

# Adding Structure to Web Search with iTrails

## [Position Paper]

Marcos Antonio Vaz Salles

Jens Dittrich

Lukas Blunschi

ETH Zurich  
8092 Zurich, Switzerland  
dbis.ethz.ch | iMeMex.org

**Abstract**— We would like to discuss with workshop participants the iTrails framework for pay-as-you-go information integration, which was recently presented at VLDB 2007 [1]. iTrails allows users to provide mini-mappings on their data that sharply increase the quality of search results. The core idea is to extend the semantics of a standard graphical search engine such that the quality of search results approaches the quality of a full-blown information integration system. In contrast to [1], this paper shows how iTrails can be used to tackle the challenges of adding structured information support to web search engines. We will show how iTrails enriches a web search engine with a powerful query rewriting mechanism enabling this engine to perform not only search but also integration of structured information.

### I. INTRODUCTION

Historically, two extreme solutions have been used to query information spread over a set of data sources: *search engines* and *information integration systems*. On one extreme, a search engine allows users to pose simple keyword and path queries over unstructured or semi-structured sources. The semantics of those queries are not precise, meaning that the quality of query answers is evaluated in terms of precision and recall. On the other extreme, an information integration system allows users to pose complex structural queries over semi-structured or structured sources. These systems offer precise query semantics, defined in terms of data models and schemas.

In [1], we have introduced the iTrails framework as a new point in the design space in-between these two extremes. The core idea of our approach is to take a graphical (desktop/web/enterprise) search engine as a baseline and then add hints (aka *trails*) to this baseline such that query results approach the quality of a full-blown information integration system. This gradual semantic enrichment process is termed *pay-as-you-go information integration*. Our approach is an important step towards the vision of a DataSpace Support Platform (DSSP) introduced in 2005 by Franklin et al. [2].

In contrast to [1], this paper shows how the iTrails framework can be used to address the challenges of adding structure to web search engines. Among these challenges are integrating structured information in search results (e.g. Google Base or Google Music Search), tapping deep web data sources, supporting personalization, and exploiting community corpora (e.g. Wikipedia or del.icio.us). In contrast to current search engines, instead of proposing specialized solutions and implementing separate code for each of those challenges, our

approach provides search engine implementors with a single, declarative, and powerful framework — which may be extended by trail definitions to model all of these different needs.

In addition, effectively tackling some of those challenges requires web search engines to have new functionality previously available only in information integration solutions. iTrails has been designed to offer a bridge between the unstructured world of web search engines and the structured world of information integration solutions. For example, iTrails enables a search engine to rewrite a simple keyword query into a structural query that encodes schema information from a given data source. This rewritten structural query may then be recursively rewritten by iTrails into an even more complex structural query that considers schema relationships among different sources. Such added functionality will become increasingly important as the web tends to move towards a web of both structured and unstructured information [3].

This trend imposes on web search engines the burden to process large amounts of structured information and combine this information with the unstructured information they already process. Therefore, we envision that web search engines will evolve in the future into web DSSPs. As a step towards that vision, we discuss architectural options for integrating our pay-as-you-go information integration framework, iTrails, with a state-of-the-art web search engine.

### II. CHALLENGES FOR WEB SEARCH ENGINES

In this section, we describe a set of challenges faced by current web search engines. While these challenges do not exhaust the needs of state-of-the-art search engines, they are fundamental aspects to be considered as the web moves towards a mix of structured and unstructured content [3].

**Integrating Structured Information in Search.** Structure on the web is not restricted to deep web databases, but also includes semi-structured data repositories such as Google Base [4]. This means that search engines must be able to return as search results not only unstructured content but also structured data recorded in these repositories. In order to achieve this, user queries must be reformulated and also sent to these repositories, which may have different schemas and semantics to represent information.

**Tapping Deep Web Data Sources.** Structured content may also be found in deep web databases, which are estimated to

range in the millions [4]. Unlike semi-structured data repositories, deep web databases are not always self-describing. This makes it more difficult to index their data in a meaningful way. Furthermore, it is challenging to access such hidden data for indexing, as appropriate queries must be produced to extract the data from the sources. Even if we choose not to index these sources, we are still left with the problem of reformulating user searches into queries that understand the semantics of the web forms or other web interfaces used as front-ends.

**Supporting Personalization.** Personalization enables systems to customize and rank query results according to the preferences and context of individual users (see e.g. [5]). In web search engines, personalization involves, for example, modifying search queries for disambiguation and for taking into consideration user-specific vocabulary. The modification in search queries depends on a user profile, which must be updated over time to reflect changes in user behavior.

**Exploiting Community-built Corpora.** A number of community-built corpora, such as Wikipedia and del.icio.us, are available on the web. The information in these corpora may be leveraged to improve web search results. For example, information extraction techniques can be used to extract facts about entities, such as cities and persons, from Wikipedia [6]. These facts may be used to refine user searches.

A number of specialized techniques, e.g. [5], [7], have been studied to address the challenges discussed above in isolation. What is missing, however, is a single, powerful framework that can be used to tackle those challenges in a unified and principled manner. We review in the next section our framework for pay-as-you-go information integration, iTrails, and indicate how to use it as a unified solution for the integration of structured content in a state-of-the-art web search engine.

### III. OUR APPROACH: ITRAILS MEETS THE WEB

The iTrails framework consists of three different classes: *semantic*, *association*, and *lineage* trails. The first class was presented in [1]. We are currently exploring the other classes. For space constraints we reduce the following discussion to semantic trails. Note that the name ‘trails’ was inspired by [8].

The basic idea of a semantic trail is to provide a mapping from one query to another, i.e.,  $Trail = Q_L \longrightarrow Q_R$ . The semantics of this are that “whenever we query for  $Q_L$ , we should also query for  $Q_R$ .” Trails are specified by the user or mined semi-automatically from user content. Trail definitions are explored during query processing to enhance the quality of query results. Please see [1] for details.

In the following we provide some use cases illustrating how our framework may be applied to help address the challenges discussed in Section II.

**Integrating Structured Information in Search.** Trails may be used to translate simple keyword queries into more complex structural queries. Furthermore, as structural queries may be posed against distinct schemas, trails may be used to encode schema information from different data sources. Note that it is not required that all sources be fully integrated from the start:

path and keyword queries are available at all times, but are gradually enriched as more trails are created in the system.

**USE CASE 1 (STRUCTURAL REWRITES)** We may use trails to enable users to access semi-structured web repositories. For example, consider the following trails:

$T_1: music:u2 \longrightarrow$

`//base.google.com//item[type = "music"  
and "u2"]`

$T_2://item[type = "music"] \longrightarrow //item[type = "song"]$

$T_3://item[type = "music"] \longrightarrow //item[type = "band"]$

Trail  $T_1$  rewrites a search query containing the tag `music` and keyword `u2` to a path expression selecting all music data items residing in repository `//base.google.com` that mention the keyword `u2`. Trail  $T_2$  matches queries for music item types and rewrites them to queries for song item types. The same is done for trail  $T_3$ , but for band item types.

**Impact:** a simple tag-keyword query like `music:u2` is rewritten to a query considering structural knowledge of a specific web repository. Note that trails are recursive and, therefore, the original query will be rewritten to also consider the complex path expression `//base.google.com//item[(type = "music" or type = "song" or type = "band") and "u2"]`. If other item types or other repositories with different structure should also be considered, then we may add this data by simply specifying more trails. This pay-as-you-go information integration functionality is currently not present in standard web search engines, making the addition of new structured information or new sources a laborious effort.  $\square$

**USE CASE 2 (STRUCTURAL SHORTCUTS)** Search engines often offer specific metadata attributes on which queries may be posed. We may employ trails to rewrite simple keyword queries into queries that exploit the structural information indexed by the engine. For example, consider the trail:

`yesterday  $\longrightarrow$  /**[date=today()-1]`

This trail rewrites a search query containing the keyword `yesterday` to a complex query comparing the `date` attribute of any data item available in the data with the `date` of `yesterday`. The trail definition operator  $\longrightarrow$  means that the results of this trail should *not* be added to the results of the original query `yesterday`, but rather that the original query should be replaced by the query on the right side of the trail definition. We call this type of trail a *replacing trail*.

**Impact:** by typing a simple keyword like `yesterday` a query is replaced by a query considering schema knowledge that better captures the user’s intent.  $\square$

**Tapping Deep Web Data Sources.** Analogously to what was discussed in the previous subsection, we may use trails to rewrite keyword searches into structural queries conforming to the schema of a web source.

**USE CASE 3 (DEEP WEB BOOKMARKS)** Consider the trail below:

`train home  $\longrightarrow$`

`http://www.trains.com/?from=geneve&to=zurich`

This trail rewrites a search query containing the keywords `train home` to a URL retrieving the result of a train search from a web site providing railway timetable information.

**Impact:** a simple keyword query like `train home` is rewritten to a deep web query retrieving data from a web database. □

**Supporting Personalization.** Intuitively, a trail modifies the meaning of a user query according to the equivalence specified in the trail. Thus, trails may be used to model user preferences or user-specific vocabulary. In addition, trails modify user queries before they are actually processed by the web search engine.

This may be achieved by modeling the user profile *as a trail set*. Each user will have a distinct set of trails that rewrites queries according to her preferences and context. The user may then add or remove trails from her trail set to improve the quality of her search experience. In addition, the user's trail set may be automatically customized by the engine over time to reflect user behavior.

USE CASE 4 (USER-SPECIFIC VOCABULARY) Consider that a user has the following trail set:

```
mike → mike jordan
      sf=2.5
mike → mike smith
      sf=0
```

The first trail rewrites a search query containing the keyword `mike` to a search query containing the keywords `mike jordan`. In addition, it specifies a *scoring factor*  $sf = 2.5$  that states that results of the query `mike jordan` should be ranked 2.5 times higher than results from the original query `mike`. The second trail, which has a scoring factor of zero, specifies that results returned by the query `mike smith` should have score zero and, therefore, be suppressed by the search engine.

**Impact:** a simple keyword query like `mike` is rewritten to a more precise query `mike jordan`. This removes ambiguity as the user's search may return information about many different mikes, e.g., 'mike smith', 'mike jones', etc., which should not be highly ranked according to the trails that form the user profile. In fact, the trail set specifies that all information about 'mike smith' should be omitted. The scoring factors as well as the user's trail set can be tuned by adapting techniques such as relevance feedback [9]. This use-case illustrates that user-specific vocabulary is in fact a special case of the iTrails framework. The same holds for *thesauri* like *wordnet* [10], *synonyms* [7], or *language-agnostic search* [11]. □

**Exploiting Community-built Corpora.** A number of community-built sources are currently available on the web. These sources may reflect general knowledge, such as Wikipedia, or may be platforms for sharing information, such as `del.icio.us`. We may profit from trails to enhance information that either is directly available on those sources or can be extracted using information extraction techniques.

USE CASE 5 (MAPS FOR CITIES FROM WIKIPEDIA) A system like Yago [6] can be used to extract information from corpora such as Wikipedia. Such a system may extract the fact that Zurich is a city. This information can be used to

automatically create a trail that will return map information whenever a query for Zurich is posed:

```
zurich → http://www.maps.org/city=zurich
```

This trail rewrites the keyword search `zurich` to a query for a map of Zurich directed at a map server.

**Impact:** a simple keyword query like `zurich` is rewritten to also consider structured data available on the web. Note that although support for this rewrite could in theory be implemented as specific code in the search engine, the advantage of our approach is that a single declarative framework is able to rewrite queries in order to consider a number of different use cases and structured sources. □

USE CASE 6 (WEB BOOKMARKS) Standard web bookmarks are again just a special case in our iTrails framework. The following trail models a bookmark:

```
news → http://www.bbc.com
```

This trail rewrites a search query containing the keyword `news` to a URL retrieving the content of the web page `http://www.bbc.com`.

**Impact:** by typing a simple keyword like `news` a query is rewritten to also consider data available on the web. □

#### A. Where do Trails Come From?

We argue that an initial set of trails should be shipped with the initial configuration of a system. Then, a user (or a company) may extend the trail set and tailor it to her specific needs in a pay-as-you-go fashion. For instance, whenever the user detects that a query result may be enhanced by adding another trail she may do so.

We see four principal ways of obtaining a set of trail definitions:

- (1) Define trails using a drag&drop frontend,
- (2) Create trails based on user-feedback in the spirit of relevance feedback in search engines [9],
- (3) Mine trails (semi-) automatically from content,
- (4) Obtain trail definitions from collections offered by third parties or on shared web platforms.

As an example of (4.), sites in the style of bookmark-sharing sites like `del.icio.us` could be used to share trails. As shown above, a bookmark is just as a special case of a trail, in which a set of keywords induce a specific resource on the web. Sites for trail sharing could include specialized categories of trails, such as trails related to personal information sources (e.g., email, blogs, etc), trails on web sources (e.g., dictionaries translating location names to maps of these locations), or even trails about a given scientific domain (e.g., gene data). All of these aspects are interesting avenues for future work.

#### IV. ARCHITECTURAL OPTIONS TOWARDS WEB DSSPs

We argue that web search engines will evolve towards Web DataSpace Support Platforms (Web DSSPs) in order to better support structured information. In this section, we discuss how our iTrails pay-as-you-go information integration framework

could be integrated with a state-of-the-art web search engine. The resulting architectures are a step towards web DSSPs.

**Client-side Integration.** The client-side integration architecture is displayed in Figure 1(a). In this option, clients are equipped with not only a web browser but also additional software comprising the iTrails framework and a query dispatcher. The iTrails framework rewrites user queries, which are then dispatched by the query dispatcher. Clients access a traditional web search engine to obtain unstructured information and directly send reformulated queries to structured sources. The advantages of this approach are that it may be readily implemented without changes to the underlying search engine and that the user’s trail set does not need to be shared with the search engine. The latter aspect entails an added level of user privacy. In addition, local client sources may also be indexed and queried by the user. One disadvantage of this architecture is the high computational cost of mediation, especially with respect to the structured sources. This cost may be controllable if only a relatively small subset of the structured sources are of interest to the user. We are currently implementing this architecture in order to integrate our current iTrails prototype with a state-of-the-art web search engine.

**Meta-search Integration.** This architectural option is shown in Figure 1(b) and consists of interposing a meta-search layer in-between clients and web search engine or structured sources. As in client-side integration, iTrails rewrites queries before handing them to a query dispatcher. In this architecture, there is again no need to make any changes to the underlying web search engine. Unlike client-side integration, however, this architecture requires users to trust their trail sets to the meta-searcher. In addition, the computational cost of mediation may be significant, as the meta-searcher must serve queries from several users, potentially interested in a large number of structured web sources.

**Search-engine Integration.** Search-engine integration is depicted in Figure 1(c). The web search engine is extended to provide indexing both for structured and unstructured data sources. The different indexes are unified under a common query interface. User queries are reformulated by iTrails before being handed to the query interface over the indexes. One clear advantage of this approach is that the cost of query-time mediation is eliminated. On the other hand, it is necessary to index all structured sources. This process is by no means trivial, given the heterogeneity of deep web databases. Furthermore, implementing this approach requires us to have access to the code and to modify an existing web search engine. This approach is similar to the one defended by [4].

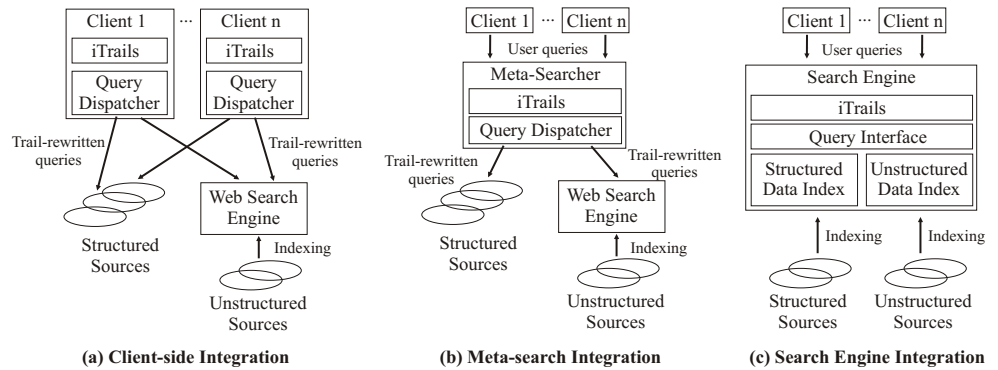


Fig. 1. Architectural options for integrating iTrails with a web search engine

## V. CONCLUSION

iTrails [1] provides a new way to enrich the semantics of a search engine by adding hints (aka trails). The iTrails framework enables users to perform pay-as-you-go information integration as envisioned in [2]. A prototype of iTrails is implemented in the iMeMex Dataspace Management System [12], [13] and available for download at [iMeMex.org](http://iMeMex.org).

In contrast to [1], which proposed the iTrails framework, this paper analyzes how iTrails can be used to support structured information over a standard web search engine. We have identified a set of challenges faced by current web search engines. Based on those challenges, we have illustrated through a set of use cases how iTrails may be used to extend a web search engine to support pay-as-you-go information integration. In addition, we have described architectural options to integrate iTrails with a standard web search engine, resulting in a step towards the vision of web dataspace support platforms.

## REFERENCES

- [1] M. A. V. Salles, J.-P. Dittrich, S. K. Karakashian, O. R. Girard, and L. Blunschi, “iTrails: Pay-as-you-go Information Integration in Dataspaces,” in *VLDB*, 2007, slides at <http://www.vldb2007.org/program/slides/s663-vazsalles.pdf>, video at <http://www.youtube.com/watch?v=F24-UHYFjHY>.
- [2] M. Franklin, A. Halevy, and D. Maier, “From Databases to Dataspaces: A New Abstraction for Information Management,” *SIGMOD Record*, vol. 34, no. 4, pp. 27–33, 2005.
- [3] A. Halevy, O. Etzioni, A. Doan, Z. Ives, J. Madhavan, L. McDowell, and I. Tatarinov, “Crossing the Structure Chasm,” in *CIDR*, 2003.
- [4] J. Madhavan et. al., “Web-scale Data Integration: You can only afford to Pay As You Go,” in *CIDR*, 2007.
- [5] G. Koutrika and Y. Ioannidis, “Constrained Optimalities in Query Personalization,” in *ACM SIGMOD*, 2005.
- [6] F. M. Suchanek, G. Kasneci, and G. Weikum, “YAGO: A Core of Semantic Knowledge Unifying WordNet and Wikipedia,” in *WWW*, 2007.
- [7] Y. Qiu and H.-P. Frei, “Concept Based Query Expansion,” in *ACM SIGIR*, 1993.
- [8] V. Bush, “As we may think,” *Atlantic Monthly*, 1945.
- [9] R. Schenkel and M. Theobald, “Feedback-Driven Structural Query Expansion for Ranked Retrieval of XML Data.” in *EDBT*, 2006.
- [10] WordNet. <http://wordnet.princeton.edu/>.
- [11] L. Ballesteros and W. B. Croft, “Phrasal Translation and Query Expansion Techniques for Cross-language Information Retrieval,” in *ACM SIGIR*, 1997.
- [12] J.-P. Dittrich, M. A. V. Salles, D. Kossmann, and L. Blunschi, “iMeMex: Escapes from the Personal Information Jungle (Demo),” in *VLDB*, 2005.
- [13] L. Blunschi, J.-P. Dittrich, O. R. Girard, S. K. Karakashian, and M. A. V. Salles, “A Dataspace Odyssey: The iMeMex Personal Dataspace Management System (Demo).” in *CIDR*, 2007.