A Deep Web Data Integration System for Job Search

□ Liu wei[1], Li xian[1], Ling yanyan[1], Zhang xiaoyu[2+], Meng xiaofeng[1]

[1]School of information, Renmin University of China, 100872

[2]School of Software, Beihang University, 100083

Abstract: With the rapid development of Web, there are more and more Web databases available for users to access. At the same time, job searchers often have difficulties in first finding the right sources and then querying over them. Providing such an integrated job search system over Web databases has become a Web application in high demand. Based on such consideration, we build a deep web data integration system that supports unified access for users to multiple job web sites as a job meta-search engine. In this paper, the architecture of the system is given first, and the key components in the system are introduced.

Keywords: Web database; Web data integration; job web site

CLC number: TP393.09

Received date:

Foundation item: Natural Science Foundation of China (60573091, 60273018); China National Basic Research and Development Program's Semantic Grid Project (2003CB317000); the Key Project of Ministry of Education of China (03044).

Biography: Liu Wei (1976-), Male, Ph.D. candidate, and Research Direction: Web Data Integration. Email: gue2@ruc.edu.cn

+ To whom correspondence should be addressed. Email : zhangxiaoyu912@gmail.com

0. Introduction

More and more databases are becoming Web accessible through form-based search interfaces. We call this kind of web data "Deep Web", the online databases "Web database", and the form-based search interfaces "query interface". The survey[1] in April 2004 estimated 450,000 online databases. As current crawlers, such as Google, cannot effectively query databases, such data are invisible to search engines, and thus remain largely hidden from users. It is of great importance to provide a unified access to multiple Web databases providing similar information because this would allow users to search and compare information from multiple sites in a same domain with ease.

At present, more and more jobs provided by corporations are published by job web sites. CompletePlanet[2], the largest depository of Deep Web, has collected more than 1,000 job web sites, but it is still a very small portion compared to the total number of job web sites in the Web[1]. Most of the job web sites have stored the job information in their Web database. At the same time, there are also myriads of persons searching for their desired jobs in the extensive Web by filling in query interfaces. Facing on so many job web sites, it is almost an impossible mission for a user to search all the job web sites one by one manually.

Based on such consideration, in this paper, we build a data integration system that supports unified access for users to multiple job web sites as a job meta-search engine (JMSE for short). Currently, there are a number of JMSEs on the Internet, such as www.flipdog.com, www.globehr.com, and www.deepdo.com, etc. However, their techniques are not publicly available. To the best of our knowledge, most existing JMSEs are built manually or semi-automatically. Furthermore, as JMSEs operate autonomously, changes/upgrades to them may affect the operation of the JMSE. As a result, maintaining the operation of a JMSE is a costly long-term effort.

Our project is an intact application for Deep Web data integration. We provide a comprehensive solution to the problems of automatically integrating the Deep Web data of job web sites. This project consists of a number of components, and each component has solved a sub-problem in the project. The architecture of our system will be discussed in the next section. There are already a number of works on Deep Web data integration we can resort to. Some new solutions are also proposed for the components which current works have still not touched.

1. The architecture and key techniques in our system

Our system aims to automate the process of building large-scale JMSEs so as to significantly reduce the cost of building and maintaining JMSEs. Fig 1 shows the architecture of Deep Web data integration system for job web sites.

This system consists of a number of components. *Job Web Site Discovery* is a special crawler which can discover job web sites on the Web efficiently and accurately; *Query Interface Extraction* is to find the real query interface from the job web sites collected in last component; *Interface Integration* integrates the query interfaces into a global interface which can give users a unified access; *Query Process* translates the query on global interface into a set of queries that can be processed by the query interfaces of job web sites; *Query Submission* and *Response Pages Retrieval* are to send a query to its corresponding web site and save the response pages in local computers; *Web Data Extraction* extracts the pure results from the saved response pages in a customized format; *Data Merging* merges the results from different job web sites into a global table in relational database.

1.1 Job Web Database Discovery

In order to perform several of the hidden-web data retrieval and integration task on job area, locating job-relevant data sources is a necessary condition. Given the dynamic nature of the web, where data sources are constantly changing, automatically find the job web sites seems particularly challenging.

Instead of a full crawl of the web, which is highly inefficient and exhaustive, another alternative using a focused crawler limiting the search to the particular job domain has been adopted in our system.

Firstly, given several job web sites as seeds, our focus crawler can start from it. For each seed, we consider all the links within this web site that no exceed than depth 3, due to the observation that all Friendship links provided are usually within depth 3. The *page classifier* is trained to classify pages as belonging to job area. Rainbow[3], a freely-available naïve Bayes classifier was used to build our page classifier. When the crawler retrieves a page P, the page classifier analyzes the page and assigns it score which reflects the probability that P belongs to the job topic. If this probability is greater than a certain threshold (0.5 in our case), the crawler regards the page as relevant. Once a page is classified as being on-topic, we shall put it into a candidate set, and add this newly found page as seed to do iterative crawl in the same way until the volume of the candidate set goes convergent.

For those pages which contain search forms were preserved, we need to filter out non-searchable forms. The *form classifier* is needed to filter out useless forms. In our algorithm, a decision tree is employed to determine whether a form is searchable or not.

Firstly, a training set of pages was obtained from the UIUC repository[4]. The authors manually judged sample pages and each form were labeled as either a search form or not. For each form, we obtained the following features: number of password tags; number of buttons; number of resets; number of textboxes; number of checkboxes; number of radio tags; number of image inputs; number of items in selects; submission method (post or get); and whether the string “search” are within the form tags.

We perform the learning task using the training set mentioned above and the produced decision tree is able to efficiently perform query interface discovery with low error rate. All those searchable forms are then selected as the input of the following interface integration process.

1.2 Interface integration

Interface integration aims to automatically build a unified access to multiple data sources which have different query forms, called *local interfaces*. There are two principles we need follow when integrating the local interfaces into one: first, the integrated interface should not weaken the query ability of any local interface; also it can’t be so complex for the users’ use. In this case, we build two kinds of interfaces for the user. The first one is a complex interface, which merges all the local interfaces into one. It may include the working place, the demanding salary and so on. Another is providing a simple textbox for keyword search, which the users can fill with any conditions. This makes our system different from the other integration system like [5, 6].

For the simple keyword search, the main problem is how to transform the users’ keyword queries into a set of queries which can be supported by each local interface. In contrary, the

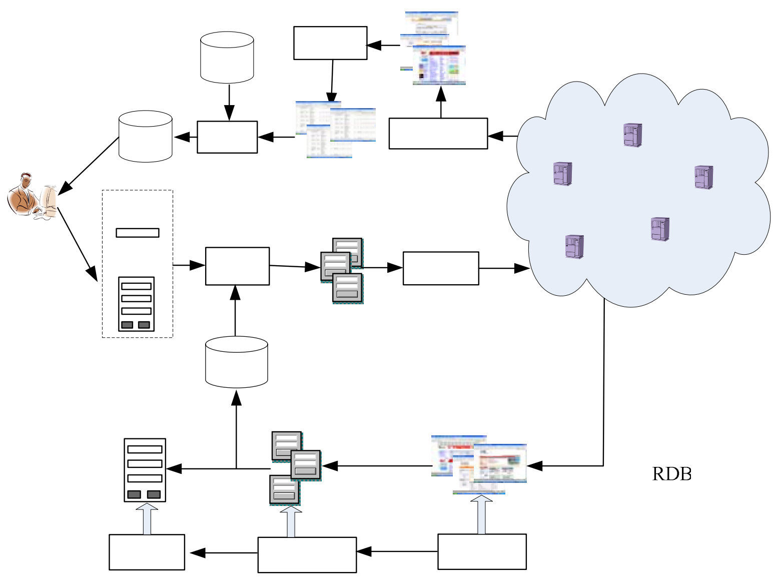

Fig 1. The architecture of Deep Web data integration system for job web sites

treatment of complex integration is far beyond simply merging all the elements of local interfaces together. Mainly it contains three steps.

As the schema of local interface is not formally defined, the first essential step is to understand the search interfaces. Typically we model a local interface as a container filled with a number of *attributes*, each of which consists of a semantic label and a control element which allows the users to enter search information.

The attribute mapping process is mainly carried out based on the attributes' information. We use WordNet[7] to identify the semantic relationship between attributes, according to the term of their semantic labels. The relationship includes: synonymy, hyponymy, etc.

The last step is to merge attribute domains. This includes the following two aspects: 1) global domain mapping, 2) the attribute global value set determination.

1.3 Query processing

After build an integrated interface, the user only need to submit their job requirements in it, and the query will be dispatched to all the local interfaces to retrieve job information. Because the integrated interface is a union of several local interfaces, its query ability is much more powerful than the local ones. The queries need be transformed into a set of queries that the local interface can support. Specially, in our system we also provide another form of integrated interface which is a single keywords search text box allowing be filled with all kinds of job hunting conditions. So the query transformation problem for us is more complex.

First, the query transformation from the complex integrated interface to the local ones is proposed. During the query transformation process, we need to reconcile three levels of query heterogeneities: attribute level, predicate level and query level[8]. Let's denote the integrated interface to be UI, and the target local interface to be L_i.

(1) *Attribute level*: If the user gives constraint on the attribute A1 of UI, so in L_i which attribute should be given this constraint. This problem can be solved using the attribute mapping relation which has been built in the interface integration step. When we determine a global attribute for a group, we also know which attributes in the local interfaces is the same as it.

(2) *Predicate level*: When a user assign a required value to an attribute. So the predicate mapping is to transform a predicate as close as possible which means subsume the query with fewest extra answers. If the user gives a constraint value in UI which is exact the value appearing in L_i or in Li it can be filled discretionarily, then the predicate doesn't need transform. Otherwise, the transformation component will choose a value which is most similar with the user's constraint.

(3) *Query level*: The local interface may have different capabilities on querying valid combinations of predicates. For example, the user will retrieve no job information if only providing the salary requirement. The site requires the user choose a job location or a job type as well. In our system we adopt the algorithm proposed in [9].

1.4 Web data extraction

When the response pages are saved locally, the pure results need to be extracted from these poor structured pages and translated into a special format, such as XML. Fortunately, there are already enough works we can resort to, and most of them are implemented based on tag tree and reach a very high accuracy. We select [10] to implement Web data extraction.

[10] is a domain-independent approach. It extracts the results by a two-step strategy: first, segment data records; second, align and extract corresponding data items from the discovered data records. With an eye to the job web sites, the results in response pages are always shown in form of table, a little change is done in order to suit for our system. Generally, there are the annotations for the results in the top, such as job title, company, and location, etc. So in the process of extraction, the annotations are also extracted, if they exist. In practice, the precision and recall can almost reach 100%. If the accuracy can not reach 100% for some web site, manual intervention will be adopted.

Because we only resort to an available approach to implement Web data extraction, the details of implementation is not discussed in this paper.

1.5 Data Merging

Until now, we have extracted all and pure data in response pages. It is necessary to store all data under a specific global schema, which is also convenient for extending system when there are new websites added into system. Our work currently focuses on finding an approach to obtain whole attribute name structure with all important attributes and determine the position of every data under their attributes name in the process of data merging.

In the process of data merge, all data has been annotated could be easily filled into global result table. For the data without annotation, we use instance-based method to distinguish them each other, finally and fill the data into its corresponding position under the global schema.

In general, there are several approaches to achieve the global schema of response pages. However, for the specific information contained in job web sites we collected, we use the instance-based method[11] to achieve global schema for results. According to our observations, it is easy to distinguish several attributes depend on data type. Some attributes composed of several high frequency words. Although these high frequency words are highly different from domain to domain, it is easy to be collected and recognized for job domain. In addition, we did not disposal attributes with their nature order in response page, we first disposal the attribute could be easily recognized using instance-based method with high accuracy. The instance-based method includes data type, high frequency words related with attribute.

2. Novel Features of Our System

Our system can not be considered as an ordinary application. It has three main novel features which are described as following.

First, it is not only the integration for Deep Web data, but also the integration for previous research works and technique in this area. In recent ten years, a great deal of researches had been done in the area of Deep Web, but they are dispersed in many issues. It is the first try to combine these works together and form a practical system to provide human kinds an opportunity to enjoy the abundant information hided in Deep Web.

Second, our system was designed in an incompact approach. The aim of us is to provide a test bed for the works which focus on the same issue. For example, there are already several works to address Web data extraction, and we should not judge their performance only by the experiment results reported in their papers. This is due to their experiment results are achieved from different data sets. In our system, any research works can be implemented as a tool and embed. Based on the same data set, we can get an impersonal evaluation.

Third, our system is not only the combination of previous works, but some new researches are also embodied. For example, a simple key-word based query interface is devised in our system. A novel technique of query translation is involved by this component. From this angle, our system provides a stage to discover interesting issues and address them.

The above is only the main features of our system, and we do not go into the detail due to the limitation of paper.

3. Related Work

As our best knowledge, several Deep Web data integration systems have been proposed until now. The representative systems are E-Metabase[5] and Metaquerier[12].

E-Metabase was the first comprehensive system. It consists of a number of components. First, a special crawler is used to crawl the Web and identify Web databases from the fetched Web pages. Second, the found Web databases are clustered into different groups such that Web databases in the same group sell the same type of products. Third, the interfaces of the Web databases in the same group are integrated into a unified interface. Fourth, a global query submitted to the EMSE is mapped to queries for the underlying Web databases Fifth, a component that is responsible for connecting to each Web database is built so that a query can be passed to and results can be returned back from each ESE. Sixth, information of every product returned by each Web database needs to be correctly extracted from the returned result pages by an information extraction program. Finally, the extracted results from different Web databases need to be filtered according to the global query and then combined into a single list for presentation to the user based on some desired features. It was very similar with our system, and the most significant difference between them is that wise-integrator is domain-independent, while our system aims at job web sites(or domain dependent).

The goal of MetaQuerier is two fold– First, to make the deep Web systematically *accessible*, it will help users *find* online databases useful for their queries. Second, to make the deepWeb uniformly *usable*, it will help users *query* online databases. MetaQuerier was built for dynamic discovery and on-the-fly integration over databases on the Web. This system focuses on the query interfaces processing, and the processing of query results is not involved in detail.

In conclusion, though these systems had proposed their architectures for Deep Web data integration, they are only the prototypes for research. So there is not a practical product at present. In contrast, our system aims to be the first an application product which can integrate the Deep Web information automatically.

4. Conclusion

With the flourish of web databases and the high demand of job searchers, we build a data integration system that supports unified access for users to multiple job web sites as a job meta-search engine. In this paper, the architecture of the system is given first, and the key components in the system are introduced. With this system, job searchers can automatically get their desired job information from the right job web sites. This system can also be applied to other topics with a little modulation, such as e-commerce web sites.

References

[1] Kevin Chen-Chuan Chang, Bin He, Chengkai Li, Mitesh Patel, Zhen Zhang: Structured Databases on the Web: Observations and Implications. *SIGMOD Record* 33(3): 61-70, 2004

[2] http://www.completeplanet.com/.

[3] A. McCallum. Rainbow. http://www-2.cs.cmu.edu/mccallum/bow/rainbow/.

[4] The UIUC Web integration repository. http://metaquerier.cs.uiuc.edu/repository.

[5] H. He, W. Meng, C. Yu, and Z. Wu. Wise-integrator: An automatic integrator of Web search interfaces for e-commerce. *VLDB*, 2003: 357-368.

[6] K. C.-C. Chang, B. He, and Z. Zhang.Toward Large Scale Integration: Building a MetaQuerier over Databases on the Web. *CIDR* 2005: 44-55

[7] WordNet: http://www.cogsic.princeton.edu

[8] Zhen Zhang, Bin He, Kevin Chen-Chuan Chang: Light-weight Domain-based Form Assistant: Querying Web Databases On the Fly. *VLDB* 2005: 97-108

[9] Y. Papakonstantinou, A. Gupta, H. Garcia-Molina, and J. D. Ullman. A query translation scheme for rapid implementation of wrappers. *DOOD* 1995: 161-186

[10] D Hu, X Meng: Automatic Data Extraction from Data-Rich Web Pages. *DASFAA* 2005: 828-839

[11] J Wang, J-R Wen, F H. Lochovsky, W-Y Ma: Instance-based Schema Matching for Web Databases by Domain-specific Query Probing. *VLDB* 2004: 408-419

[12] K C-C Chang, B He, Z Zhang: Toward Large Scale Integration: Building a MetaQuerier over Databases on the Web. *CIDR* 2005: 44-55

姓名(中文)	刘伟	姓名(英文)	Liu Wei	性别	男	出生年月	1976.9
职称(职务)	博士研究生	电话	010-62517876	Email	gue2@ruc.edu.cn		
手机	13811753603	传真		通信地址	中国人民大学品苑 3-537		
研究	中文	Web 数据管理					
方向	英文	Web Data Management					
基金	中文	国家自然科学基金面上项目，国家 863 项目					
资助	英文	Natural Science Foundation of China, National 863 Projects					

姓名(中文)	李忺	姓名(英文)	Li Xian	性别	女	出生年月	1982.9
职称(职务)	硕士研究生	电话	010-62510426	Email	xianli@ruc.edu.cn		
手机	13810099788	传真		通信地址	中国人民大学知行一楼 303 室		
研究	中文	Web 数据管理					
方向	英文	Web Data Management					
基金	中文	国家自然科学基金面上项目，国家 863 项目					
资助	英文	Natural Science Foundation of China, National 863 Projects					

姓名(中文)	凌妍妍	姓名(英文)	Ling Yanyan	性别	女	出生年月	1985.8
职称(职务)	硕士研究生	电话	010-62516183	Email	lingyy@ruc.edu.cn		
手机	13810724158	传真		通信地址	中国人民大学培训一楼 26 室		
研究	中文	Web 数据管理					
方向	英文	Web Data Management					
基金	中文	国家自然科学基金面上项目，国家 863 项目					
资助	英文	Natural Science Foundation of China, National 863 Projects					

姓名(中文)	张晓宇	姓名(英文)	Zhang Xiaoyu	性别	男	出生年月	1983.9
职称(职务)	本科生	电话	010-62760769	Email	zhangxiaoyu912@yahoo.com.cn		
手机	13811004472	传真		通信地址	北京行空航天大学 5-62 信箱 15 班		
研究	中文	Web 数据管理					
方向	英文	Web Data Management					
基金	中文						
资助	英文						

姓名(中文)	孟小峰	姓名(英文)	Meng Xiaofeng	性别	男	出生年月	1964.05
职称(职务)	教授，博士生导师	电话	010-62519453	Email	xfmeng@ruc.edu.cn		
手机	13501265356	传真	010-62519453	通信地址	中国人民大学信息学院		
研究	中文	Web 数据集成，移动数据管理， Native XML 数据库					

方向	英文	Web Data Integration, Mobile Data Management, Native XML Database
基金	中文	国家自然科学基金面上项目，国家 863 项目
资助	英文	Natural Science Foundation of China, National 863 Projects