On Mediated Search of the United States Holocaust Memorial Museum Data

Jefferson Heard*, Jordan Wilberding*, Gideon Frieder**,
Ophir Frieder, David Grossman, and Larry Kane*

Information Retrieval Lab
Illinois Institute of Technology

Abstract. The United States Holocaust Memorial Museum (USHMM) recently celebrated its ten-year anniversary. The museum was established to bear witness to the human atrocities committed by the Nazi reign of terror. As such, related data must be collected, and means to store, search, and analyze the data must be provided. Presently, the data available reside in various formats, sizes, and structures, in videotape and films, in microfilms and microfiche, in various incompatible structured databases, as unstructured electronic documents, and semi-structured indexes scattered throughout the organizations. Collected data are partitioned over more than a dozen languages, further complicating their processing. There is currently no single search mechanism or even department of human experts that can sift through all the data in a fashion that provides global, uniform access. We are currently experimenting with our developed Intranet Mediator technology to provide answers, rather than a potential list of sources as provided by common search engines, to questions posed in natural language by Holocaust researchers. A description of a prototype that uses a subset of the data available within the USHMM is described.

1 Introduction

Quoted directly from its website, the mission of the United States Holocaust Memorial Museum (USHMM) states:

> "The United States Holocaust Memorial Museum is America's national institution for the documentation, study, and interpretation of Holocaust history, and serves as this country's memorial to the millions of people murdered during the Holocaust."

We are developing a Mediator search tool to provide a natural language, single point of querying interface to the United States Holocaust Memorial Museum data in all its various formats. This effort is conducted independently from, but in cooperation with the archives division of the USHMM.

* Also of Intranet Mediator Inc.
** Also of George Washington University.

The Intranet Mediator [3] is a search tool based on patented technology [2] capable of providing a layer of abstraction over structured (SQL) databases, semi-structured (XML, MARC, LOC) document repositories, and unstructured text repositories using their own search engines. This layer of abstraction allows the user a natural-language interface to query all of these repositories in an intelligent manner from a single place. It also places no requirements on the underlying systems that it searches. We have developed a system that allows a researcher to enter a natural language question and query up to six sources, some of which are structured, semi-structured (XML), and text, all at the direction of the Mediator.

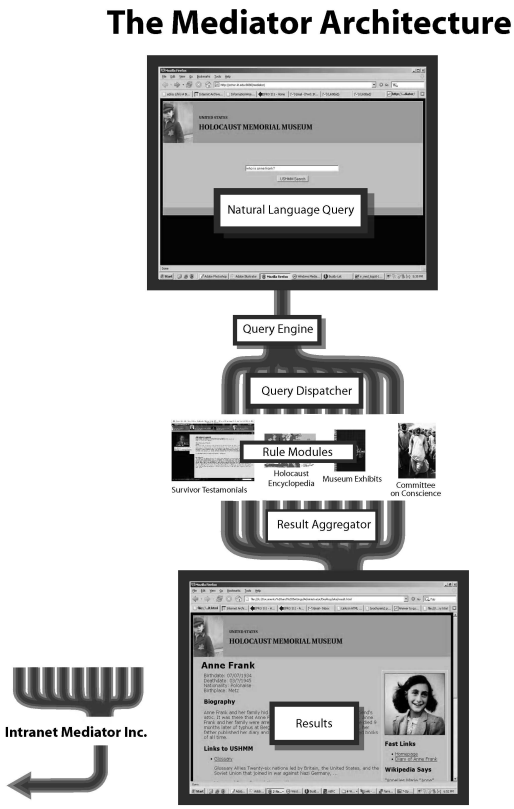

Fig. 1. Architecture of the Intranet Mediator.

The USHMM's archives contain many collections in languages other than English. In addition to our work in this study, we are developing a method of querying these collections in their own languages and returning interpretable results. Towards this, we have included in our prototype an English search over the Slovak Jewish Census collection, a small index of boxes of text and film documents that contains abstracts written in the Slovak language of each record. In the search of these documents, results are presented to the user in their original language of Slovak. For description on recent efforts in cross language information retrieval, see [4].

2 Related Work

Question and Answer systems abound and include the likes of START [7] and JAVELIN [9]. These systems handle natural language questions, like our Mediator, however they operate on only one kind of data and most often use "canned" answers, such as Wikipedia for their results. In addition, the Mediator uses techniques from Natural Language Processing, such as named-entity extraction [8] and text classification [1] to extract the neccessary data from the queries to pose them to the different types of systems and also to direct them towards the most salient procedure for obtaining and composing a final answer.

3 Solution Architecture

The global architecture of the Intranet Mediator is illustrated in Figure 1. When a user issues a natural language question to the Mediator, a *Redirector* routes it through several levels of processing and then directs the user's browser to the answer. Inside the *QueryEngine*, the user's query is first broken up by a *Tokenizer*. The sequence of tokens is then analyzed by a *KnownEntityTagger* that finds, concatenates, and tags subsequences that exist within the structured repositories available to the Mediator. This sequence of tagged data is passed through an *InferredEntityTagger*, which further tags data through finding patterns that have previously surrounded a known entity. The *KnownEntityTagger* finds entities in the query by matching substrings with the results of one or more pre-defined structured queries on the Mediator accessible databases. The *InferredEntityTagger* takes the results from the *KnownEntityTagger* and uses them as training as well as checks them for inferrable entities. The *InferredEntityTagger* uses a simple trigram tagging scheme similar to [6] to find entities.

This digested query is sent to the *QueryDispatcher*, which passes the query through a *BayesianLogisticRegressionClassifier*, selecting the top-level rule with the best potential to answer the question. The sequence of tagged entities and the name of the selected rule are then hashed together to determine if the answer has already been cached. If the question has no cached answer, the top-level rule is given the query and entity values and the *QueryDispatcher* executes it. A top-level rule, shown in Figure 2, consists of:

```
<rule name='Person'>
  <questions recognizer='com.imi.mediator.querydispatcher.BLRQueryRecognizer'>
      <query>Who was $Name?</query>
      <query>Who is $Name?</query>
      ...
  </questions>
  <required-fields>
      <field p='0.0' name='Name' />
      <field p='0.8' name='Birthdate' />
      <field p='0.8' name='DeathDate' />
      ...
  </required-fields>
  <logic>
     required_fields['Name'] = titlecase(required_fields['Name'])
  </logic>
  <answer>
      <subanswer fields='Name'><h1>%</h1><hr /></subanswer>
      <subanswer fields='Birthdate,Deathdate'><h2>(%-%)</h2></subanswer>
      ...
  </answer>
</rule>
```

Fig. 2. A sample rule.

- A set of questions with known answers, found in the <*questions*> section which is used to train the classifier.
- A set of required fields, found in the <*required-fields*> section which is filled in by the entities found in the query and by sub-rules that use the top-level rule's information to spelunk for more data. Assigned along with each field is the probability the top-level rule can answer the query given that the field cannot be provided by either the query or a sub-rule.
- A logic section, denoted as <*logic*>, written in Python [10] that operates via *QueryModules* with the searchable data repositories and the set of fields filled in by the query.
- A set of formatting instructions for the *Aggregator*, the <*answer*> section that take various pieces of the result data and make them human-readable.

A rule, in our approach, is a template that answers an entire class of similarly phrased questions. Once a rule is selected and query fields are filled in, sub-rules are selected to create an execution plan. A sub-rule is a rule that contains no set of answerable questions, but merely a list of fields the rule requires to execute and a list of fields the rule provides upon completion. For each required field in the top-level rule, a priority queue of sub-rules to execute is created. This structure is prioritized by the probability that the sub-rule will assign the correct value to

the field. Once the execution plan is created, the first item of each priority queue is polled, and the logic section is executed. Note that the rules are independent from one another and are executed in parallel. Each time a field is successfully filled in by a rule, the field's priority queue is emptied. As long as there are non-empty priority queues, the execution plan continues. As each field is filled, it is passed to the *Aggregator*, which uses the formatting instructions in the top-level rule to build the answer that is displayed to the user.

One of the unique features of the Mediator is that it attempts to return an answer to a question rather than a set of hyperlinks or some other form of a ranked result set, as is common for traditional search engines. While question answering systems abound, no question answering system exists currently that is capable of exploiting multiple types of data as the IIT Intranet Mediator is [3]. Furthermore, the Mediator is actually querying the underlying data as opposed to selecting from a predetermined set of possible answers as in efforts such as Ask Jeeves (ask.com).

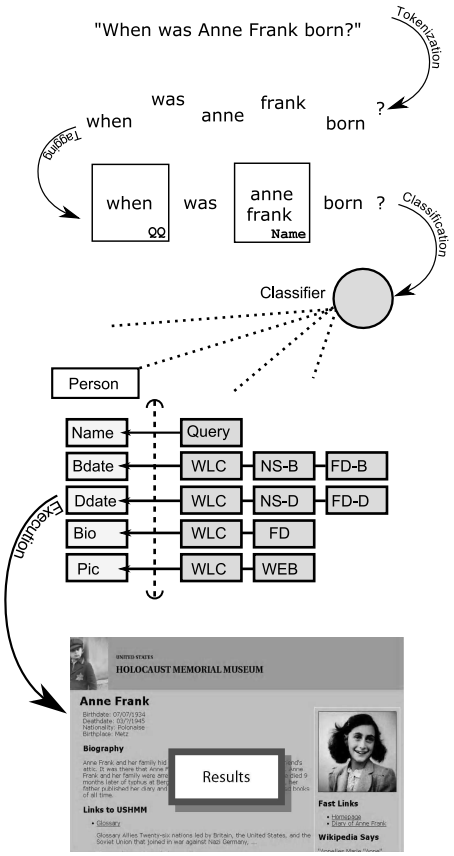

Fig. 3. Flow of "When was Anne Frank Born?" through the mediator

An answer is determined entirely by the top-level rule that a question triggers; therefore it is important that the proper rules exist in the system for each question. In general, one top-level rule should exist for each different type of answer needed. In the case of our prototype system, we created a rule for answering questions about people, one for answering questions about places, and one for events. In a more robust system, these might be broken into more subtle rules, such as answers for questions about a person's biography versus questions about a person's geneology, or questions about concentration camps versus questions about a city. All questions are answered by the Mediator; it is simply a matter of granularity of answer that one needs to consider in designing and implementing rules.

To demonstrate the Mediator, in Figure 3, we illustrate the flow of "When was Anne Frank born?" through the system we developed using the USHMM data. The query is first tokenized, case-folded, and then tagged; since "Anne Frank" exists in the USHMM's structured repositories, this tagging is done via the *KnownEntityTagger*. Then the tagged sequence is passed to the classifier, which decides on "Person" as the rule meaning a person's biographical data. An execution plan is created for the person rule, including data from four separate sources: the query, the USHMM's database of victim and survivor names, a.k.a. "NameSearch", our fact database, and the USHMM web search. The leftmost rule in each queue is executed until there are values filled in for all fields: *Name, Bdate, Ddate, Bio, and Pic*. Each final field value will become part of the final answer, shown at the bottom of the figure. The answer resembles a hand-written encyclopedia article despite the fact that it has been assembled from multiple sources on the fly through the work of the Mediator.

Previously, although many retrieval engines were capable of querying multiple sources, they were limited to a single type of data: structured, unstructured, or semi-structured. The Mediator is needed to provide this one-query, one-result approach to managing user requests over these different kinds of data, because queries on each are different both in terms of the method in which a data source is queried and in terms of the nature of the returned results. For example, it may be difficult to "rank" structured results, as all results of a structured query may provide part of an answer or be used in a statistical fashion to provide a more high-level answer based on low level data. However, on an unstructured text repository, ranking of whole documents may be the only way to return data. A single query will behave in vastly different ways depending upon what kind of data it is querying. Mediation is a way of intelligently managing these different behaviors. Our approach to this is that each different type of source is handled by a query module: a highly configurable object that connects to a data source, issues a query, and returns results in a uniform way. In our Mediator, we have the implemented a JDBC query module to handle SQL queries over structured data, a CGI query module to handle queries to any web-based CGI script via HTTP GET or POST requests, an XPath query module to handle semi-structured queries against XML repositories, and Google, Yahoo, and AIRE[5] modules to handle unstructured searches over text collections.

To compose an anwer from the results returned through these multiple types of queries on multiple sources, we employ a final processing engine called an *Aggregator*. The *Aggregator* takes each field provided and decides how to display it. When there is duplicate or conflicting data this is handled by dropping all duplication except the data with the highest probability of correctness and displaying only that. The Aggregator will display all fields that have confidences above a configured threshold. A simple printf-like XML template language is included as part of a toplevel rule definition which determines the final formatting of displayed results.

Our solution consists of the Mediator running on a single HP DL-380 running Gentoo GNU/Linux 2005.1 with access to data sources residing on the local machine, other machines on the LAN, and at the USHMM. We have rules for answering questions about people, places, events, and a default rule that answers unknown queries with a set of hyperlinks via a traditional search engine. To date, six data sources are incorporated into the answers we provide:

- USHMM web-site text search
- Learning Center Database: an MS SQL Server database with the full content of the USHMM Permanent Exhibit, including pictures, videos with their subtitles in French, Spanish, and English, text snippets with structure associated, name databases, and more
- Hand-built database of facts, including birthdates and deathdates and some small amount of biographical data on famous people from World War II.
- USHMM NameSearch, a CGI-interfaced database that resides on the website and contains thousands of names of survivors and victims and their associated data.
- Index of the archives of the ministry of interior of the Slovak government in the years 1938-1945 in the Slovak language.
- Semi-structured location database of townships, metropolitan areas, and counties in Slovakia that had Jewish residents before the Holocaust.

Table 1. Frequency of most common queries.

Query	Occurrences	Percent of Log
Holocaust	6409	2.37%
Auschwitz	5190	1.92%
Anne Frank	4799	1.78%
Concentration Camps	3753	1.32%
Hitler	2934	1.09%

We obtained from the USHMM a query log providing all user searches over an six-month period of time, including major Jewish holidays, for a total of approximately 270,000 queries. A team of researchers analyzed these queries for uniqueness and total number of occurrences within the set, and we set out to cover as many of the most common queries as possible. We treat queries referring to the same named entity, such as "Adolph Hitler" and "Hitler" as equivalent, since our answer should be the same in both cases. In Table 1, we illustrate the five most common queries and their frequencies across the log. We created several query types and tested these against a sample of the query log to determine the coverage of queries that were handled in some manner more intelligently than simply passing all the query terms along to the USHMM web search engine. That is, we determined the coverage of queries that experienced some integration of data across several sources.

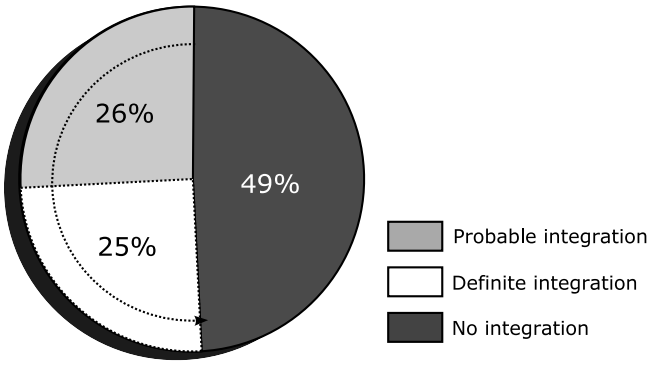

Fig. 4. Percentage of queries in log benefiting from mediation

To analyze our system's coverage of the query log, we first sorted the query log by the number of occurrences of each query, then took the top 50% of the queries in the sorted list. These queries were issued to the Mediator and 50% reported some form of data integration beyond the unstructured USHMM web search. In Figure 4, these queries form the pie slice designated "definite integration". After we had confidence in these results, we took a random sample of one thousand queries throughout the whole query log and issued these to the mediator, with a resulting 51% of these queries returning results integrated from more than the unstructured USHMM web search. In Figure 4, the additional handled queries implied by our statistical sample form the pie slice designated "probable integration". The pie slice labeled "no integration" covers only those queries that would return answers containing only links from the web search.

4 Future Work

We would like to develop a way to analyze query data so that likely rules can be automatically suggested, if not added to the system. As we added more rules to our engine, the number of mediation-enhanced queries grew rapidly. We do not yet know the trade-off point where adding new rules does not significantly increase the coverage our engine gets. As more sources were made available to the engine, its answers became more accurate and better focused. We are working to determine a metric for how to define "accurate" so that we can measure the amount of contribution having a particular data source available makes to the overall answer.

5 Acknowledgments

We gratefully acknowledge the cooperation of United States Holocaust Memorial Museum in providing access to all data used herein. We also acknowledge the Holocaust IPRO team from the Illinois Institute of Technology. Special thanks go to Steven Beitzel, Eric Jensen, and Michael Lee, the primary architects and developers of the initial Intranet Mediator[3]. We gratefully acknowlege the Intranet Mediator, Inc. for its generous support. This effort is dedicated to the millions who lost their lives in the Holocaust. HY"D.

References

1. S. Eyheramendyl, A Genkin, W. Ju, D. Lewis and D. Madigan. Sparse Bayesian Classifiers for Text Categorization. Journal of Intelligence Community Research and Development, Volume 13, 2003.
2. O. Frieder and D. Grossman. Intranet Mediator. US Patent #6,904,428, June 2005.
3. D. Grossman, S. Beitzel, E. Jensen, and O. Frieder. IIT Intranet Mediator: Bringing data together on a corporate intranet. IEEE IT PRO, January/February 2002.
4. D. Grossman and O. Frieder. Information Retrieval: Algorithms and Heuristics. Springer Publishers, 2nd ed., 2004.
5. T. Infantes-Morris, P. Bernhard, K. Fox, G. Faulkner, K. Stripling. Industrial Evaluation of a Highly-Accurate Academic IR System. ACM CIKM, New Orleans, Louisiana, 2003.
6. D. Jurafsky and J. Martin. Speech and Language Processing. pp. 577-583. Prentice Hall, 2000.
7. B. Katz, G. Marton, G. Borchardt, A. Brownell, S. Felshin, D. Loreto, J. Louis-Rosenberg, B. Lu, F. Mora, S. Stiller, O. Uzuner and A. Wilco. External Knowledge Sources for Question Answering. In the Proceedings of TREC-2005, Gaithersburg, Maryland, November 2005.
8. C. Manning and H. Schutze. Foundations of Statistical Natural Language Processing. MIT Press, 1999, pp.353
9. E. Nyberg, R. Frederking, T. Mitamura, M. Bilotti, K. Hannan, L. Hiyakumoto, J. Ko, F. Lin, L. Lita, V. Pedro and A. Schlaikjer. JAVELIN I and II Systems at TREC 2005. In the Proceedings of TREC-2005, Gaithersburg, Maryland, 2005.
10. The Python Language. http://www.python.org.