Mesa: A Search Engine for Querying Web Tables

Sergio Mergen[1], **Juliana Freire**[2], **Carlos Heuser**[1]

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil

[2]School of Computing – University of Utah
Salt Lake City, U.S.

{mergen,heuser}@inf.ufrgs.br, juliana@cs.utah.edu

Abstract. *The volume of structured data on the Web has grown considerably in the recent past. In contrast to unstructured (textual) documents, which can be searched through simple keyword-based interfaces, the presence of structure enables rich queries to be posed against Web data. In this paper we present a search engine designed for querying structured information sources on the Web and show how our system can support on-the-fly, complex queries over content published in hundreds HTML tables.*

1. Introduction

From data produced by Web services to HTML tables, the volume of structured data on the Web has grown considerably in the recent past. Whereas keyword-based interfaces have been widely used for unstructured content, more expressive query interfaces are possible when the content is structured. This creates new opportunities for exploring and integrating information from an unprecedented number of Web sources.

Existing approaches to querying multiple information sources [Halevy 2000, McBrien and Poulovassilis 2003], however, are not suitable for this scenario. Because these approaches require that mappings be provided between information sources and a mediated schema, they are not scalable: it is not feasible to manually create and maintain mappings for hundreds of sources. In addition, because new sources are constantly added and existing sources modified (both content and structure), keeping track of sources as they change and updating mapping information and global schema can be prohibitively expensive.

In this paper we describe Mesa, a new search engine designed to support queries over structured Web content. We focus on content published as HTML tables, whose schema and contents can be extracted automatically. Because Mesa does not require a pre-defined mediated schema and automatically maps user queries onto queries over the information sources, it is able to effectively handle the scale and dynamic nature of structured Web sources.

The remainder of this paper is organized as follows. In Section 2, we present an overview of the Mesa system. In Section 3, the system architecture and implementation. We review related work in Section 4 and in Section 5, we discuss directions for future work.

2. Querying Tables with `Mesa`

Similarly to a traditional search engine, such as Google and Yahoo!, `Mesa` allows users to query information from HTML tables that have been previously indexed (see Section 3). One of the unique features of the query interface is the ability to handle structured queries. Currently, `Mesa` supports conjunctive queries formulated in SQL. `Mesa` uses a best-effort approach to rewrite the user query into queries over the indexed HTML tables. As shown in Figure 1a), the structured query `select title, gross from movie` returns a list of rewritings that contain possible answers to the query. Each rewriting retrieves answers from one or more source tables and is displayed together with a summary of the source tables used in the rewriting as well as links to the URL of the source pages where the tables are located.

When the user clicks on a rewriting, `Mesa` accesses the actual source page and extracts the desired information (specified in the query) from the source tables located in that page. The output of the selected rewriting is displayed to the user in a tabular format. The content of the output table is a subset of the content that can be found in the source table, including links and images (relative links are automatically converted to their respective absolute links). By keeping the hypertext-based information we give the user the power to navigate through the results, exploring its contents and discovering additional information that are related to her query.

In what follows, we describe some additional features of the prototype.

Querying Modes. Besides SQL queries, `Mesa` also supports keyword-based queries (see text box labeled *Keywords* in Figure 1). Keyword-based queries can be used alone or in conjunction with structured queries. If the user issues a keyword-based query, `Mesa` derives rewritings that access pages that contain the keywords specified. Each rewriting refers to a particular table and covers the contents in all columns of the table. For instance, Figure 1e) shows the first rewriting produced when the query `groucho` is issued. This rewriting retrieves a source table that in a page that contains information about the `Marx Brothers`.

When a structured query and keyword-based query are used in the same request, rewritings are derived that: i) cover the columns specified in the structured part of the query; ii) come from a page from where the keywords can be found. For example, suppose the user enters `select year from movie` as the structured query and `Groucho` as the keyword query. Also suppose that all indexed source tables cover the column `year`, but only one page contains the requested keyword. In this case, only the tables that come from this specific page will be used in the rewritings and the rewriting will return only the `year` column.

Query Expansion and Refinement. The query expansion feature allows the user to discover additional information that is available in the retrieved tables. For instance, consider the query output presented in Figure 1a). If the user clicks on the link *"Click here to expand query"* below the rewriting, as depicted in Figure 1c), a table is displayed that contains all columns from the source table.

Next to each column header, a drop-down list is provided that enables users to refine a query by switching a returned column to any other column that is available in the

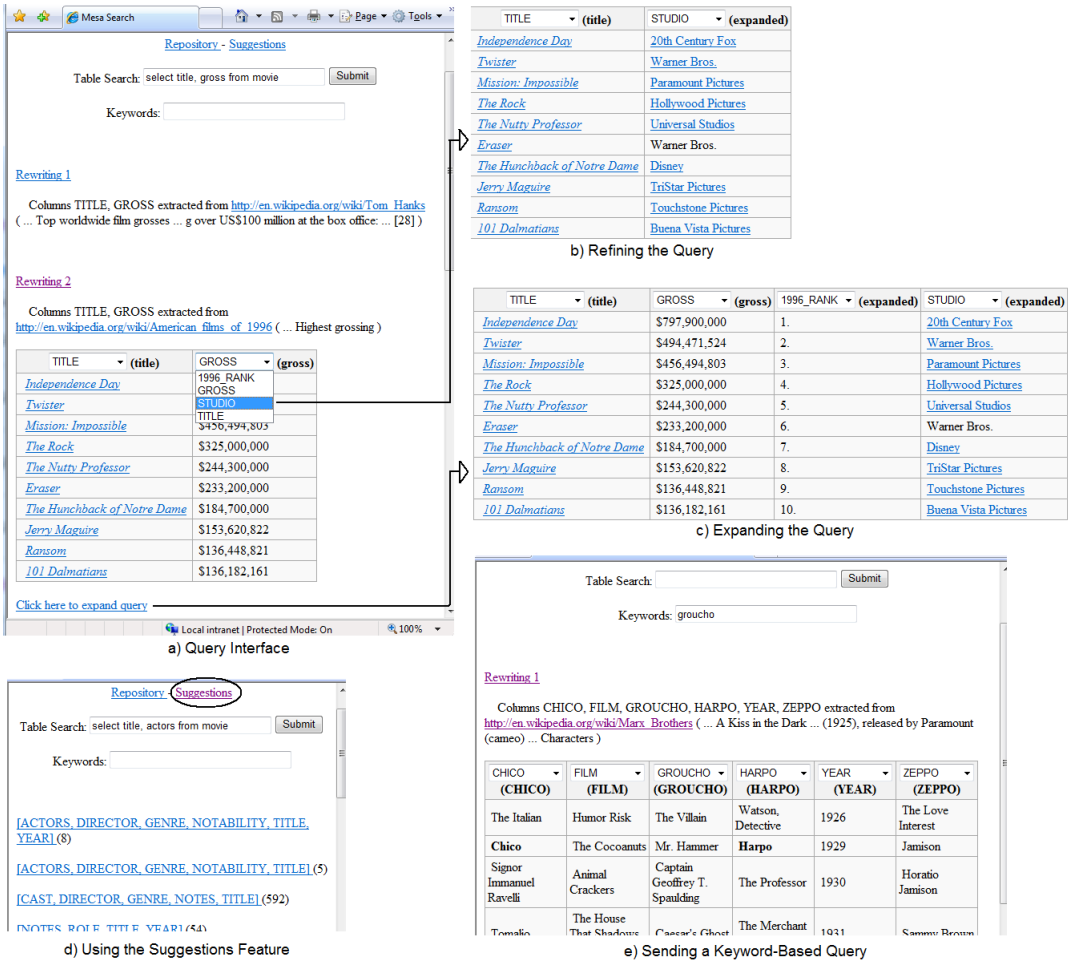

Figure 1. Query interface of *Mesa*

source table. The table in Figure 1b) is returned when a user user switches `gross` for `studio` in the rewriting shown in in Figure 1a).

Exploring Table Schemas. Besides allowing queries over the contents of HTML tables, `Mesa` also allows users to explore their schemas. The *"Suggestions"* link (located above the `Table Search` box, Figure 1a)) can be used to discover table schemas that are related to the user query. For instance, Figure 1d) shows schemas that are related to the query `select title, actors from movies`. Each entry in the result consists of the schema description and the number of source tables that use this schema. The schemas are ranked according to their relevance to the user query. Schemas that cover more columns of the user query are deemed more relevant. If two schemas cover the same number of columns, the one that can be found in the higher number of source tables is deemed more relevant. When the user clicks on a schema, the rewritings that use the selected schema become available.

Users can also browse the different schemas for all tables in the repository using the *"Repository"* link (located above the `Table Search` box, Figure 1a)). These schemas are ranked according to their frequency, i.e., the number of source tables in the repository which follow that schema. Similarly to the *Suggestions* link, when the user clicks on a schema, the rewritings that use the selected schema become available.

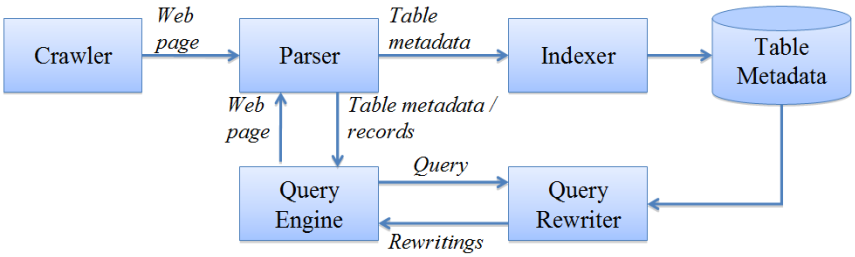

Figure 2. High-level architecture of *Mesa*

Selection Predicates. Selection predicates (such as `year > 1999`) are used to conditionally select data from a table. `Mesa` uses summarization techniques in order to verify which source tables can contribute with at least one record that satisfies the predicates without having to navigate through the actual source pages. This means that the rewritings will always bring some result that satisfies the selection predicates (as long as the content of the page did not change since the last time it was indexed).

3. System Architecture and Implementation

As illustrated in Figure 2, `Mesa` consists of five components, which we briefly describe below.

Locating Web Tables. The *Crawler* is responsible for finding relevant structured Web sources. Because HTML tables are sparsely distributed on the Web, we use a focused crawler for this task. For the current prototype, which allows searches over movie information, the ACHE focused crawler [Barbosa and Freire 2007] was configured to locate pages that contain HTML tables whose header has at least 3 columns and where either `title`, `film` or `movie` appear in the header. The purpose of these constraints is to: i) detect tables that are used for data tabulation purposes instead of formatting purposes; and ii) discover relevant sources of data for the movie domain.

As the seed for the crawler, we used a Wikipedia page that brings a list of the 100 best American movies [1]. We stopped crawling after 400 web pages were retrieved. From this collection of pages, we extracted 993 HTML tables that satisfy our condition. We manually checked a sample of 10% of the collection and we verified that 98% of the extracted tables are indeed related to movies.

Extracting Table Schemas and Content. The *Parser* is responsible for extracting the table schema and records from the retrieved source pages. For our prototype, we implemented a simple parser that automatically extracts this information from well-behaved source tables (i.e., HTML tables whose header row is defined with TH tags). The output of the Parser is used by the *Indexer* (for populating the repository) and by the *Query Engine* (for constructing the query results).

Indexing Tables. The *Indexer* is used to build a document index and a table index. The document index is a full-text index built using Lucene [2]. This information is used to

[1]The list was created by the American Film Institute `http://en.wikipedia.org/wiki/100_Years...100_Movies`

[2]http://lucene.apache.org

answer queries that contain keywords. The table index stores information about the source tables. This index provides the the Query Rewriter quick access to source tables that contain columns that match the user query (see details below). In addition, the table index keeps track of the position of the table inside the page. If a page contains many tables, the parser uses this information in order to locate the correct one. The table index also maintains a summary of the table contents which can be used to prune rewritings when the query uses selection predicates.

Rewriting Queries. When a user issues a structured query, the *Rewriter* uses the information in the repository to derive rewritings for the user query. The rewriting derivation process involves finding which source tables match the tables described in the query. Given each query table, the process is divided in two steps. In the first step, the rewriter matches the columns of the query table with the columns that can be found in the repository. This match is computed using a normalized string similarity technique. In the second step, we use the vector space model to compute the cosine similarity between a query table and the source tables. The weights of the vectors are derived from the column matches computed in the first step [3]. The most similar source tables are then used in the rewritings (i.e., tables whose cosine similarity is greater than a predefined threshold). The rewritings are ranked according to the similarity between the query tables and the source tables that they represent. Source tables that cover more columns of a query table are more likely to have a better ranking.

Evaluating Queries. The *Query Engine* executes a rewriting and displays the results to the user in a tabular fashion. The query engine treats a query as a tree whose nodes are operators of the relational algebra (i.e., select, project, join and sort). Some very basic optimization techniques are implemented, such as pushing the select operators down and using merge-join to combine the data from two different tables.

4. Related Work

Because previous approaches to data integration rely on pre-defined mappings between a global schema and the underlying information sources [Ullman 2000, Halevy 2000, McBrien and Poulovassilis 2003, Friedman et al. 1999], they are not suitable for integration tasks on the Web, where there is a very large number of information sources and these sources are highly volatile. Instead of requiring mappings to be pre-defined, based on source metadata, `Mesa` derives mappings on-the-fly in response to user queries.

Cafarella et al. [Cafarella et al. 2008] also addressed the problem of querying Web tables. Their approach, however, is limited to keyword-based queries. They build a full-text index of the table contents and rank the query by giving a higher relevance to some table features, such as the information on the header.

5. Conclusions

In this paper we described `Mesa`, a new system for querying structured Web information. Because `Mesa` does not require pre-defined mappings, it is scalable: new sources can be added to the system and queried without the overhead of creating new mappings (or

[3] The name of the tables is not used in the rewriting, only the columns names are considered.

updating a global schema). But scalability comes at a cost: since `Mesa` uses a best effort approach to match user queries against the information in the sources, it cannot provide guarantees with respect to recall and precision. Nonetheless, our preliminary experiments with a prototype we built to search over HTML tables in the movie domain have shown that the approach is promising and indicate that the system is effective at helping users explore structured information sources and discover how they can be connected. This prototype, which currently indexes close to one thousand tables, can be accessed at `http://cumbuco.cs.utah.edu:8080/servlets-examples/servlet/Mesa`.

A limitation of the current prototype is that it requires users to formulate SQL queries that specify how the different tables should be joined. It is well known that keyword search forms are easier to use than SQL. However, their expressiveness is limited. As future work, we intend to investigate alternative query interfaces that provide a better balance between expressiveness and simplicity of use.

References

Barbosa, L. and Freire, J. (2007). An adaptive crawler for locating hidden-web entry points. In *WWW*, pages 441–450.

Cafarella, M. J., Halevy, A., Wang, D. Z., Wu, E., and Zhang, Y. (2008). Webtables: Exploring the power of tables on the web. In *VLDB (to appear)*.

Friedman, M., Levy, A. Y., and Millstein, T. D. (1999). Navigational plans for data integration. In *AAAI/IAAI*, pages 67–73.

Halevy, A. Y. (2000). Theory of answering queries using views. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 29(4):40–47.

McBrien, P. and Poulovassilis, A. (2003). Data integration by bi-directional schema transformation rules. In *ICDE*, pages 227–238.

Ullman, J. D. (2000). Information integration using logical views. *Theoretical Computer Science*, 239(2):189–210.