Domain Ontology and Galois Lattice Structure for Query Refinement

Brigitte Safar and Hassen Kefi
University of Paris Sud-CNRS (LRI), INRIA (Futurs),
LRI, Building 490, 91405 Orsay Cedex, France
{safar, kefi}@lri.fr

Abstract

We present a system for interactively querying a topical resources repository and for navigating the set of responses obtained from the repository. Our system makes use of a domain ontology for dynamically constructing a Gallois lattice. Nodes in this structure are resources clusters which are interactively examined by the user during the construction process.

1. Introduction

Topical repositories are sets of resources (documents, data sources or services) related to a particular domain and described by a set of terms belonging to a domain ontology. In the context of the semantic web, such repositories are being developed for some years in different application domains like tourism, medicine, biology, etc. Queries for accessing the repository information usualy consists of sets of keywords as in boolean search engines. However, terms in the query are usualy restricted to a subvocabulary coming e.g. from the domain ontology. As for classical Information Retrieval engines, users of topical repositories are often swamped by a mass of answers. Classical ranking or clustering schemes may be used in order to help users finding the most relevant resources. However specific solutions which take benefit from the specificities of topical repositories can be more efficient and convenient to the user.

This is the problem we adress in this paper. We present OntoRefiner, a system that creates clusters from returned answers and allows the user to refine these clusters interactively. This system helps focusing on the more relevant returned resources and facilitates query refinement. The topic of the repository is described by a domain ontology which is composed of a hierarchy of terms or keywords. We consider that the description of each resource has already been performed automatically or manually, using a set of terms belonging to the domain ontology and we consider that users ask queries using terms from the domain ontology too. Resources returned by the search engine are the ones whose description maps all the terms of the query. The inputs of the clustering algorithm are the domain ontology and descriptions of search engine results. From these inputs, OntoRefiner creates general clusters and identifies in each cluster some possible lines of specialization which consists in selected terms from the ontology. The system allows the user to choose the most relevant cluster and specialization lines for refining the query, and iterates the process until the user is satisfied.

The paper is organized as follows. Section 2 introduces the domain ontology and its use for bringing out different points of view in a resource description. Section 3 presents the Galois Lattice clustering structure upon which our work is based and its use for query refinement. We then show, in section 4, how our algorithm makes use of the ontology to avoid constructing the whole Galois Lattice without losing the potential of this type of structure. We conclude with a short discussion in section 5.

2. Ontology and description of resources

We consider a domain ontology composed by a hierarchy of terms (denoted $\mathcal{D}_h$) and expressed by a set of rules of the form: $C_1 \rightarrow C_2$, where C_1 and C_2 are terms. Throughout the paper we will use for an illustration a toy example from the tourism products domain. For such an application, possible ontology terms could be AccomodationPlace, Localization, Equipment, Activities. Figure 1 shows a fragment of the hierarchy describing term Localization according to different points of view like physical and geographical characteristics. The following two definitions establish the notion of subsumer of a term.

Definition 1 *A term c' is a* **direct subsumer** *of a term c if the rule $c \rightarrow c'$ is in $\mathcal{D}_h$. c is a direct specialization of c'.*
A term can have many direct subsumers and specializations.

Definition 2 *c' is a* **subsumer** *of a term c if c' is a direct subsumer of c or if there exists c" such that c' is a direct subsumer of c" and c" is a subsumer of c.*

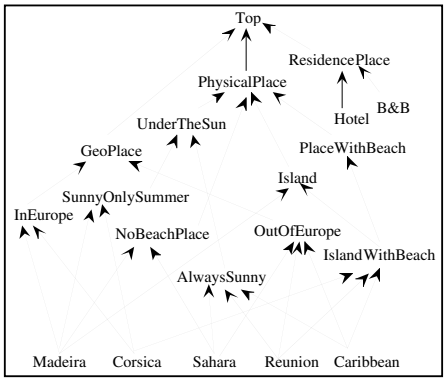

Figure 1. An example of a hierarchy

Example 1 The direct subsumers of Reunion Island are AlwaysSunny, OutOffEurope and IslandWithBeach while other subsumers are PlaceWithBeach, Island, etc. The direct specializations of UnderTheSun are SunnyOnlySummer and AlwaysSunny while other specializations are Corsica, Sahara, etc.

Resources are supposed to be indexed with the most specialized terms of the domain ontology. Using the hierarchy of figure 1, a resource focusing on a hotel situated on Reunion Island will have as description the set of terms {Hotel, Reunion}.

Before clustering, the description of each resource returned by the engine is enriched, according to the ontology, with all subsumers of the resource index terms. This is called **saturation** and is defined as follows:

Definition 3 *Sat(C) is the* **saturated** *of the set of terms C, if for each term c in C, all subsumers of c are in Sat(C).*

Example 2 Using the hierarchy in figure 1, for C = {Hotel, Reunion}, Sat(C) = {Hotel, ResidencePlace, Reunion, AlwaysSunny, UnderTheSun, OutOfEurope, IslandWithBeach, PlaceWithBeach, Island, GeoPlace, PhysicalPlace}.

Conversely we define **desaturation** of a set of terms C as the process of removing from this set all subsumers whose specializations are in C.

Definition 4 *DeSat(C) is the* **desaturated** *of the set of terms C, if for each term c in DeSat(C), c is in C and there not exists c" in C such that c" is a specialization of c.*

The purpose of the saturation step is to make appear similar terms between descriptions that do not share any terms in their initial description and describe related resources. For example, the intersection of the descriptions for {Madere} and {Réunion} is empty, the saturated descriptions do share as common subsumers {GeoPlace, UnderTheSun, Island}. Saturation also allows to analyze similarities between resources at a more abstract level than the initial description (viz. using the most general terms of the description) and along different points of view inherent to the ontology. Let us remark that for database applications, ontologies do focus on specific points of view so that explicit characteristics are relevant only to a particular domain. In the tourism products domain for example a Place will be characterized by its facilities for nautic, swimming or other leisure activities, whereas in another settings, historical, political or geological perspectives would be expressed.

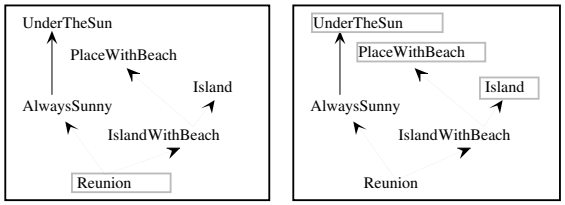

Figure 2. fig.2a : Desat(C), fig.2b : mgt(C)

When looking for shared terms, we work on non desaturated resource descriptions and we focus on the most general terms which are more prone to be shared between descriptions. The most general terms for a set of terms are defined below.

Definition 5 *mgt(C) is the set of the* **most general terms** *of a set of terms C if for each term t in mgt(C), t is in C and there not exists t' in C such that t' is a subsumer of t.*

Example 3 In figure 2, using the hierarchy in figure 1, for C = {Reunion, AlwaysSunny, UnderTheSun, IslandWithBeach, PlaceWithBeach, Island}, DeSat(C) = {Reunion} and mgt(C) = {UnderTheSun, PlaceWithBeach, Island}. The terms of DeSat(C) and mgt(C) respectively appear in boxes in fig. 2a and fig. 2b.

3. Clustering method

Our clustering method is based on Galois lattices [11], which have already been used for supporting browsing retrieval by [5, 3]. Given two finite sets D (set of documents or more generally resources) and T (set of terms), and a binary relation $\mathcal{R}$ between these two sets, the Galois lattice is a particular set of clusters, in which each cluster is a pair, composed of a subset of documents $D' \subseteq D$, called **extent**, and a subset of terms $T' \subseteq T$, called **intent**. Each pair (D', T') must be a complete pair with respect to $\mathcal{R}$, meaning that T' must only contain the terms shared by all the documents in D', and the documents in D' must be precisely those sharing all the terms in T'.

$T' = f_1(D')$ where $f_1(D') = \{t' \in T' \mid \forall d' \in D', d'\mathcal{R}t'\}$

$D' = f_2(T')$ where $f_2(T') = \{d' \in D' \mid \forall t' \in T', d'\mathcal{R}t'\}$

Example 4 Figure 3 shows an example containing 5 documents described by 7 terms, {a,b,c, d, e, f, g}. Document 1 is indexed by terms {a, c, f}. Term g appears in documents 2 and 3.
$f_1(\{1,4\})$ = {c, f} and $f_2(\{c, f\})$ = {1,4, 5}. ({1,4},{c, f}) is not a complete pair (because $f_2(\{c, f\}) \neq \{1,4\}$) whereas the pair ({2, 3},{a, g}) is complete since $f_1(\{2, 3\})$ = {a, g} and $f_2(\{a, g\})$ = {2, 3}.

Given two pairs $C_1 = (D_1, T_1)$ and $C_2 = (D_2, T_2)$, a partial ordering relation ($<$) is defined on this set of pairs by: $C_1 < C_2 \Leftrightarrow T_1 \subseteq T_2$ or $C_1 < C_2 \Leftrightarrow D_2 \subseteq D_1$. The partial order is used to generate the lattice graph in the following way: there exists an edge (C_1, C_2) if $C_1 < C_2$ and there is no other element C_3 in the lattice such that $C_1 < C_3 <$

	a	b	c	d	e	f	g
1	•		•			•	
2	•		•				•
3	•			•			•
4		•	•			•	
5		•	•		•	•	

Figure 3. An example of relation

C_2. C_1 is called **parent** of C_2 and C_2 **child** of C_1. The graph reveals the generalization/specialization relationship between the pairs corresponding to the subset relationship between the D or T parts.

Example 5 In figure 4a showing the lattice graph corresponding to the relation of figure 3, we can see that the pair ({a},{1,2,3}) is more general than the pair ({a, g},{2,3}).

The Galois lattice representation exhibits a lot of redundancy. For a pair $C = (D', T')$, D' will be present in every ancestor of C and T' will appear in all its descendants. Godin et al. showed in [6] that this redundancy may be eliminated without losing any information if the graph is maintained. For a pair $C = (D', T')$, let T'' be the non redundant element in T', $T'' = \{t' \in T' | t' \in f_1(D')$ and there is no other pair $C' = (X, Y) < C$ such that $t' \in Y\}$. Godin defines the $T'-$ *inheritance concept lattice* ($T' - ICL$) using the set of pairs (D', T''), and shows that for a given pair C, the corresponding values of T' can be computed by taking the union of the T'' sets of the ancestors of C including C itself. The features of T' not in T'' are inherited from their ancestors. In this work, we will use the set of pairs (D', T'').

Example 6 Comparing fig. 4.a and 4.b we can see that the intent of the pair ({a, c},{1,2}) of fig. 4.a, appears empty in fig. 4.b. The intent of this pair can be computed by taking the union of the T'' sets of its ancestors the pairs ({a},{1,2,3}), ({c},{1,2,4,5}) and ($\emptyset$, {1,2,3,4,5}).

This clustering structure is especially relevant in the context of information retrieval because one cluster is directly described in intension by T' which is a simple set of terms easily readable and understandable by the user. Furthermore this lattice structure allows overlap: a document may appear in different clusters reflecting different points of view. This is an advantage compared to hierarchical classification for example.

Godin [5] and Carpineto [3] have used this type of structure to index a document collection manually annoted by a set of terms. Once the lattice characterizing the whole database is built, user's queries are compared with the intents of the lattice pairs. Indeed, for each pair, its intent can be seen as a conjunctive query and its extent as the documents answering this query. In this setting, answering a user's query consists in looking for the more general pair whose intent contains all the terms of the considered query. Once this pair is found [1] the user is allowed to explore its direct neighbors in the lattice. The children of the pair correspond to a minimal refinement of the query and the parents to a minimal enlargement.

[1] If the query has no answer in the set of documents, the searched pair does not exist. The problem of cooperative answering is not studied here and the reader interested in this problem is refered to [3], [2] or [9].

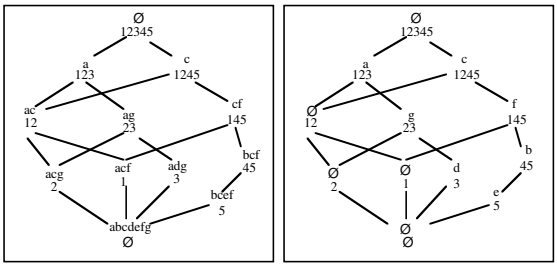

Figure 4. Galois lattice graphs

[5] or [3] build their lattice structures from a static database. Our context is different since the response resources set is generated dynamically for a user's query: resources clustering is ephemeral and the construction of the full Galois lattice for each set of responses would be too costly. In order to avoid this, we first construct all nodes from the first level of the lattice, and identify for each node its potential axes of refinement. In a second step, the user is asked to select a relevant node (denoted **active node** in the following) and an axis for this node. A new set of refined nodes is then built from the active node: they are the children of the node which share the selected axis characteristic. This process is then iterated. Doing this way allows to explore only a subpart of the original Gallois lattice under the user guidance. The user is allowed during his exploration to backtrack to a previous choice so as to explore alternative nodes.

4. Algorithm

Given a query, the input to our algorithm are the set D of resources whose description maps all the terms of the query and for each d in D, a set of terms called **current description** $Cdesc(d)$ initialized with the saturated of the term set describing the resource. The output is a refined query composed of a set of descriptors.

Example 7 Let us suppose that the user query is Q = ResidencePlace. Let us suppose too (for a not too big example) that the search engine finds only four resources mapping this query and that the sets of terms describing these resources are:
- d_1: {Hotel, Reunion}, - d_3: {B&B, Sahara},
- d_2: {Hotel, Carribean}, - d_4: {B&B, Corsica}.

The current description of the resource d_1 is then initialized as follows: Cdesc(d_1) = {Hotel, ResidencePlace, Reunion, AlwaysSunny, UnderTheSun, OutOfEurope, IslandWithBeach, PlaceWithBeach, Island, GeoPlace, PhysicalPlace}.

The algorithm realizes 3 tasks at each iteration: i) the selection and cleaning of an active node, ii) the construction of refined nodes as its children, iii) the identification of potential axes of refinement for each refined node, then the hand is returned to the user who decides to iterate or stop the process.

4.1. Selection and cleaning of an active node

When an active node has been selected by a user, the descriptors shared by all the resources in D of this node are identified as its intent. Since these resources are returned by the search engine, all their current descriptions share, at least, the descriptors mapping the terms of the query.

This set of shared descriptors is removed from the current description of any resource in the active node, the set is kept as intent for the active node. Doing so we eliminate the redundant elements in T'as in the Godin's $T' - ICL$.

Example 8 Current descriptions of the four resources of the example shared the following descriptors: {ResidencePlace, UnderTheSun, GeoPlace, PhysicalPlace}. The active node is then (1234, {ResidencePlace, UnderTheSun, GeoPlace, PhysicalPlace}).

4.2. On line construction of the lattice nodes

This step clusters the documents from the active node and creates children nodes in the lattice each defined by its Extent and Intent. Its originality is that using the specialization-generalization hierarchy of terms from the ontology allows for a more efficient algorithm than the traditional algorithm.

We consider the current description of each returned resource of the active node whose shared descriptors identified in the previous step have been removed. We start by identifying the set of the **most general terms** of the level, Mgt, which is the union of the most general terms appearing in the current description of each resource d_i of the active node, $Mgt = \bigcup_{i \in [1 \dots n]} mgt(Cdesc(d_i))$.

Example 9 For the current description of d_1, Cdesc(d_1) = {Hotel, Reunion, AlwaysSunny, OutOfEurope, IslandWithBeach, PlaceWithBeach, Island}, mgt(Cdesc(d_1)) = {Hotel, AlwaysSunny, OutOfEurope, PlaceWithBeach, Island}.

For the 4 resources, Mgt = {Hotel, B&B, AlwaysSunny, SunnyOnlySummer, OutOfEurope, InEurope, PlaceWithBeach, NoBeachPlace, Island}.

Then, for each term t in Mgt, the algorithm constructs the complete pair (D', T') centered around t, i.e., such that the resources in D' are the ones whose current description contains t and such that T' are the terms shared by all the current descriptions of the resources in D'. We only keep the most general complete pairs, the ones whose extent is not included in the extent of an other complete pair built. Each most general complete pair built is a cluster whose intent will be displayed to the user, providing then an informative way of summarizing its contents and helping the user to choose a further cluster to explore.

Example 10 The complete pair centered around AlwaysSunny is: (123, {AlwaysSunny, OutOfEurope}).
It is more general than the one centered around Hotel, wich is: (12, {Hotel, AlwaysSunny, OutOfEurope, PlaceWithBeach}). Only the first one is kept.

Let us show that the general complete pairs obtained here are the same as the ones calculated by a classical descendant Galois Lattice algorithm. The classical descendant algorithm builds the complete pairs by calculating the closure of each term which belongs to the descriptions of the resources, whereas we only calculate the closure of each term $t \in Mgt$. Let us suppose there exists a term t such that $t \notin Mgt$ for which there exists a complete pair (X, Y) centered around t and built by a classical algorithm. Since $t \notin Mgt$, in each description where t appears there exists a subsumeer t' of t. All descriptions of resources X containing t contain t' too which means $t' \in Y$ and the complete pair (X, Y) has been built by our algorithm too.

Example 11 3 clusters are finally contructed in this step:
- (34, {B&B})
- (123, {AlwaysSunny(y), OutOfEurope})
- (124, {PlaceWithBeach, Island, IslandWithBeach})

4.3. Identification of axes of refinement

During the interactive cluster construction, as an additional help to the user, we will propose him some axes of refinement for each cluster. They correspond to terms among which he will be allowed to choose the one he wants to appear in the futur query. Each cluster is first cleaned as in step 1: the intent of the cluster is removed from each cluster document description. The most general terms in the new descriptions are then the refinement axes. A document may appear in different brother clusters. After step 2, its description will be the same for all these clusters. After step 3, since the cluster intent has been removed, this description will be specific for each cluster.

Example 12 After step 2, the current description of d_1 was Cdesc(d_1) = {Hotel, Reunion, AlwaysSunny, OutOfEurope, IslandWithBeach, PlaceWithBeach, Island} for the two brother clusters created in step 2 where it appears. After step 3, in the two clusters, the new current descriptions of d_1 are now:
- for the cluster (123, {AlwaysSunny, OutOfEurope}), Cdesc(d_1) = {Hotel, Reunion, IslandWithBeach, PlaceWithBeach, Island}.
- for the cluster (124, {PlaceWithBeach, Island, IslandWithBeach}), Cdesc(d_1) = {Hotel, Reunion, AlwaysSunny, OutOfEurope}).

Each new set Mgt, calculated for each cluster, is displayed to the user as the set of axes which can be chosen to refine the clusters. For each term appearing in each Mgt, the number of documents matching this term are displayed too.

Example 13 For the cluster (123, {AlwaysSunny, OutOfEurope}), mgt(Cdesc(d_1)) = mgt(Cdesc(d_2)) ={Hotel, PlaceWithBeach,Island} and mgt(Cdesc(d_3)) = {B&B, NoBeachPlace}.

Then, the Mgt of this cluster is {Hotel, B&B, PlaceWithBeach, Island, NoBeachPlace} and the displayed set of axes is {Hotel 2, B&B 1, PlaceWithBeach 2, Island 2, NoBeachPlace 1}.

4.4. Interaction with the user

The different clusters and their associated set Mgt are displayed. The user must select a cluster (D', T'), then he can

• choose one axis of refinement, t, among the terms of Mgt associated with the selected cluster. The algorithm identifies the subset of resources d' in D' whose $Cdesc(d')$ contains t, and goes back to the task 1 with this subset.

Example 14 Let us suppose that the user chooses the cluster (123, {AlwaysSunny, OutOfEurope}), and the term PlaceWithBeach as axis of refinement. Among the resources 123, the ones mapping this axis are 1 and 2.The algorithm goes back to step 1 with this subset of resources and their current descriptions.

• or say that the selected cluster is relevant. The refined query is then built by taking the union of the intents of clusters successively chosen by the user, and by computing the desaturated of the set obtained.

Example 15 Let us suppose that the user chooses only cluster (123, {AlwaysSunny, OutOfEurope}).

The refined query is then composed from the union of the following sets: {ResidencePlace, UnderTheSun, GeoPlace, PhysicalPlace} which is the intent of the initial node and {AlwaysSunny, OutOfEurope} which is the intent of the cluster chosen. After desaturation, the resulting refined query is {ResidencePlace, AlwaysSunny, OutOfEurope}.

If the user is not satisfied with the built clusters, he can backtrack to a previous choice and explore alternative nodes. This process allows him to make refinement step by step as he could do in a whole Galois lattice.

5. Related Works

Ephemeral clustering as an help to navigate through returned sets of documents has been explored by different authors [8, 12]. The requirements of ephemeral clustering are defined in [8]: it requires high precision because the user who is not a domain expert is less tolerant to errors, a presentation layer that enables users to effectively browse the clusters, including visualization techniques and automatic annotations of the clusters. In order to increase the precision, Maarek et al. [8] use a Hierarchical Agglomerative Clustering method and define a similarity measure between individual documents based on indexing units which consist of a pair of words that are linked by a lexical affinity. In order to increase the automatic annotations of the clusters and provide concise and accurate descriptions, Zamir and Etzioni [12] use Suffix tree clustering based on phrases shared between the short documents snippets returned by Web search engines. These two works do not accept user feedback and do not allow to help the user reformulate his initial query.

To improve interactive retrieval and take into account user feedback [7] combine ranked lists and clustering: the documents are ordered by their relevance probability and the user is supposed to follow this ordering until the first relevant document is found. This first document is used to initiate the clustering algorithm with the unexamined documents.

To help the user to focus on relevant items and to drive the process of query reformulation, Berenci et al. [1] visualize the retrieval results gathered according to all the possible subsets of query terms that can be generated from the user query.

The works based on Galois lattice structure [5, 3, 4] and presented in section 3 are the ones nearest to ours. Its advantages are twofold. The annotation of the clusters is made automatically: it is the intention of a node which groups the terms shared by all the documents in the node. And this structure allows to explore the results of refinement of a query by exploring the neighbors of a node in the lattice. The principal problem of this clustering method is its cost: it cannot be applied to Web-based retrieval because it is computationnally expensive. In the context of topical repositories where the number of returned resources is limited, and and with our algorithm that compute the Gallois lattice at a low cost compared to classical methods, this clustering method is attractive.

An original aspect of our work is the use of a detailed ontology for query refinement, and moreover, the fact that in this ontology, terms may have many direct subsumers and specializations. The use of this ontology allows to automatically enrich the resource description with many terms that generalize the original resource description and along different points of view. So, the use of these different points of view allows to increase the precision of the clusters built. The proposed algorithm is implemented in Java and is actually experimented in the mediator Picsel[10] .

References

[1] E. Berenci, C. Carpineto and V. Giannini, Improving The Effectiveness Of Web Search Engines Using Selectable Views Of Retrieval Results. *J. of Universal Computer Science*, 4, 737-747, 1998.

[2] A. Bidault, Ch. Froidevaux, B. Safar, Repairing Queries in a Mediator approach. In *Proc. of ECAI'00*, 406-410, Berlin, August 2000.

[3] C. Carpineto and G. Romano, Information Retrieval through hybrid navigation of lattice representations. *International Journal of Human-Computer Studies*, 45, 553-578, 1996.

[4] P. Eklund and R. Cole, Structured Ontology and Information Retrieval for Email Search and Discovery. In *Proc. of ISMIS'02*, LNAI 2366, pp.75-84, Lyon, June 2002.

[5] R. Godin, R. Missaoui and A. April, Experimental comparison of navigation in a Gallois lattice with conventional information retrieval methods. *Int. Journal of Man-Machine Studies*, 38, 747-767, 1998.

[6] R. Godin, R. Missaoui, H. Alaoui, Incremental Concept Formation Algorithms Based on Galois (Concept) Lattices. *Computational Intelligence,*11(2), 246-267, 1995.

[7] Anton Leuski and James Allan, Improving Interactive Retrieval by Combining Ranked Lists and Clustering. In *Proceedings of RIAO'2000*. 665-681, April 2000.

[8] Y. Maarek, R. Fagin, I. Ben-Shaul and D. Pelleg, Ephemeral Document Clustering for Web Applications. IBM Research Report RJ 10186, April 2000.

[9] T. Gaasterland, P. Godfrey, and J. Minker, An overview of Cooperative Answering. *J. of Intelligent Inf. Systems*, vol.1, 123-157, 1992.

[10] M.-Ch. Rousset, A. Bidault, Ch. Froidevaux, H. Gagliardi, F. Goasdoué, C. Reynaud et B. Safar, Construction de médiateurs pour intégrer des sources d'information multiples et hétérogènes : le projet PICSEL, Dans *Revue I3,* Vol 2,(1), pp. 9-59, Septembre 2002.

[11] R. Wille, Concept Lattices and Conceptual Knowledge Systems. *Computers and Mathematics with Applications* 23, 403-515, 1992.

[12] O.Zamir and O. Etzioni, Web document Clustering: A Feasibility Demonstration. In *Proc. of SIGIR'98*, 46-54, Melbourne, August 1998.