Towards an adaptation of semi-structured document querying

Corinne Zayani, André Péninou, Marie-Françoise Canut, Florence Sèdes

Towards an adaptation of semi-structured document querying

Corinne Amel Zayani[1,2], André Péninou[1,2], Marie-Françoise Canut[1,2] , and Florence Sèdes[1,2]

1 IRIT, 118 route de Narbonne, 31062 Toulouse cedex 4, France
2 LGC, 129 A, avenue de Rangueil B.P 67701, 31077 Toulouse cedex 4, France
{zayani, peninou, canut, sedes}@irit.fr

Abstract. *In our research work, we consider that access to semi-structured documents is carried out by a data-oriented query. With different users and a same query, the returned results are always the same although users' characteristics (interests, preferences, etc.) may be different. In order to solve this problem and to offer a personalized access to semi-structured documents, our objective is to improve this type of query in order to adapt the result to each user according to his characteristics. On the one hand, we suggest to reorder the results according to user's interests. On the other hand, we also suggest to establish user's interests implicitly from his queries.*

Keywords: User profile, semi-structured documents, adaptation.

1 Introduction

In order to adapt the content of numeric document, different content adaptation techniques have been defined for different adaptive hypermedia systems such as MetaDoc [1], Plan and User Sensitive Help (PUSH) [2], Hypadapter [3], Personal reader [4]. These techniques are based on rules conceived a priori according to the particular domain.

In our research works, we use semi-structured documents which are stored in centralized documentary repository. These semi-structured documents as well as the returned results by the queries are represented in a logical structure in XML. This structure comprises documentary units (elements of the XML structure). The semi-structured documents can belong to any domain, in this case when different users query this type of documents with a same query, the returned results are always the same although users' characteristics (interests, preferences, etc.) may be different.

On this fact, we aim to automate the content adaptation in generic case. In this paper, we present an algorithm in order to implement another content adaptation technique that is already published. This algorithm allows to reordering the documentary units.

In the suggested approach, in order to reduce the cognitive overload [5] (selection and automatic ordering of the results), the objective is to enrich the query with the user interests determined implicitly [6]. In this paper, we present our research works

related to the enrichment of queries in the XQuery language with the user profile in order to adapt the returned results.

The paper is structures as follows. Section 2 presents a general idea of our research works. Section 3 describes the suggested user profile which plays an important role in the process of reordering the documentary units. Section 4 describes the enrichment algorithm of user query. We illustrate in section 5 our research works by an example concerning cottage renting information system in the "Midi-Pyrénées" on the south of France which exists in a platform called PRETI[1].

2 Our proposal

Our context of research works concerns the adaptation of results to the user when he submits queries to semi-structured documents. We work on existing documents which are stored in centralized documentary repository. These semi-structured documents as well as the returned results by the queries are represented in a logical structure in XML. This structure comprises documentary units.

The problem tackled by our work is that documents may not be a priori for adaptation. They are existing semi-structured documents in repository not designed nor built for an adaptation. In such case, different users submitting the same query will get the same results and the same presentation. Because users may have different characteristics, the idea is to present differently the contents for each user when they submit the same query. The objective is to automate this adaptation in order to reduce the cognitive overload of the user when he queries semi-structured documents. We propose an adaptation technique that consists to order the documentary units to be presented to the user according to his own characteristics.

Generally, individual characteristics of users are modeled in user profiles [5], [7]. The characteristics correspond to several information relating to each individual, such as personal information (name, age, etc), interests, preferences, etc. Our objective is to define the interests of each user implicitly (i.e. to determine automatically user interests).

In order to acquire user's interests, we analyze the queries made by the user. In queries, conditions are asserted over elements of semi-structured documents. We consider that conditions may define some user's interest, at a given moment, over the document collection. So, conditions of a query help us to determine user's interests: conditions on elements may become a user's interest. We consider that a condition in a query at a given moment denotes a user's interest. For example, the condition of a query "city=Narbonne" at a given moment denotes the user's interest for that condition. In the user profile, an interest is represented as a condition of a query (an XQuery expression for a condition).

Having such interests, we propose to enrich the user query with some user's interests in order to order the documentary units returned by the query in order to adapt the result to the user.

1 www.irit.fr/PRETI

To carry out the exploitation of the user's interests, the enrichment of the query and the update of the profile, we have defined an algorithm [6]. In this paper, we present an improvement of this previously proposed algorithm.

3 User profile

The user profile which we proposed comprises two characteristics (see figure 1.a):

- permanent characteristics which are introduced by the user and which remain fixed in time, such as name, first name, etc.
- changing characteristics which evolve over time. This type of characteristic introduces the user's interests and user's preferences that are determined implicitly without the user intervention. In this paper, we are interested in the user's interests described in XML Schema[2] in the figure1.b.

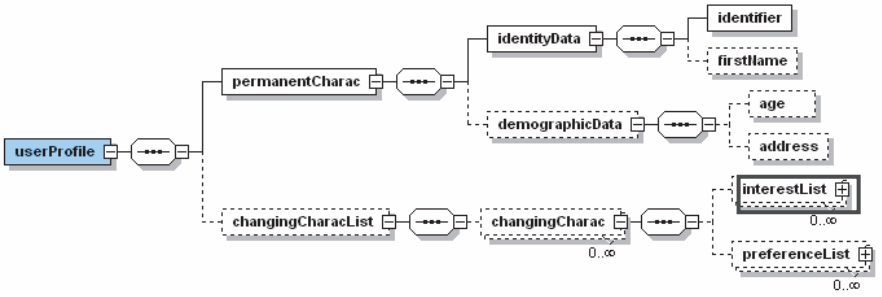

Fig. 1.a. The user profile in XML Schema

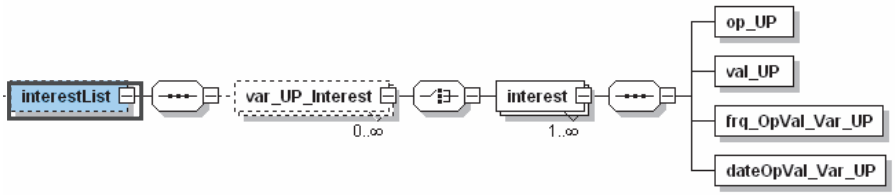

Fig. 1.b. The user's interests in XML Schema

We present below an example of a user profile which represents the instantiation of the XML schema for the PRETI platform (see section 5 for more details). For example, the user is interested in the distance to the sea-distance (var_UP_Interest = sea-distance) which is < (op_UP = "<") to 2 (val_UP = 2) with a frequency of 3 (frq_opVal_Var_UP = 3).

[2] http://www.w3.org/TR/xmlschema-0/

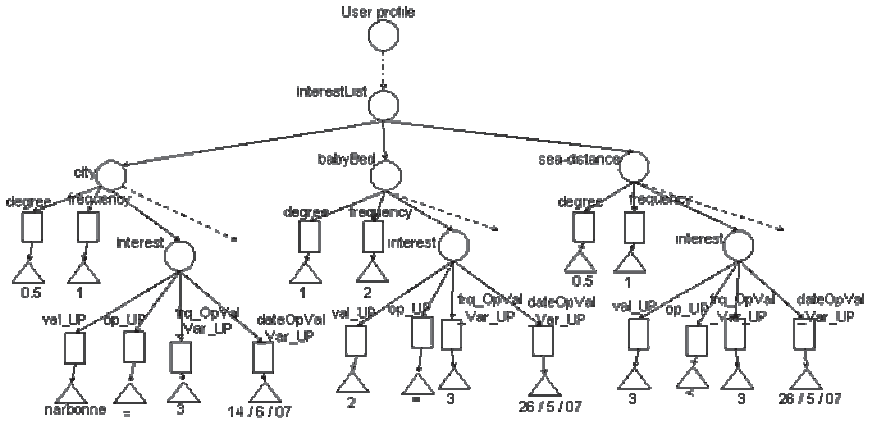

Fig. 2. An example of user's interests

The figure 1 shows the elements that appear in changing characteristics of the user profile. Each element of interest variable "var_UP_Interest" (for example "city", "babyBed" in the figure 2) is associated with an attribute "frequency" that defines "the number of times the interest is used" in the condition part of queries. This attribute "frequency" is incremented each time the element "var_UP_Interest" is used in a query. For example, in the state of the profile of the figure 2, the condition of a new query: "sea-distance<2" will increment the frequency attribute of the "var_UP_Interest" (i.e. sea-distance). According to these frequencies, we proposed to calculate the user's interest degree by the following equation:

$$\text{degree\_interestVal}_i = \frac{\text{frequency\_interestVal}_i}{\sum_{j=1}^{j=n} \text{frequency\_interestVal}_j} \quad (1)$$

In order to standardize the values of interest degree in the interval [0..1], we proposed the following equation:

$$\text{degree\_interestVal}_i\text{\_final} = \frac{\text{degree\_interestVal}_i}{\max_{j=1}^{n} \text{degree\_interestVal}_j} \quad (2)$$

In the equations 1 and 2, "n" represents the total numbers of element "var_UP_Interest" that exist in the user profile.

4 Algorithm for query enrichment

4.1 Principles of the algorithm

For adaptation purpose, the objective of our research work consists in the adaptation of the results during semi-structured documents querying process. For that, we suggest to order (or sort) the documentary units of the results in priority according to user's interests. To carry out this objective, we suggest to enrich the initial query, that is the user's query, with user's interests. Our hypothesis consists in calculating the same returned results in content and length as if no enrichment was made; that is the set returned by the initial query. Only the ordering of the results is modified before presentation to the user. The process should be implemented only after the query has been evaluated by the document repository. However, a query in Xquery language may only return some documentary units of the documents, typically XML elements, meanwhile our process could need to access other elements, typically the entire document. For this reason, we implement an enrichment of user's query which goal is only to order the result, not to change the result in itself. This allows to access any part of the documents when necessary.

We have first proposed an algorithm for query enrichment based on two generic functions in [8]. It takes into account the user's interests correctly if only one interest is given. The two functions suggested have been described in XQuery language [8]. So, we propose to improve this first version in order to take into account "n" interests. As described in section 3, a user's interest is represented as an XQuery condition; for example city="Narbonne", sea-distance<2, …

For taking into account only one user's interest in a given query, the solution consists in ordering results in the following way: 1) those results relevant to the query and to the user's interest, 2) those relevant to the query and *not* to the user's interest. For example, if the user's interest is sea-distance<2, the ordering of the results for a query "Q" (with no conditions on sea-distance) will consist in giving: firstly the cottage having a "sea-distance<2" ("Q and sea-distance<2"), and then the other ones (that is "Q and sea-distance>= 2", or written differently "Q and not(sea-distance<2)"). Without this ordering, the same results would be presented to the user but in some "random" order, regardless of sea-distance value.

When multiple interests are to be taken into account, the ordering problem is to present to the user the same result set in content and length as would give the initial query. Any given result may match no user's interests at all, only ne of them, 2 of them, n of them, all of them. The idea is then to "rank" each result according to its matching to interests in order to get its range in the result presented to the user. That will be a relative range, that is subsets of results matching "n" interests. Finally, results matching all interests will be presented first, then those matching "less" interests, and so on.

For that purpose, we suggest to combine the set of interests and enrich the user query with those combinations. The enriched query will return the results in the correct order. Each interest admits a boolean value when evaluated for a given document. For example, the interest city="Narbonne" will be evaluated to true or false for each cottage renting document (see example in section 5). So, combining

interests is a boolean combination of interests and of their negations. For example, combining interests sea-distance<2 and city="Narbonne" leads to 4 expressions: i) sea-distance<2 AND city="Narbonne", ii) sea-distance<2 AND NOT city="Narbonne", iii) NOT sea-distance<2 AND city="Narbonne", iv) NOT sea-distance<2 AND NOT city="Narbonne".

The order of the boolean combination is important since right elements are negated before left ones. So, we use frequencies of interests in the user profile in order to get the right combination that leads to satisfy the "more important" interests first (those having the highest degrees or frequencies), then the lower one, and so on. In the example above, the interest sea-distance<2 is considered as more important than city="Narbonne". We suppose that the user will then prefer cottages that satisfy a sea-distance<2 but not located in "Narbonne" than those located in "Narbonne" but not having a sea-distance<2. Obviously we suppose that he first prefer cottages having a sea-distance<2 and located in "Narbonne", and lastly those corresponding to no interests. The order or stages of the combinations traduces these interests.

Another issue in this approach is to select the user's interests to take into account for enriching a given query. Using all interests of the user profile regardless of the query may lead to inconsistent conditions and "out of sense" query. So, we try a heuristic solution that is to take into account interests: i) not used in the query condition, ii) having some "sense" in the context of the query. To define a sense relationship between a user's interest and query conditions, distances measurements inside the document structure between query conditions and user's interests are used. Heuristic threshold can decide if interests are semantically closed to query conditions, and therefore can be used to adaptation.

Supposing that "n" is the number of the user's interests that will be extracted from the user profile (and added to the query), the interpretation of the algorithm leads to 2^n stages. Each stage leads to a new "partial" query, that we can consider as a part of the enriched query to be evaluated. At each stage, two parts are inserted in the query being built: a static part and an evolutionary part. The static part is the conditions of the initial query. The evolutionary part is made up of the users's interests. Its evolution depends on boolean combinations of interests, that is the change of operators (e.g. =, >, <, etc.) by their negation as described above.

In the first stage, we keep the initial operators which are extracted from the profile. That corresponds to find all documents existing in the collection that both match the query and all user's interests. Afterwards, in each stage, a boolean combination of interests is used. The combination of the interests is made in the decreasing order of frequency existing in the user profile. That corresponds to find documents from the collection that match the query and match the less and less the user's interests: these documents first match all user's interests but not the lower frequent, and so on. In the last stage, all operators for user's interests are the negation in comparison with the first stage. That corresponds to find documents that match the query but that do not match user's interests.

Finally, in order to adapt the results to the user, the system should evaluate all of these queries, in the order they have been generated. That ensures a correct ordering of the documentary units according to user's interests.

4.2 Algorithm implementation

The proposed algorithm is divided into three parts:
1) The first part consists in the selection of user's interests to use for query enrichment. This part extracts user's interests from the user profile. A user's interest is selected if: 1) it is not used in the query, 2) it is similar to at least one condition of the query, that is, "it has some common sense and link in the context of the query". Similarity measures are presented hereafter.
2) The second part of the algorithm consists, on the one hand, in sorting the selected user's interests in the decreasing order of frequencies (degrees) and in combining them into a boolean combination as described previously. On the other hand, it generates the enriched query.

To set up these combinations, we use a matrix M [Line, Column] which has a number of columns (Column) equals to the user's interest number (n) and has a number of lines (Line) equals to 2^n. This matrix is filled by values 1 to indicate a user's interest such as stated in the user profile and 0 to indicate the negation of the user's interest. To produce this matrix, we use a function based on classical binary combinations of "n" elements.

We have defined a function called "enrichQuery" (combination_Matrix, initialQuery, interestList). This function takes as parameter the matrix "combination_Matrix", the condition part of the user's query "initialQuery", and user's interests "interestList". This function returns a list named "list_EnrQuery" which contains all the possible combinations of the user interests with the condition part of the user's query.

We have also defined an algorithm using the list defined above ("list_EnrQuery") and able to generate a new query which combines all this sub-queries in a single one to produce the expected results. We use as many let expressions of XQuery language as necessary sub-queries, each one evaluated in the correct order. This generated query is the enriched query evaluated by the documentary repository.
3) The third part of the algorithm makes it possible to update the user profile from the query. This update consists in adding new interests (conditions of the initial query) to the profile or increasing frequencies for existing ones. The execution of the enriched query is carried out via an execution processor SAXON[3]. The JAVA implementation of this algorithm generates only one query in XQuery language submitted to the execution processor.

The first and the third parts of the algorithm are based on the similarity measurements in order to compute the distance between the documentary units and the interests that are in the user profile. The similarity measurements used in our algorithm are inspired by the work of Yi, Huang and Chan [9].

The similarity measurement between the user profile interests and the query conditions is determined on the basis of their elements properties. These elements properties (XML elements properties) are found in the documents themselves. In this paper, we are interested in the similarity measurement for the names of elements taken from [8], that is determined by the name matching of the two elements "a" and "b" as follows:

3 http://saxon.sourceforge.net/

$$S_{name}(a,b) = \frac{N_{n(a)\cap n(b)}}{N_{n(a)\cap n(b)} + \alpha(a,b)N_{n(a)/n(b)} + (1-\alpha(a,b))N_{n(b)/n(a)}}$$

Where $N_{n(a)\cap n(b)} = |name(a) \cap name(b)|$ returns 1 if the two names have a common string, else 0. $N_{n(a)/n(b)} = |name(a)/name(b)|$ returns 1 if "a" has a string name difference with "b" string name, else 0. For example, the two names "depart" and "department" have a common string "depart" and the set difference of two names is the string "ment". Therefore $N_{department \cap depart} = 1$, $N_{department/depart} = 1$, $N_{depart/department} = 0$. Supposed that weight α=0.5 (the relative importance of $N_{n(a)/n(b)}$ and $N_{n(b)/n(a)}$ is the same), the similarity of the two names is equal to $S_{name}(department, depart) = 0.66$.
This similarity measurement is used in the first part of the enrichment algorithm, in order to verify if a user interest is used or not in the query conditions. It is also used in the third part of the algorithm to decide for new or existing user's interests.
In our context, we have defined the similarity of parents for "a" and "b" as follows:

$$S_{parent}(a,b) = \begin{cases} 1 \ \ if \ \ S_{name}(parent(a), parent(b)) > 0 \\ 0 \ \ if \ \ S_{name}(parent(a), parent(b)) = 0 \end{cases}$$

This similarity of parents of two elements is used in the first part of the algorithm. It allows to select a user's interest if it has a similarity of parent with at least one condition of the query greater than a threshold. For the moment, this threshold is determined manually and depends on the documents structure.

5 Application

We present, on figure 3, the document structure of the PRETI application[4] in XML schema.

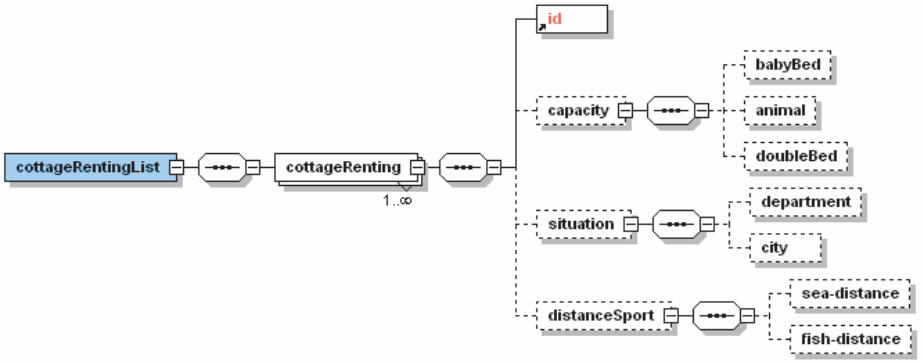

Fig. 3. Example of XML documents of cottage renting

4 www.irit.fr/PRETI

We suppose that a user wants to find rented cottages in "aude" department accepting animals (department = "aude" and animal = "yes"). This query is written in XQuery language as follows:

```
for $a in doc ("cottageRenting.xml")
  where      department="aude"      and      animal="yes"
return $a
```

The result returned by this query contains ordered elements according to the structure of XML document of cottage renting (figure 4).
In figure 4, we suppose that a given user profile is presented on the figure 2 (see section 3) and that the current conditions expressions of query are presented on the figure 4.

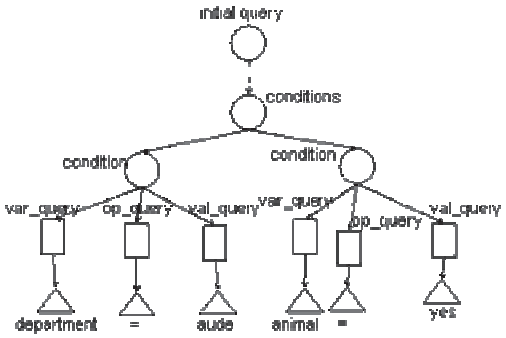

Fig. 4. Example of query conditions

We have applied the algorithm proposed in the section 4. At the beginning, the algorithm determines that:

- the "city" element with value "narbonne" that exits in the user profile has a parent similarity with "department" that exits in the query.
- the "babybed" element that exits in the user profile has a parent similarity with "animal" that exits in the query.

Once these elements are given, the algorithm sorts them according to the decreasing order of their frequencies. In the example, that's give babyBed/city.

On the other hand, the algorithm enriches the query by generating 4 queries (2^2 where the power 2 represents the number of used user's interests). The generated stages in the described example are:

- First stage
  department = "aude" and animal = "yes" and babybed = 1 and city = "narbonne"
- Second stage
  department = "aude" and animal = "yes" and babybed = 1 and city != "narbonne"
- Third stage
  department = "aude" and animal = "yes" and babybed != 1 and city = "narbonne"
- Fourth stage
  department = "aude" and animal = "yes" and babybed != 1 and city != "narbonne"

So the enrichment of the user query with his interests enables to offer to the user a reordering of documentary units of cottage renting.

We have evaluated the results returned by the initial query and by the enriched query with some users of our laboratory. This evaluation is carried out on the corpus of the database PRETI which includes a collection of 700 documents of cottage renting. The result of the evaluation is presented for the first ten documentary units in the following figure:

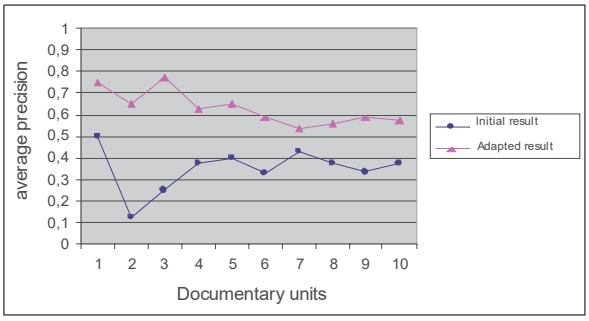

Fig. 5. Evaluation curves with user's judgments

The result of enriched query is presented by the curve of adapted result. The result of initial query is presented by the curve of initial result. The evaluation of these both results in figure 5 shows that the enriched query answers better to user's interests than initial query.

6 Conclusion and prospects

In this paper, we have proposed an improvement of an algorithm for adaptation of results of queries when querying corpus of semi-structured documents. The overall goal is to reduce the problem of cognitive overload. This algorithm consists in selecting the user's interests that are to be taken into account. It also consists in enriching the initial user's query to reorder the results according to the selected interests. A first evaluation with users shows that the first results presented to the user better fits their needs for the enriched query than for the initial users' one. An originality of this work is that this adaptation may be applied to many documents belonging to different domains, especially existing ones where adaptation was not planned when documents were created.

We are now implementing the proposed algorithm in the PRETI platform. This implementation will then enable us to carry out more advanced evaluations. These evaluations must lead to validate our approach from user's point of view. One issue is the number of the stages of our solution in the form of 2^n, n being the number of selected user's interests. One solution could be to modify the enriched query and use the "order by" expressions of XQuery language. Another one could be to give the user a list of subsets of results corresponding less and less to his interests. In that case,

only n subsets (n queries) should be evaluated. This possible solutions need to be more deeply studied. Another issue is the selection of user's interests that are to be taken into account. For that point, we investigate to study other methods from the literature.

Bibliography

1. B. B. Craig D. and A. O. Metadoc : An adaptive hypertext reading system. User Model. User-Adapt. Interact., 4(1) :1–19, 1994.
2. H.-D. B. Hubertus Hohl and R. Gunzenhauser. Hypadapter: An adaptive hypertext system for exploratory learning and programming. User Model. User-Adapt. Interact., 6(2-3), 1996.
3. H. K. Evaluating the utility and usability of an adaptive hypermedia system. In Knowledge Based Systems, 10(5), 1998.
4. P. Dolog, N. Henze, W. Nejdl, and M. Sintek. The personal reader: Personalizing and enriching learning resources using semantic web technologies. In AH, pages 85–94, 2004.
5. P. Brusilovsky. Adaptive hypermedia. User Model. User-Adapt. Interact., 11(1-2), pp. 87–110, 2001.
6. C. Zayani, A. Peninou, M.F. Canut, and F. Sèdes. An adaptation approach: query enrichment by user profile. In Signal-Image Technology and Internet–Based Systems (SITIS), pages 24–35, IEEE/ACM, 2006.
7. Kobsa, A., and Koenemann, J., and Pohl, W. "Personalized Hypermedia Presentation Techniques for Improving Online Customer Relationships," in The Knowledge Engineering Review 16(2), 111-155, 2001.
8. C. Zayani, F. Sèdes. An approach to query-based adaptation of semi-structured documents. Dans : International Conference on Enterprise Information Systems (ICEIS 2006), Paphos/Cyprus, INSTICC Press, p. 156-161, mai 2006.
9. S. Yi, B. Huang, and W. T. Chan. Xml application schema matching using similarity measure and relaxation labeling. Inf. Sci., 169(1-2):27–46, 2005.