Entwining Structure into Web Search

Stelios Paparizos
Microsoft Research
1065 La Avenida
Mountain View, CA 94043, USA
steliosp@microsoft.com

ABSTRACT

Keyword based retrieval has dominated web search with its simplicity and effectiveness. However, given the abundance of structured and semi-structured data, there is an opportunity for an improved user experience. In this paper, we present an overview of our past work on analyzing keyword queries and extracting latent structured semantics, mapping them to corresponding data sources and enriching the user experience with the inclusion of structured content.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Data Mining

General Terms

Performance

Keywords

Structured Web Queries, Semantic Analysis, Semantic Web

1. INTRODUCTION

Web search has evolved from a carefully selected hierarchy of web page bookmarks, to a huge collection of crawled documents requiring fast retrieval algorithms to identify the few most relevant results, and more recently, to highly sophisticated answering ecosystems utilizing information from multiple diverse sources. Yet today's prominent solutions still leave a lot to be desired for certain classes of queries driven by real user needs.

Consider for example a user trying to find a digital camera for under $300, with great low light capabilities that fits their jeans' back pocket. If the user is lucky to have an expert friend they can find their answer quickly. The alternative solution is to spend time going through the reviews in a specialized website such as `dpreview.com`. For obvious reasons, textual relevance using IR-like techniques is not the best approach in such a scenario.

DBRank 2013 August 30 2013, Riva del Garda, Italy

ACM 978-1-4503-2497-7/13/08 ...$15.00
http://dx.doi.org/10.1145/2524828.2524830.

We should note that this is not a rare hand-picked example applicable only to one domain. On the contrary, similar trends appear in other real user scenarios for a variety of domains. For example, user queries from other domains include '*movies with Pacino and De Niro*', '*sauce recipes with tomato, basil but not cilantro*' or '*used compact cars less than 5 years old near by mountain view*'. With the abundance of available structured data, a search engine has the information to provide the answer to such queries. Yet, enabling such functionality poses a few systemic challenges:

Web Speed & Web Scale: Web users are accustomed to fast responses; even tiny delays often result in query abandonment and loss of revenue for search engines. Making things worse, there are hundreds of millions of queries per day targeting numerous and sizeable data stores. The combination makes an efficient end-to-end solution non trivial.

Free-text queries: Web users seek information in the open world and issue queries oblivious to the existence of structured data sources – let alone their schema and their arrangement. To produce meaningful results, latent structured semantics should be extracted from the query keywords. A mechanism that directly maps keywords to structure can lead to misinterpretations of the user's intent for a large class of queries. There are two possible types of misinterpretations: between web versus structured data, and between individual structured tables. For example, consider the query '*white tiger*' and assume there is a table available containing Shoes and one containing Books. A potential mapping can be *Table* = "Shoes" and attributes *Color* = "white" and *Shoe Line* = "tiger", after the popular 'Asics Tiger' line. A different potential mapping can be *Table* = "Books" and *Title* = "white tiger", after the popular book. Although both mappings are possible, it seems that the book is more applicable in this scenario. On the flip side, it is also quite possible the user was asking information that is not contained in our collection of available structured data, for example about "white tiger", the animal. Hence, although multiple structured mappings can be feasible, it is important to determine which one is more plausible among them and which ones are at all meaningful.

Interactive User Interfaces: The efficiency of showing a set of urls and their summarized info has dominated web search. However, it is does not translate well to visualizing results from structured data stores. Instead, faceted search and rich structured information are preferred in specialized domains. The selection of attributes and values to be shown are dictated by expert designers of the specialized domains. With the vast structured data sources available in generic web search a manual solution is not feasible. So the system needs to automatically learn what is the best visual-

ization for the user query and corresponding results.

2. OUR APPROACH

Our solution assumes structured data exists and we show how it can be used to create mappings from keywords to tables and attributes (details in [1, 2, 4, 6]). Once the mappings are established we discuss query translation and execution and a new type of optimization opportunities (details in [3]). Finally, we present a way to dynamically generate facets and data item summaries by mining the popular attributes and values for each data typedetails in [5]).

2.1 Structured Data Model

We start our discussion by giving the intuition for some basic concepts. We assume that structured data is organized as a collection of *tables*. A table T is a set of related *entities* sharing a set of *attributes* $T.\mathcal{A}$. Attributes can be either *categorical* or *numerical*. The *domain* of a categorical attribute is the set of possible values it can take. We assume that each numerical attribute is associated with a single *unit* U of measurement. An example of two tables is shown in Figure 1. The first table contains TVs and the second Monitors. They both have three attributes: Type, Brand and Diagonal. Type and Brand are categorical, whereas Diagonal is numerical. The domain of values for all categorical attributes for both tables is $\mathcal{T}.\mathcal{A}_c.\mathcal{V}$ = {TV, Samsung, Sony, LG, Monitor, Dell, HP}. The domain for the numerical attributes for both tables is $\mathrm{Num}(\mathcal{T}.\mathcal{A}_n.\mathcal{U}) = \mathrm{Num}(\{\mathrm{inch}\})$. Note that $\mathrm{Num}(\{\mathrm{inch}\})$ does not include only the values that appear in the tables of the example, but rather all possible numbers followed by the unit "inch".

A *token* is as a sequence of characters including space, i.e., one or more words. We define the *Open Language Model* (OLM) as the infinite set of all possible tokens. All keyword web queries can be expressed using tokens from OLM. A *typed token* t for table T is any value from the *domain* of $\{T.\mathcal{A}_c.\mathcal{V} \cup \mathrm{Num}(T.\mathcal{A}_n.\mathcal{U})\}$.For the rest of the paper, for simplicity, we often refer to *typed tokens* as just *tokens*. The *Closed Language Model* CLM of table T is the set of all duplicate-free typed tokens for table T. Since for numerical attributes we only store the "units" associated with $\mathrm{Num}(\mathcal{U})$, the representation of $\mathrm{CLM}(T)$ is very compact. The closed language model $\mathrm{CLM}(\mathcal{T})$ for all our structured data $\mathcal{T}$ is defined as the union of the closed language models of all tables.

The closed language model defines the set of tokens that are associated with a collection of tables, but it does not assign any *semantics* to these tokens. An *annotated token* for a table T is a pair $AT = (t, T.A)$ of a token $t \in \mathrm{CLM}(T)$ and an attribute A in table T, such that $t \in T.A.\mathcal{V}$. In the example of Figure 1, the annotated token (LG, TVs.Brand) denotes that the token "LG" is a possible value for the attribute TVs.Brand. The *Closed Structured Model* of table T, $\mathrm{CSM}(T) \subseteq \mathrm{CLM}(T) \times T.\mathcal{A}$, is the set of all annotated tokens for table T. Note that in the example of Figure 1, the annotated token (LG, TVs.Brand) for $\mathrm{CSM}(\mathrm{TVs})$ is different from the annotated token (LG, Monitors.Brand) for $\mathrm{CSM}(\mathrm{Monitors})$, despite the fact that in both cases the name of the attribute is the same, and the token "LG" appears in the closed language model of both TVs and Monitors table. Furthermore, the annotated tokens (50 inch, TVs.Diagonal) and (15 inch, TVs.Diagonal) are part of for $\mathrm{CSM}(\mathrm{TVs})$, despite the fact that table TVs does not contain entries with those values.

The CSM for the collection $\mathcal{T}$ is defined as the union of the structured models for the tables in $\mathcal{T}$. In practise, this is a fast lookup process as annotated tokens can be kept in a hash table. To keep a small memory footprint, $\mathrm{CSM}(\mathcal{T})$ can be implemented using token pointers to $\mathrm{CLM}(\mathcal{T})$, so the actual values are not replicated.

TVs

Type	Brand	Diagonal
TV	Samsung	46 inch
TV	Sony	60 inch
TV	LG	26 inch

Monitors

Type	Brand	Diagonal
Monitor	Samsung	24 inch
Monitor	Dell	12 inch
Monitor	HP	32 inch

Figure 1: A two-table example

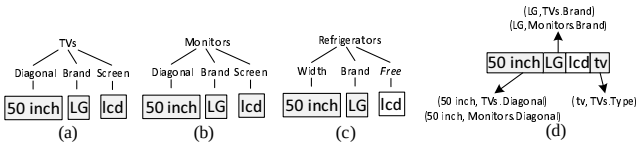


Figure 2: Examples of annotations and annotation generation.

2.2 Query Annotations

A *Structured Annotation* S_q of query q over a table collection $\mathcal{T}$, is a triple $\langle T, \mathcal{AT}, \mathcal{FT}\rangle$, where T denotes a table in $\mathcal{T}$, $\mathcal{AT} \subseteq \mathrm{CSM}(T)$ is a set of annotated tokens, and $\mathcal{FT} \subseteq \mathrm{OLM}$ is a set of words such that $\{\mathcal{AT}.t, \mathcal{FT}\}$ is a segmentation of q. Intuitively, a segmentation of a query is a sequence of non-overlapping tokens that cover the entire query. A *structured annotation* is a mapping of the user-issued keyword query to a structured data table T, a subset of its attributes $\mathcal{AT}.A$, and a set of *free tokens* $\mathcal{FT}$ of words from the open language model.

Now, consider the keyword query q ="50 inch LG lcd". Assume that we have a collection $\mathcal{T}$ of three tables over TVs, Monitors, and Refrigerators, and there are three possible possible annotations $\langle T, \mathcal{AT}, \mathcal{FT}\rangle$ of q (shown in Figure 2(a-c)). Although many structured annotations are possible, only a handful, if any, are *plausible* interpretations of the keyword query. For instance, annotation S_1 (Figure 2(a)) is a perfectly sensible interpretation of q. This is not true for annotations S_2 and S_3. S_2 maps the *entire* keyword query to table Monitors, but it is highly unlikely that a user would request Monitors with such characteristics, i.e., $(50\ \mathrm{inch}, \mathrm{Monitors.Diagonal})$, as users are aware that no such large monitors exist (yet?). Annotation S_3 maps the query to table Refrigerators. A request for Refrigerators made by LG and a Width of 50 inches is sensible, but it is extremely unlikely that a keyword query expressing this request would include free token "lcd", which is irrelevant to Refrigerators. Note that the existence of free tokens does not necessarily make an annotation implausible. For example, for the query "50 inch lcd *screen* LG", the free token "screen" increases the plausibility of the annotation that maps the query to the table TVs. Such subtleties demand a robust scoring mechanism, capable of eliminating implausible annotations and distinguishing between the (potentially many) plausible ones.

Given the set of all annotations $\mathcal{S}_q$ for a query q, our scorer outputs for each annotation $S_i \in \mathcal{S}_q$ the probability $P(S_i)$ of a user requesting the information captured by the annotation. We model this using a *generative probabilistic model* assuming users "generate" an annotation S_i (and the resulting keyword query) as a two step process. First, with probability $P(T.\mathcal{A}_i)$, they decide on the table T and the subset of its attributes $T.\mathcal{A}_i$ that they want to query (to include free tokens in the query, we extend the set of attributes of each table T with an additional attribute $T.f$ that emits free tokens). In the second step, given their previous choice of attributes $T.\bar{\mathcal{A}}_i$, users select specific annotated and free tokens with proba-

bility $P(\{\mathcal{AT}_i, \mathcal{FT}_i\}|T.\tilde{\mathcal{A}}_i)$. After some simplification independence assumptions we can write this as:

$$P(S_i) = P(\mathcal{AT}_i|T.\mathcal{A}_i)P(\mathcal{FT}_i|T)P(T.\tilde{\mathcal{A}}_i) \quad (1)$$

To capture the possibility the query was not targeting structured data our model incorporates the open-language "table" OLM which has only the free-token attribute OLM.f and generates all possible free-text queries. We populate the table using a generic web query log. We generate an additional annotation $S_{\text{OLM}} = \langle \text{OLM}, \{\mathcal{FT}_q\}\rangle$, and we evaluate it together with all the other annotations in $\mathcal{S}_q$. Thus the set of annotations becomes $\mathcal{S}_q = \{S_1, ..., S_k, S_{k+1}\}$, where $S_{k+1} = S_{\text{OLM}}$, and we have:

$$P(S_{\text{OLM}}) = P(\mathcal{FT}_q|\text{OLM})P(\text{OLM}) \quad (2)$$

The S_{OLM} annotation serves as a "control" against which all candidate structure annotations need to measured. More specifically, for some $\theta > 0$, we say that a structured annotation S_i is *plausible* if $\frac{P(S_i)}{P(S_{\text{OLM}})} > \theta$. In other words, an annotation, which corresponds to an interpretation of the query as a request which can be satisfied using structured data, is considered plausible if it is *more probable* than the open-language annotation, which captures the absence of demand for structured data.

The detailed mathematical formulations as well a description of how the annotation mechanism was implemented efficiently is covered in [6].

2.3 Semantic Synonyms

It is possible to extend the *Closed Structured Model* by enhancing the attribute domains with synonyms, e.g., by using "in" for "inches" and "Hewlett Packard" for "HP". Discovery of synonyms can be done using web query and click logs. Details on the formulas and algorithms used can be found in [1, 2]. We give here an intuitive summary of our approach.

We begin by seeking the Web pages that are good representation for entities, we call them *surrogates*. For each entity value we issue a query to the Bing Search API and keep the top-k results as surrogates. It may also be possible to use *Click Data*, however, clicks are not always available for this purpose, as the entities' data values usually come in the canonical form (e.g., the full title name of a movie), and therefore may not be used as queries by people. Having identified each entity's surrogates, we next find out how users access those surrogates. Clicks are useful in this direction for discovering when users access the same page from different queries. We collect all queries that resulted in clicks on the surrogate pages and consider them as synonym candidates. Figure 3 shows a visualization of this bipartite graph representation.

To estimate the likelihood that a candidate query string is a synonym of the input value, we identify two measures that capture the *strength* and *exclusiveness* between them. The *Intersecting Page Count* (IPC) captures the number of pages that are overlapping between two nodes on the opposite sides of the bipartite graph and is an indication of the strength of the relationship. The *Intersecting Click Ratio* (ICR) measures the volumes of clicks captured by the edges in the bipartite graph and is an indication of the exclusive nature of the relationship.

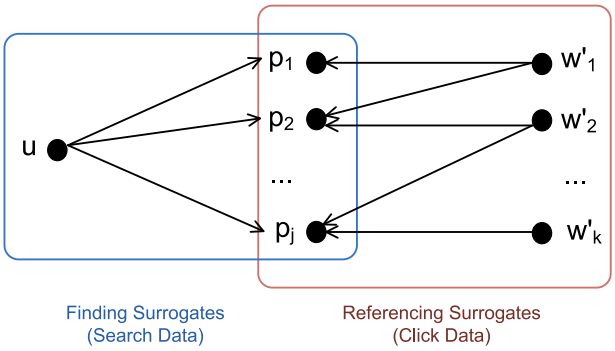
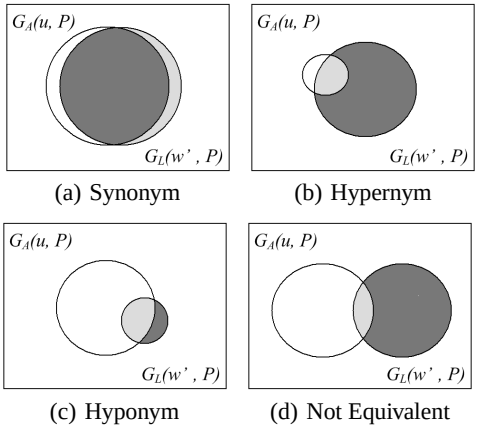


Figure 3: Candidate Generation Phase



(a) Synonym (b) Hypernym

(c) Hyponym (d) Not Equivalent

Figure 4: Venn Diagram Illustration for Synonyms

We use a Venn diagram illustration in Figure 4 to describe how the above two measures work in selecting the best synonyms. Consider the example where the input is the movie title "Indiana Jones and Kingdom of the Crystal Skull". Figure 4(a) illustrates the case where a candidate (e.g. "Indiana Jones 4") is a likely synonym by having high IPC and ICR values. Figure 4(b) illustrates the case of a hypernym (e.g. "Indiana Jones"). Since a hypernym considers a broader concept, it may be used to refer to many more pages (e.g. other Indiana Jones movies), and consequently most of the clicks fall outside of the intersection (low ICR). A hyponym concerns a narrower concept, where there might be more specific pages about the concept outside of the intersection that receive the most clicks (Figure 4(c)). Finally, a candidate such as "Harrison Ford" is only related, with low IPC and ICR (Figure 4(d)). We produce the final synonyms by applying threshold values β and γ on IPC and ICR respectively.

2.4 Optimizing for Result Quality

A major challenge in using structured data to answer keyword queries is that users often lack domain expertise and may pose queries that lead to very few or no results. As an example, consider the query '*50 inch samsung led tv*', that can be thought as (`50 inch` ⇒ `display size`, `Samsung` ⇒ `Brand`, `led tv` ⇒ `display type`). If the query is directly evaluated as specified, there will be no results that can satisfy the interpretation as Samsung does not make 50-inch LED TVs. On the other hand, Samsung makes *46-inch* and *55-inch LED* TVs and 50-inch *PLASMA* TVs. Arguably, the users would prefer to see such results that are close to their original query instead of looking at an empty page with no results because they did not know the appropriate precise values when typing the query. To achieve this we rewrite the queries through semantic

term expansion. For example, the above query may be rewritten as '*(46 to 55 inch) samsung (led or plasma) tv*'. The rewritten query should preserve the meaning of the original query as closely as possible, and the rewritten query should ensure there are sufficiently many results returned to the user.

If time had not been an issue, a simple solution would be to iteratively make small relaxations to the query, issue it to the database to find out the number of matches, and repeat until we have found k results. However, due to the performance requirements imposed by web search, this approach is infeasible as numerous database accesses are costly. Instead, we use database estimates and relax each attribute at a time with near by values. As the optimal problem is NP-hard we considered a greedy and a dynamic programming algorithm. The benefit of the greedy approach is that it will find a relaxation that is better than the original query no matter what is the size of the given time envelope. Whereas the dynamic programming solution works best above a minimum time threshold after which it always does a better job than greedy. Algorithmic details and the experiments to back this claim can be found in [3].

2.5 Automatic Facet Generation

We consider a *facet system* for table T an ordered list of k *facets*. A facet consists of an attribute A_i and an ordered list of m_i values. A facet system is exposed through the user interface, and it enables the user to navigate through the data in the table T by selecting values for the different attributes. In any faceted search system, the facet selection is paramount to the success of the system. The order of facets is also important since it determines how the facets will be displayed, with more important facets being displayed more prominently. The goal is to maximize the user engagement and satisfaction, and the i-th facet corresponds to the i-th best attribute for this task. A similar reasoning applies also to the ordering of the facet values within a facet. However, for a selected facet attribute there is usually a greater flexibility in the presentation of values (drop-down menus, slider bars, text boxes) making the value selection and ordering less critical. We thus consider the *facet attribute ordering* problem and the *facet attribute value ordering* problem separately, with the former being more important than the latter.

We can construct such facet system automatically using web search queries. We assume that we have a query log, consisting of queries posed to a web search engine. For each query q_i we also have a weight w_i denoting the importance of the query, e.g., the number of times that the query appears in the query log. After producing the *Structured Annotations* for each query, we can extract the metadata and compute an *Attribute Utility* with the summation of the weights for each query the attribute is present (similarly for value utility). The solution would be simple counting if there was no attribute mapping ambiguity. To understand the ambiguity, consider diagonal, width, height are all similar dimensions for televisions, and computer monitors. And diagonal and width even have relatively similar value distributions. As a result a naive approach could confuse the significance of monitors over televisions and width over the more common diagonal dimension – or even worse, split the weight between the two and consider none significant enough.

The detailed problem formulations and our mathematical models are discussed in [5]. Intuitively, we solve the ambiguity issues, by modeling the process as a double optimization problem over the query log. The first pass resolves the ambiguity across tables (i.e. between monitors and televisions) and the second focuses only on the queries associated with a single table and resolves the ambiguity across attributes (i.e. between diagonal and width for televisions). After the ambiguity is resolved, each query is assigned with some weight probability to a table, attributes and values. Then the values are added for each attribute and the attributes for each table and the ordering for the facet system for the table is automatically generated.

To verify our findings, performed an extensive user study and found that our automatically mined facets outperformed the ones created by experts for a popular shopping website like Amazon (more details in [5]).

3. CONCLUDING THOUGHTS

Applications of web search over structured data are becoming more common and more important. Their challenges go beyond traditional keyword search over databases and incorporate aspects of web scale & speed and semantic understanding of free-form text queries along with rethinking the user interface paradigm. In this paper we presented an overview of work we did in this area and the solutions we considered. Of course, there is a variety of other interesting approaches in related work that we did not cover as they were out of the scope of this paper. Yet, we believe this is an undeveloped area with tons of opportunities for further research.

As final words, we would like to draw attention to the lack of an appropriate benchmark that can be used to make the comparison amongst solutions feasible and help measure systemic innovation. Database research in the past has benefited a lot by the benchmarks that measured execution performance. However, the results of database queries follow precise semantics and only the speed by which they are retrieved can be different. In structured web queries, a benchmark must consider relevance of results and user satisfaction in combination with speed of result retrieval.

4. ACKNOWLEDGMENTS

The author would like to thank the following collaborators for their invaluable contributions at various aspects of the presented work: Rakesh Agrawal, Tao Cheng, Sreenivas Gollapudi, Sam Ieong, Hady Lauw, Alex Ntoulas, Jeff Pound, Nikos Sarkas, John Shafer and Panayiotis Tsaparas.

5. REFERENCES

[1] T. Cheng, H. W. Lauw, and S. Paparizos. Fuzzy matching of web queries to structured data. In *Proc. ICDE Conf.*, 2010.

[2] T. Cheng, H. W. Lauw, and S. Paparizos. Entity synonyms for structured web search. *IEEE Trans. Knowl. Data Eng.*, 24(10):1862–1875, 2012.

[3] S. Gollapudi, S. Ieong, A. Ntoulas, and S. Paparizos. Efficient query rewrite for structured web queries. In *Proc. CIKM Conf.*, 2011.

[4] S. Paparizos, A. Ntoulas, J. C. Shafer, and R. Agrawal. Answering web queries using structured data sources. In *Proc. SIGMOD Conf.*, 2009.

[5] J. Pound, S. Paparizos, and P. Tsaparas. Facet discovery for structured web search: a query-log mining approach. In *Proc. SIGMOD Conf.*, 2011.

[6] N. Sarkas, S. Paparizos, and P. Tsaparas. Structured annotations of web queries. In *Proc. SIGMOD Conf.*, 2010.