SIGMOD Rec.* **34**(4), 27–33.

Hagen, M., Potthast, M., Stein, B. & Braeutigam, C. (2011), Query segmentation revisited, *in* 'Proceedings of the 20th international conference on World wide web', WWW '11, ACM, New York, NY, USA, pp. 97–106.

Halevy, A., Franklin, M. & Maier, D. (2006), Principles of dataspace systems, *in* 'PODS '06: Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems', ACM, New York, NY, USA, pp. 1–9.

He, B., Tao, T. & Chang, K. C.-C. (2004*a*), Organizing structured web sources by query schemas: a clustering approach, *in* 'CIKM '04: Proceedings of the thirteenth ACM international conference on Information and knowledge management', ACM, New York, NY, USA, pp. 22–31.

He, B., Zhang, Z. & Chang, K. C.-C. (2005), Metaquerier: querying structured web sources on-the-fly, *in* 'SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data', ACM, New York, NY, USA, pp. 927–929.

He, H., Meng, W., Yu, C. & Wu, Z. (2004*b*), 'Automatic integration of web search interfaces with wise-integrator', *The VLDB Journal* **13**, 256–273.

Hendrix, G. G., Sacerdoti, E. D., Sagalowicz, D. & Slocum, J. (1978), 'Developing a natural language interface to complex data', *ACM TODS* **3**(2), 105–147.

Kabisch, T., Dragut, E. C., Yu, C. & Leser, U. (2010), 'Deep web integration with visqi', *Proc. VLDB Endow.* **3**, 1613–1616.

Khare, R., An, Y. & Song, I.-Y. (2010), 'Understanding deep web search interfaces: a survey', *SIGMOD Rec.* **39**, 33–40.

Li, X., Wang, Y.-Y. & Acero, A. (2009), Extracting structured information from user queries with semi-supervised conditional random fields, *in* 'SIGIR '09: Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval', ACM, New York, NY, USA, pp. 572–579.

Madhavan, J., Afanasiev, L., Antova, L. & Halevy, A. (2009), Harnessing the deep web: Present and future, 4th Biennial Conference on Innovative Data Systems Research (CIDR).

Madhavan, J., Ko, D., Kot, L., Ganapathy, V., Rasmussen, A. & Halevy, A. (2008), 'Google's deep web crawl', *Proc. VLDB Endow.* **1**(2), 1241–1252.

Meng, F. (1999), A natural language interface for information retrieval from forms on the world wide web, *in* 'ICIS', Association for Information Systems, Atlanta, GA, USA, pp. 540–545.

Raghavan, S. & Garcia-Molina, H. (2001), Crawling the hidden web, *in* 'Proceedings of the 27th International Conference on Very Large Data Bases', VLDB '01, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 129–138.

Sarkas, N., Paparizos, S. & Tsaparas, P. (2010), Structured annotations of web queries, *in* 'Proceedings of the 2010 international conference on Management of data', SIGMOD '10, ACM, New York, NY, USA, pp. 771–782.

Si, L. & Callan, J. (2005), Modeling search engine effectiveness for federated search, *in* 'SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval', ACM, New York, NY, USA, pp. 83–90.

Wu, P., Wen, J.-R., Liu, H. & Ma, W.-Y. (2006), Query selection techniques for efficient crawling of structured web sources, *in* 'Data Engineering, 2006. ICDE '06. Proceedings of the 22nd International Conference on', p. 47.

Wu, W., Yu, C., Doan, A. & Meng, W. (2004), An interactive clustering-based approach to integrating source query interfaces on the deep web, *in* 'Proceedings of the 2004 ACM SIGMOD international conference on Management of data', SIGMOD '04, ACM, New York, NY, USA, pp. 95–106.

Zhang, Z., He, B. & Chang, K. C.-C. (2004), Understanding web query interfaces: best-effort parsing with hidden syntax, *in* 'SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data', ACM, New York, NY, USA, pp. 107–118.

Zhao, P., Huang, L., Fang, W. & Cui, Z. (2008), Organizing structured deep web by clustering query interfaces link graph, *in* 'ADMA '08: Proceedings of the 4th international conference on Advanced Data Mining and Applications', Springer-Verlag, Berlin, Heidelberg, pp. 683–690.

Zhou, Q., Wang, C., Xiong, M., Wang, H. & Yu, Y. (2007), Spark: adapting keyword query to semantic search, *in* 'ISWC'07/ASWC'07', Springer-Verlag, Berlin, Heidelberg, pp. 694–707.

Notes

[1] `http://www.w3.org/TR/1999/REC-html401-19991224/interact/forms.html\#h-17.13`
[2] `http://www.bing.com/shopping/`,
`http://www.google.com/prdhp?hl=en&tab=pf`,
`http://www.bing.com/recipe/search?q=chocolate`,
`http://www.google.com/search?q=recipe+lasagna`,
`http://www.pricerunner.co.uk/`,
`http://shopping.yahoo.com`.
[3] `http://www.kayak.com/`,
`http://www.travelocity.com/`,
`http://www.cheapoair.com/`