A Flight Meta-Search Engine with Metamorph [*]

Bernhard Krüpl
Wolfgang Holzinger
Yansen Darmaputra
DBAI Group
TU Wien, Austria
{holzing,kruepl,darmap}@dbai.tuwien.ac.at

Robert Baumgartner
Lixto Software GmbH
Vienna, Austria
baumgartner@lixto.com

ABSTRACT

We demonstrate a flight meta-search engine that is based on the Metamorph framework. Metamorph provides mechanisms to model web forms together with the interactions which are needed to fulfil a request, and can generate interaction sequences that pose queries using these web forms and collect the results. In this paper, we discuss an interesting new feature that makes use of the forms themselves as an information source. We show how data can be extracted from web forms (rather than the data behind web forms) to generate a graph of flight connections between cities.

The flight connection graph allows us to vastly reduce the number of queries that the engine sends to airline websites in the most interesting search scenarios; those that involve the controversial practice of creative ticketing, in which agencies attempt to find lower price fares by using more than one airline for a journey. We describe a system which attains data from a number of websites to identify promising routes and prune the search tree. Heuristics that make use of geographical information and an estimation of cost based on historical data are employed. The results are then made available to improve the quality of future search requests.

Categories and Subject Descriptors: H.3.4 [Information Storage and Retrieval]: Systems and Software

General Terms: Algorithms, Design, Experimentation.

Keywords: Hidden Web, Web Data Extraction, Web Form Mapping, Web Form Extraction.

1. INTRODUCTION

A typical flight meta-search engine forwards the route query that it receives to other websites in its domain. The retrieved results are aggregated and presented to the user. We implemented a system that mimicks the so called creative ticketing practice of travel agencies by searching for the more complex itineraries that involve changing airlines during the journey. Many special air fare offers are only available through the airline's websites; but each request to an airline website is a costly procedure. We thus implemented a system that uses a flight connection graph and a query planner to limit the number of requests by checking only the most promising paths. Our system has been built on top of Metamorph, a meta-search framework aiming to generate maintainable vertical deep web search engines [1]. We use Metamorph's ontology and rule based form interaction modelling to map and fill out the web search forms.

In related research, the WISE project [2] aims to integrate access to web databases. The MetaQuerier [3] uses the regularities of web forms and automatically matches interfaces. Metamorph focusses on modelling form interactions, and our meta-search system retrieves its initial route knowledge from web forms themselves and concentrates on creative ticketing heuristics via hub identification.

2. EXTRACTING DATA FROM FORMS

Web forms are the gateway to the hidden or deep Web. It is quite clear that any automated search system has to model the meaning of these forms in order to be able to pose queries. There is a wealth of literature about mapping forms, and a number of the suggested solutions makes use of the information on the forms themselves such as labels. Many modern web forms though employ Javascript and AJAX to provide instant feedback about the validity of data, or even suggest possible data. A technique called dynamic dependent drop-down lists populates a drop-down list according to what a user selected in another drop-down list. In the flight search domain, this is often used when a destination airport list is automatically updated when a user selects an origin airport.

In the process of building a model for a web search form, Metamorph uses its ontologies to identify and tag known concepts in that form. We can use data extraction from forms to extract relations between such concepts; in flight search, the most important relation is the one between origin and destination airport. Form data extraction works like this: The connector, which has access to a full-featured web browser component, issues a click event on the origin airport list and sequentially selects each of the options on that list. After that, it adds an event hook to monitor the destination airport list for possible changes. The changes could be done by running client-side Javascript code or by reloading from the server. If changes happen, the destination airport list is analysed for airport or IATA names and stored in the local database together with the origin airport selection it depends on. This method works on large number of airline websites, and is good enough to bootstrap the system with initial knowledge about flight connections.

Currently, we are generalizing our form extraction approach by automatically detecting dependencies between form controls and integrating the results directly into the Metamorph form ontology.

[*]This research is supported in part by the Austrian Forschungsförderungsgesellschaft FFG under project grant 812991.

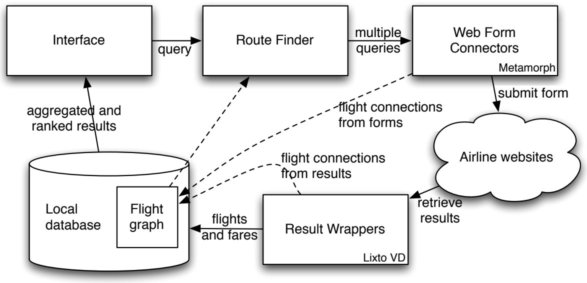

Figure 1: flight meta-search architecture

3. MODELLING THE SYSTEM

Our aim was to develop a light-weight flight meta-search system that can be bootstrapped with a minimal amount of data. Figure 1 shows the basic building blocks of our system. There are two components that handle the interaction with airline websites: web form connectors and result wrappers. The web form connectors fulfil two different purposes: To extract an airport connection matrix from the form itself whenever possible; and to map and process the interactions that are required to submit a request. The automatic, supervised creation of the connectors [4] is part of the Metamorph framework. Results are piped into wrappers that were interactively generated on a per-website basis using the Lixto Visual Developer tool. All retrieved data is stored in a local repository, which also serves as basis for route finder.

The system is bootstrapped by providing a set of airline website URLs that contain search forms and a set of wrappers that can interpret their result pages. Initially, the system tries to collect as many flight connections as possible from the web forms themselves and stores the connections in a local flight connection graph. No additional knowledge is provided, and all other processing takes place whenever an end user poses a query via the system's search interface.

These steps take place when a query containing origin, destination airport and date of departure is submitted to the system: The route finder uses the available data to generate a number of promising routes. E.g., when a user wants to research flights from Vienna to Singapore, in addition to the obvious search from Vienna to Singapore, the route finder will additionally generate search requests for flights from Vienna to promising hubs such as Paris and flights from there to Singapore. These requests are then forwarded to an execution engine that uses the appropriate connectors and wrappers to check the price and availability of the particular flight at an airline's website. Finally, the results are aggregated, re-ranked, and presented to the user; the fares are stored in a database to provide hints for future requests.

4. ROUTE AND HUB FINDER

The search for direct flights from origin to destination fulfils two purposes: It provides us with a gold standard fare that the system will have to beat with creative ticketing; and it will enable the system to grow its flight connection graph by adding direct connections as well as connections to the transit cities that will often appear on the result pages of larger carriers. This is how the graph is used to find routes which should be checked:

First, the route finder uses a Dijkstra algorithm to find the cheapest routes in the system's flight connection graph if fares are already available in the database. Second, direct origin-destination routes for the most promising (according to historic fare data) airlines are generated. Third, routes to promising transit points are generated. This is essential for the creative ticketing approach that we have explained above, because this kind of routes will not show up in normal flight searches. Since the flight connection graph that is available to us is incomplete, we have to guess a limited number of itineraries that could be cheap. We do this by looking for routes through what we call promising hubs.

We use a set of heuristics to rank airports to become promising hubs: A high number of airlines that serve an airport increases promising hub likelihood, as well as its appearance as a transit point in result pages of airlines. By using a geo location service, we can omit hubs from the search that are obviously too far away. Additionally, we can also include nearby airports by assuming default ground transfer times per kilometre.

5. EVALUATION AND CONCLUSION

The system was implemented using our own Metamorph framework for form mapping and ontological representation together with all related ontology tools (esp. the ontology editor and Jena), Lixto Visual Developer, and PostgreSQL/PostGIS. We attached 15 airline websites to the system and developed a Lixto wrapper for each of the result documents. From all airline websites, 8 contained dynamic drop down-lists that we could use for the extraction of data. Upon bootstrapping, the flight connection graph contained 1612 connections, and was expanded to 2442 connections after 100 randomly generated requests. Interesting creative ticket itineraries showed up after about 20 queries.

Unlike available flight search systems, our approach supports dynamic packaging of connecting flights from different airlines, including the consideration of hub points that are usually neglected. We are well aware that our system is not complete. Neither the change of fares over time, nor for different times of a day are considered by the route finder. Also, the logic for allowing sufficient transit time at airports is immature.

On the other hand, the main motivation in our research is to prove the validity of our approaches in form interaction modelling, form mapping, and form data extraction, and not to come up with a commercial search system. We believe we have shown that the development of a lightweight meta-search system with the tools at hand is feasible, that interesting information can extracted from web forms themselves, and that an intelligent query planning in a domain such as flight search can provide good results by issuing just a limited number of requests.

6. REFERENCES

[1] Metamorph, A Framework for Automated Ontology-driven Metasearch Generation (2008), *Technical Report TR-DBAI-2008-3*

[2] He H., Meng W., Yu C., Wu Z. (2003). WISE-Integrator: An Automatic Integrator of Web Search Interfaces for E-Commerce.

[3] He, B., Zhang Z., Chang K. (2005). Towards Building a MetaQuerier: Extracting and Matching Web Query Interfaces. In *ICDE*.

[4] Holzinger, W., Krüpl, B., Baumgartner R. (2008). Exploiting Semantic Web Technologies to Model Web Form Interactions. In *WWW2008* .