Introducing Groups to an Annotation System

Amjad W. Hawash

Sapienza University, Rome, Via Salaria 113, Italy,
hawash@di.uniroma1.it

Abstract. Annotation of online documents has become an important practice, allowing the addition of valuable information to existing resources without corrupting them, thus introducing a novel way of user collaboration. The MADCOW annotation system provides services for annotating multimedia contents in HTML pages. Introducing groups with their related services to the MADCOW system enhances the security and privacy of annotations for group members as well as their collaboration. Usability problems arose with respect to the identification of significant matches between users' interests and group topics, to support the process of admitting users to groups. Therefore, the use of matching methods based on ontologies and URL comparison enhances the joining process with respect to time, effort and relevance. This research illustrates our current work for introducing groups to the MADCOW system, and inspects the Groups-Users relevance problem by analyzing and implementing relevance measurements.

Keywords: Annotation; Groups; Collaboration; Ontology.

1 Introduction

The MADCOW[1] system is a tool enabling its users to annotate contents (texts, images and video clips) of Web pages with textual annotations [1]. The 3-tier architecture of the system in Figure 1 guarantees security in the processing of handling users requests and in returning results.

Annotations published to MADCOW are saved in a dedicated server and can be of three types:

1. *Public*: viewable by any user.
2. *Private*: viewable by their submitters only.
3. *Group-related*: viewable by any member of the group to which they are posted, and nobody else.

Collaboration among users is enhanced by public annotations while privacy and security are enhanced by private ones [2, 3]. Groups were introduced in [4] as a solution to the collaboration-privacy and security conflict so that sets of users sharing interest in some domain will be able to conduct a collaborative

[1] Multimedia Annotation of Digital Content Over the Web

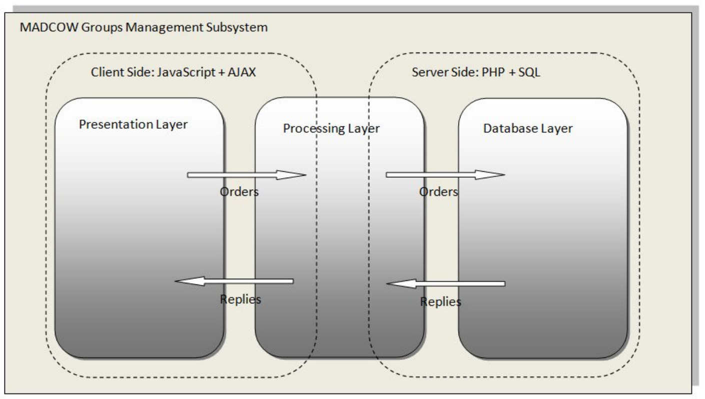


Fig. 1. MADCOW 3-tier Architecture.

discussions within one group, while keeping it protected and confidential with respect to users outside the group.

The MADCOW system supplies group owners with capabilities to manage their groups and to extend some of these capabilities to group *moderators* (group owner and moderators are collectively called *authorisers*). They are able to create, update and delete their groups, and to select one of three different policies to allow users to join their groups, i.e. to become *members*:

1. *Public*: any user can join the group.
2. *Invite*: users can join the groups only if invited by some group member with *authoriser* status.
3. *Apply*: users apply for joining the group, and admission is subject to approval by an authoriser.

A user invited to a group is given *reader* or *writer* privileges. A reader can only read annotations. A writer can both read and submit annotations.

Further operations allow a finer management of group composition. Atomic operations for management of group members and of their annotations have been formalized and used to describe more complex ones. In particular, group owners may execute the *Extraction*, *Union*, *Intersection* and *Subsumption* operations.

Originally, only a manual process was available to allow users to join a group: a group authoriser had to select from the list of all MADCOW users the set of users to be invited, while a user had to search for groups to send join requests to their authorisers.

Eight undergraduate classes from different disciplines, for a total of 152 students, have been involved in a supervised test which lasted for 15 days. Different Human Computer Interaction metrics (such as understandability of individual steps and interface ease of use) have been tested, while the time needed to complete each operation was collected in the background. Table 1 presents information about the number of times the different operations on groups were used and on the average time needed to complete an operation.

Table 1. Number of execution and average duration (secs) for operations.

	Create	*Update*	*Invite*	*Join*
# of times	72	51	719	125
Average	37.3	15.9	99.25	5.6

Two problems emerged with respect to this manual group joining process:

1. *Time-Effort*: The manual process is a tedious operation that consumes a lot of time (99.25 sec. as average invitation time from Table 1) and effort, especially with a huge number of users.
2. *Irrelevance*: Group authorisers have no idea about **"who are the interested users to invite"**, and users have little indications about **"which groups to join"**.

An automatic groups-users suggestion process has therefore been introduced. Authorisers are presented with the set of those users whose public annotations are mostly related to the group topics and could likely be interested in becoming writers. The same technique is used to suggest groups for users, so that they are presented with those groups that could interest them. *Ontology*-based and *URL*-based matching processes have been used to implement a group-user matching after defining ontological domains in the system, so that each domain is referred to some ontology and is the possible core topic for a set of groups.

Class Match Measure (CMM) presented in [5] is used to measure the group-user relevance in the ontology-based matching. Authorisers for a group may request a ranked list of probable writers. Also users can request a ranked list of domains, for each of which the set of groups referring to it is given. A pilot test on the use of CMM has shown a decrease in the average invitation time from 99.25 to 10.6 seconds, offering a promising solution to the time, effort and irrelevance problems.

This research represents a Ph.D. work supervised by **Prof. Paolo Bottoni** at the Department of Computer Science at Sapienza University of Rome, Italy.

2 Related Work

Several tools allow creation of annotations and collaboration between annotators in various domains, but few of them deal with groups, mostly supplying public annotations only. Among those which do offer groups, little support is given for group management, making the transfer of annotations among groups difficult.

The Diigo Toolbar[2] lets users create their own groups and invite other members, and provides some services for groups archiving and dissemination. A.nnotate[3] takes a snapshot of a document, Web page or image, to produce a read-only copy, which can be annotated and shared with other users, with some approximation to a sort of grouping. In Bounce[4] a user can produce notes for snapshots of Web pages and collaborate on them with other users.

Considering group operations, some work has been done in the context of classifying documents according to user annotations [6]. The work in [7] deals with the annotation of data inside databases and the propagation and merging of such annotations through queries.

Ontologies as representatives of knowledge were used in several domains. Paralic and Kostjal [8] enhanced the retrieval process by considering ontologies as representational schemes for domain knowledge. An efficiency comparison with the vector and the latent semantic indexing models obtained promising results. Patel *et al.* explored the use of ontologies in order to automate common clinical tasks, e.g. selection of a patient cohort for clinical trials, by considering the matching of patient records to clinical trials as a problem of semantic retrieval [9].

Several works deal with measuring the relevance of an ontology to a collection of terms. In [10], a similarity is defined between sets of concepts belonging to a common ontology. They define a similarity between a single concept and a set of concepts and between two sets of concepts establishing some criteria that should be met. An implementation of the relevant algorithms within the Jena framework[5] was also developed.

MADCOW enhances the annotation process by both introducing groups and its related services as well as solving the problem of group-user matching by combining ontology-based [11] and URL-based matches.

3 Applied Methodology

My Ph.D. project is concerned with analyzing and implementing all MADCOW group-related services in order to enhance the group joining process, ensure security and privacy, and support collaboration for group members. Currently, our work is focused on enhancing the process of joining groups by automating the suggestion of users to groups' authorisers and of groups to users.

3.1 Groups Services

The work on groups in MADCOW began by implementing their preliminary services (Adding, Updating and Deleting Groups) as well as:

[2] http://www.diigo.com/
[3] http://a.nnotate.com/
[4] http://www.bounceapp.com/
[5] http://jena.sourceforge.net

- **Group Management**: handling of invitations and applications to groups, appointment/removal of *moderators*, blacklists of group writers, which prevent them from adding new annotations until their removal from the blacklist, definition of restrictions on display of annotations in a group, listing all public groups in the system, and ordinary search for groups by their titles.
- **Atomic Groups Operations**: creation of new groups could be done manually by any user, or could be the result of atomic operations (except for subsumption). Considering groups with their contained members as sets of elements, the MADCOW system provides the following collection of set-based operations[6] (depicted in Figure 2):
  1. *Extraction*: creation of a new group from an existing one, while keeping the original one unchanged.
  2. *Union*: creation of a new group with all members from two existing ones, with the option to delete or keep the original ones.
  3. *Intersection*: creation of a new group with members in common between two existing ones.
  4. *Subsumption*: deleting a group whose members are all members of another one, absorbing their annotations in the latter.

A detailed description for all related groups management is given in [12]. The article contains a full description (both textual and through a mathematical formalisation) of all related services, the system architecture, depicting the system static structure and dynamic behavior, a comprehensive test that measures several HCI metrics, as well as two detailed scenarios illustrating the mechanisms for system services.

3.2 Current Work

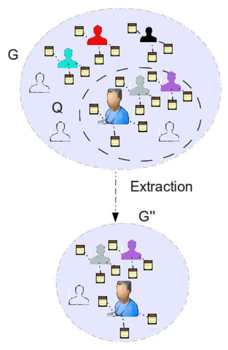
Our efforts now are directed to solve the problems (time, effort and irrelevance) of the group joining process. We propose an automatic matching process between users and groups, so that groups' authorisers are presented with a ranked list of the most relevant users, and users are presented with a ranked list of the most relevant groups. An ontology- and a URL-based matching processes are proposed to automate groups-users suggestions.

Current work (with its related terms, concepts, methodology) is described in [11] as well as the full mathematical representations, algorithmic approach, pseudocodes and illustrating scenario.

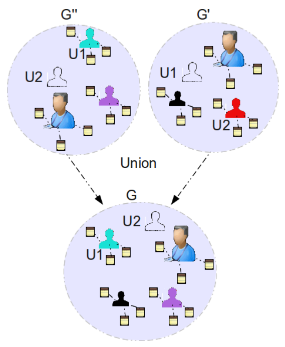
Ontology-Based Matching Before discussing the whole process, we introduce some definitions to illustrate the role of ontologies in the process of groups-users matching:

1. **Domain**: a unique name designating the area of knowledge with which an ontology is associated.

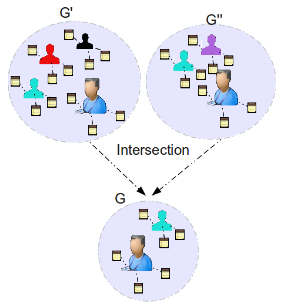
[6] Special considerations are implemented to manage members' privileges and annotations before and after each operation.



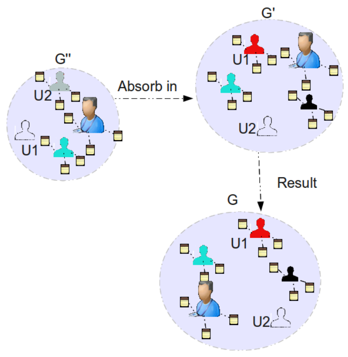
(a) Extraction



(b) Union



(c) Intersection



(d) Subsumption

Fig. 2. A schematic view of operations on groups.

2. **Concept**: the name associated with a node in some ontology.
3. **Terms**: lexicalisations of the concepts in some ontology.
4. **Tags**: words (possibly terms in some ontology) provided by a user to characterise the annotation content.

Domains-Ontologies Association Simply, the idea relies on using ontologies (as a universal source of knowledge) to represent sets of related groups, and executing a match between users' annotations (public and private) and concepts in ontologies available in a repository to better match groups and users. We introduced the notion of MADCOW domains, representatives for ontologies inside the system. Each domain has to be associated with an ontology before it is used to represent groups in the system. Associating ontologies with domains minimizes the time needed in the matching algorithm since each domain represents a set of groups. Referring a group to a domain could be done on a manual or automated basis. Groups' owners could manually refer their groups to the most relevant domains (by checking their associated ontologies concepts), or, if they are not satisfied with the available domains, they can provide the system with a set of terms that best describe the intention of their groups. The system performs a matching to present the most appropriate ontologies (ranked according to their group relevance) to let owners choose one of them. The selected ontology concepts become the tags to be used when submitting annotations to the group. Figure 3 depicts the Groups-Domains-Ontologies association.

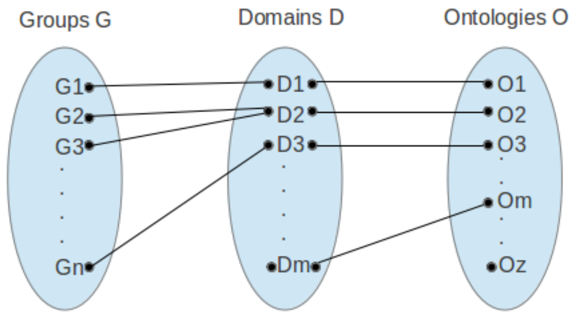


Fig. 3. Groups-Domains-Ontologies Association.

The following pseudocode illustrates the process of associating a group with a domain. The code starts by executing the function `selectGroup()` that saves the selected group in the variable `selectedGroup`. The names of all the available domains in the system are loaded via `loadDomains()` that saves them in `domains` variable. A loop is entered which first enables the owner to select one of the domains with `selectDomain(domains)` function; after that, the execution of `checkDomain(selectedDomain.getTerms())` enables the owner to check whether the domain terms represent the intention of the selected group or not. The loop keeps going till either the owner is satisfied with the selected domain, as indicated by the function `checkdomain()` returning `true`, or all the domains have been considered. In the first case the execution of `associate(selectedGroup,selectedDomain)` creates an association between the selected group and domain. In the second case, the owner can use the automatic support for group-domain association. This process starts by declaring an empty list `matchedOntologies`; then the function `askForTerms()` asks the owner to provide a set of terms which he or she deems representative of the group and saves them in the collection `terms`. A matching between these terms and the terms in all ontologies available in the `Ontology Repository` then takes place by executing the function `CMM()`. Each ontology that satisfies the matching condition is added to `matchedOntologies`. Finally, a call to `rank()` generates a ranking of all the contents of `matchedOntologies` according to their relevance values. The function `selectOntology()` enables the group owner to select the most appropriate ontology from the list of matched ontologies. The system then checks whether or not the selected ontology is associated with a domain by executing the function `isAssociated()`. If it is not associated, then the function `associateWithDomain()` is executed to associate it with a domain related to its title. In the final step, `associate()` generates an association between the selected group and the domain linked with the selected ontology.

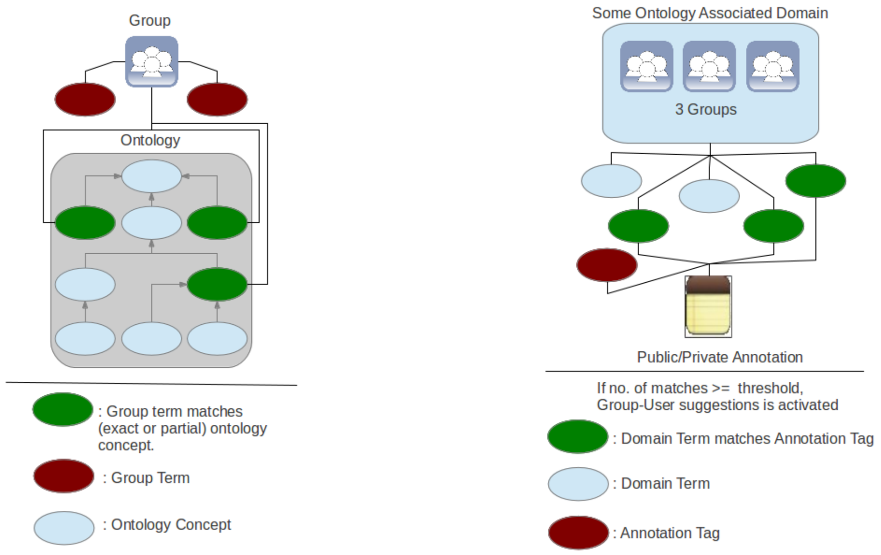
```
selectedGroup = selectGroup();
domains = loadDomains();
repeat {
  selectedDomain = selectDomain(domains);
    if(checkDomain(selectedDomain.getTerms())) {
      associate(selectedGroup,selectedDomain);
      break;
    }
} until(allDomainsConsidered());

if(!group.checkAssociation()) {
  matchedOntologies = new List();
  terms = askForTerms();
  foreach (ontology in MADCOW.Ontologies)
    if(CMM(terms,ontology.getTerms())>0)
      matchedOntologies.add(ontology);
  matchedOntologies.rank();
  selectedOntology = selectOntology();
  if(!selectedOntology.isAssociated())
    selectedOntology.associateWithDomain(selectedOntology.Title);
  associate(selectedGroup,selectedOntology.getDomain());
}
```

Annotations Submission For annotations submitted as public or private, users can use any set of tags as metadata associated with the annotation. Tags attached to these annotations will be used in the matching process to propose groups to users, and users to authorisers (only for public annotations).

The following pseudocode represents the submission of public, private and grouped annotations. The code first executes the function `submitToGroup()` to check if users submitted annotations to one of the groups they own or they are members in. In this case the function `assignTags()` is executed to make a group terms become annotation tags. If users decide to submit public or private annotations, the function `attachTagsToAnnotation()` is executed, in this case users are free to attach some informative tags to their annotations to better describe the intention for the annotation submission by executing the function `assignTags()` that executes the function `askForTags()` which in turn asks users to provide tags to be assigned with annotations.

```
if(submitToGroup(annotation,group))
  assignTags(annotation,getGroupTerms(group));
elseif(attachTagsToAnnotation())
  assignTags(annotation,askForTags());
```

Groups-Users Matching Group owners could ask the system to suggest users to their groups. As a consequence, the system executes a textual comparison between all ontology concepts and all the tags employed by users to adorn their public annotations. The same comparison takes place when users ask the system to propose some groups to send request joins. Comparison here includes both public and private annotations of users. Figures 4(a) and 4(b) depict the matching between a group and an ontology and between a domain and a user, respectively.


Group
Ontology
: Group term matches (exact or partial) ontology concept.
: Group Term
: Ontology Concept
Some Ontology Associated Domain
3 Groups
Public/Private Annotation
If no. of matches >= threshold, Group-User suggestions is activated
: Domain Term matches Annotation Tag
: Domain Term
: Annotation Tag

(a) Group-Ontology Matching (b) Domain-User Matching

Fig. 4. A visual depiction of Group-Ontology and Domain-User matches.

The *CMM* (exact and partial) is meant to evaluate the coverage of an ontology for a given set of search terms. By assuming that a given group is characterised by a set of terms, this measure is used to calculate to which extent an ontology covers the terms provided by the group. Hence, we measure the relevance of an ontology to a group in terms of matches between group terms and ontology lexemes. We use the same measure to calculate the relevance of groups for users depending on the terms associated with a group and the set of tags employed by users to adorn their annotations.

Domain-Annotation Matching Formalisation Given a domain D and an annotation A, let $L_D = \{r_1, r_2, \ldots, r_n\}$ be the set of lexicalisations of domain terms and let $T = \{t_1, t_2, \ldots, t_m\}$ be the set of tags. The matching between D and A is the process of checking the existence, within A, of a term from L_D, considering exact and partial matches.

URL-Based Matching We are also discussing another way for automating the groups-users suggestion by matching URLs that are annotated by both group members and non-grouped users, assuming them as indicators of common interests related to the group intent. From this point of view, group authorisers could be interested in those users sharing those interests. The same process could be applied by users who are searching for interesting groups to join. URL-based matching is illustrated in Figure 5.

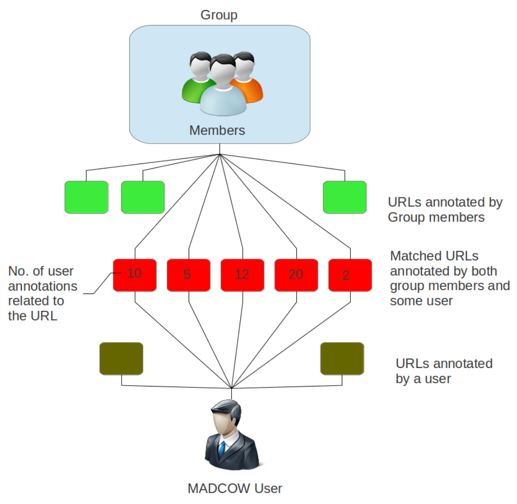

Group
Members
URLs annotated by Group members
No. of user annotations related to the URL
10
5
12
20
2
Matched URLs annotated by both group members and some user
URLs annotated by a user
MADCOW User

Fig. 5. URL Matching.

The following pseudocode illustrates the URL-based matching between a group G and all users not members in G:

```
groupURLs = G.getURLs();
users = getAllUsers();
usersMatch = new User[users.getSize()];
foreach (groupURLs as gURL)
  foreach(users as user)
    foreach(user.getURLs() as uURL)
      if(gURL == uURL) usersMatch[user]++;
ranks(usersMatch);
```

The code begins by executing the member function `getURLs()` that retrieves all annotated URLs related to the group G (for all members in G) and stores the result in the array `groupURLs`. The list of all users that are not members in G are stored in the array `users` as a result for executing the function `getAllUsers()`. A nested loops then are executed in which a list of all URLs annotated by a given user are retrieved by the member function `getURLs()` and matched against the array `groupURLs`. The result of the accumulative matching for every user with the user ID are stored as an element in the associative array `usersMatch` to be used after that in the function `ranks()` for ranking purposes.

3.3 Involved Methodology

We built a repository from 6 different ontologies: `Animals` (with 899 concepts), `Plants` (709), `Finance` (2037), `Artificial Intelligence` (2386), `Vehicles` (168) and `Viruses` (296) gathered from [13]. The ontologies are composed of set of concepts (classes) that have only IS_A relationship between them, with a text describing each class. Ontologies are represented via MySQL tables for faster access [14] and fast calculation of exact and partial matches. A fragment of the Entity-Relationship diagram for the ontology repository is shown in Figure 6.

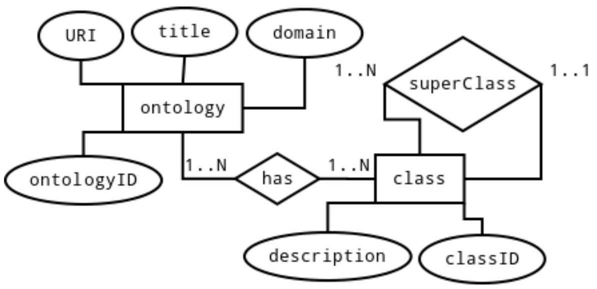


Fig. 6. A fragment of the Ontology Repository Scheme.

We conducted a pilot test on the matching process. 17 participants were divided into 3 disjoint sets (6, 6, 5). Users from the first set created 16 various MADCOW groups and assigned domains for them, then manually invited participants from the second set to join the groups (20 invitations were created). These accepted the invitations and submitted annotations to the joined groups (26 annotations). Participants in the third set submitted private and public annotations (31 annotations), all adorned by suitable tags. We asked users to annotate a set of similar websites to check URL matching (for this we used 10 different websites). Group owners requested the system to suggest members and invitations were sent (9 invitations). Participants of the third set required the system to suggest proper groups and sent membership requests (12 requests). By comparing the joint duration for group owners of the processes of obtaining user suggestions / selecting users / sending invitations with that of the manual invitation process (in the first work), we note that the average invitation duration decreased from 99.25 (in Table 1) to 10.6 seconds.

4 Conclusion and Future Works

Introducing groups to MADCOW system solved the tension between users' security and privacy and improved collaboration. Proper groups-users matching enhanced the groups-users suggestions, minimizing time and effort for both groups authorisers and users, and solved the irrelevance problem. The reduction in average time for user invitation between the two tests encourages us to investigate other relevance measures that could provide further refinements in groups-users matching as far as relevance is concerned.

References

1. Bottoni, P., Civica, R., Levialdi, S., Orso, L., Panizzi, E., Trinchese, R.: MADCOW: a multimedia digital annotation system. In: Proc. AVI'04, ACM (2004) 55–62
2. Heck, R., Luebke, S., Obermark, C.: A Survey of Web Annotation Systems (2008)
3. Wolfe, J.L.N., M., C.: From the margins to the center: The future of annotation. J. of Business & Technical Communication **15**(3) (2011) 333–371
4. Avola, D., Bottoni, P., Laureti, M., Levialdi, S., Panizzi, E.: Managing groups and group annotations in madcow. In: Proc. DNIS 2010. Volume 5999 of LNCS. (2010) 194–209
5. Alani, H., Brewster, C., Shadbolt, N.: Ranking ontologies with AKTiveRank. In: Proc. ISWC'06, Springer (2006) 5–9
6. Denoue, L., Vignollet, L.: Personal information organization using Web annotations. In: Proc. WebNet 2001. (2001) 279–283
7. Kostylev, E., Buneman, P.: Combining dependent annotations for relational algebra. In: Proc. ICDT 2012. (2012)
8. Paralic, J., Kostial, I.: Ontology-based information retrieval. In: Proc. IIS 2003. (2003) 23–28
9. Patel, C., Cimino, J., Dolby, J., Fokoue, A., Kalyanpur, A., Kershenbaum, A., Ma, L., Schonberg, E., Srinivas, K.: Matching patient records to clinical trials using ontologies. In: Proc. ISWC'07/ASWC'07, Springer (2007) 816–829
10. Cordì, V., Lombardi, P., Martelli, M., Mascardi, V.: An ontology-based similarity between sets of concepts. In: Proc. WOA 2005, Pitagora Editrice (2005) 16–21
11. Avola, D., Bottoni, P., Hawash, A.: Using ontologies for users-groups matching in an annotation system. In: Proc. CSIT 2013 5th International Conference on CSIT. (2013) 38–44
12. Avola, D., Bottoni, P., Hawash, A.: Introducing groups to an annotation system. Journal of Visual Languages and Computing (major revision in preparation)
13. Velardi, P., Faralli, S., Navigli, R.: OntoLearn reloaded: A graph-based algorithm for taxonomy induction. Computational Linguistics **39(3)** (2013)
14. Atzeni, P., Paolozzi, S., Nostro, P.D.: Ontologies and databases: Going back and forth. In: Proc. ODBIS 2008. (2008) 9–16