Semantic and bayesian profiling services for textual resource retrieval

Eufemia Tinelli
Department of Computer Science
University of Bari
Italy, Bari 70126
Email: tinelli@di.uniba.it

Pierpaolo Basile
Department of Computer Science
University of Bari
Italy, Bari 70126
Email: basilepp@di.uniba.it

Eugenio Di Sciascio
Dipartimento di Elettrotecnica ed Elettronica
Politecnico di Bari
Italy, Bari 70126
Email: disciascio@poliba.it

Giovanni Semeraro
Dipartment of Computer Science
University of Bari
Italy, Bari 70126
Email: semeraro@di.uniba.it

***Abstract*— This paper presents an integrated approach to textual resource retrieval, which combines logical inference services with user profiles, in which a structured representation of the user interests is maintained. Learning is performed on documents which have been disambiguated by exploiting the WordNet lexical database, in an attempt to discover *concepts* describing user interests. The proposed approach relies on several additional features compared to classical lexical knowledge systems, including: structured user recommendation, numeric value management, definition of strict and negotiable constraints and keywords to retrieve potential interesting resources w.r.t. both user request and profile.**

I. Introduction

The main goal of this paper is to propose a strategy to design advanced semantic search engines based on the idea of combining semantic matchmaking with Bayesian text categorization. By means of formal ontologies, modeled using OWL [24], the knowledge on a specific domain is modeled and exploited in order to make explicit the implicit knowledge, and to reason on it by means of the formal semantics expressed in OWL. On the other hand, a content-based recommender, which is able to learn user profiles from disambiguated documents, is used for customized search. The recommender exploits lexical knowledge in the linguistic ontology WordNet [26].

The success of a retrieval system strongly relies also on query formulation and ranking functions. Especially for an ontology-based system, the query language has to be very simple for the end user but, at the same time, its expressiveness must be able to capture the real user needs and to retrieve only what the user is really looking for. In this paper we present a system able both to help the user during the query formulation process via an intensional navigation of the ontology, and to return relevant resources via a ranking function exploiting both the ontology-related semantics of the query and the user profile managed by the content-based recommender.

Hence, the system suggests interesting items to user by taking into account three elements: user profiles, semantic item descriptions and lexical item descriptions.

The rest of the paper is structured as follows: the next section outlines the work which mainly inspired this paper. In Section III a brief summary of semantic-based matchmaking in Description Logics is presented together with a Naïve Bayes method for user profiling. The description of the framework architecture, a domain reference ontology, together with an example of query example satisfying user needs, is presented in Section IV, while conclusions close the paper.

II. Related Work

Recent years have witnessed a growing interest towards profiling based resource retrieval. Among the most relevant systems adopting a bayesian classifier we cite *LIBRA* [22] which produces content-based book recommendations by exploiting product descriptions obtained from Amazon.com Web pages. Documents are represented by using keywords and are subdivided into slots, each one corresponding to a specific section of the document as authors, title, abstract. *SiteIF* [20] exploits a sense-based representation to build a user profile as a semantic network whose nodes represent senses of the words in documents requested by the user. *OntoSeek* [14] explores the role of linguistic ontologies in knowledge-based retrieval systems. *AMAYA* [27] delivers context-aware recommendations, which are based on provided feedback, context data, and an ontology-based content categorization scheme. Former system deals with user profile and on this basis can provide a prediction/recommendation about interesting items for the end user but w.r.t ITR [11] uses content-based filtering algorithms.

Our reference system is ITem Recommender (ITR) whose strategy is to shift from a keyword-based document representation to a sense-based one in order to integrate lexical knowledge in the indexing step of training documents. Several methods have been proposed to accomplish this task. Inn [21] is proposed to include WordNet information at the feature level by expanding each word in the training set with *all* the synonyms for it, including those available for each sense, in order to avoid a word sense disambiguation (WSD) process [28]. This approach has shown a decrease of effectiveness in the obtained classifier, mostly due to the word ambiguity problem. In [28] is pointed out that some kind of disambiguation is required in any case. Subsequent works [3], [32] show that embedding WSD in document classification tasks can improve classification accuracy.

Besides, for improving search and visualization various *example-based* search tools have been developed, such as SmartClient [33]. SmartClient uses constraint satisfaction techniques, allows to refine (critique) preference values specified in the first step of the search and supports trade-off analysis among different attributes, *e.g.*, looking for an apartment a user can make a compromise between distance and rent (more distant less expensive). Also in [19] a candidate/critiques model has been presented, which allows users to refine candidate solutions proposed. Here, preferences are elicited incrementally by analyzing critiques through subsequent iterations. It is an *Automated Travel Assistant* (ATA) for planning airline travels, and similarly to SmartClient, ATA exploits CSP techniques: preferences are described using soft constraints defined on the values of attributes. AptDecision [31] is a tool supporting elicitation of preferences in the real estate domain: browsing the domain, users can discover new features of interest and through their refinement of apartment features, agents can build a profile of their preferences using learning techniques. FindMe [6] uses case-based reasoning as a way of recommending products in e-commerce catalogs. FindMe, and its enhanced version The Wasabi Personal Shopper [4], combines instance-based browsing and tweaking by difference. Different FindMe-like systems have been developed, in various domains. Among systems based on FindMe the most renowned is *Entrée* [5], a restaurant recommender, which allows users to refine a query on the basis of the results displayed, so it is possible to choose a restaurant less expensive or closer than the restaurant shown after the first query.

Recently, there has been a growing interest toward systems supporting semantics exploitation, in different domains. In [15] an application is presented, improving traditional web searching using semantic web technologies: two Semantic Search applications are presented, running on an application framework called TAP, which provides a set of simple mechanisms for sites to publish data onto the Semantic Web and for applications to consume these data via a query interface called GetData. The results provided by the system are then compared with traditional text search results of Google.it Web pages. Story Fountain [23] is an ontology-based tool, which provides a guided exploration of digital stories using a reasoning engine for the selection and organization of resources. Story Fountain provides support for six different exploration facilities to aid users engaged in exploration process. The system is being used by the tour guides at Bletchley Park. The approach has been further investigated in [8]. An intelligent query interface exploiting an ontology-based search engine is presented in [7]; the system enables access to data sources through an integrated ontology and supports a user in formulating a query even in the case of ignorance of the vocabulary of the underlying information system.

We do not present here related work on semantic matchmaking (the interested reader is referred to [9]) but only a framework of semantic-enabled e-marketplaces aimed at fully exploiting semantics of supply/demand descriptions in B2C and C2C e-marketplaces [13]. Main features of this framework are the followings: full exploitation of non-standard inferences for explanation services in the query-retrieval-refinement loop; semantic-based ranking in the request answering; fully graphical and usable interface, which requires no prior knowledge of any logic principles, though fully exploiting it in the back-office.

III. Basic Services and Algorithms

A close relation exists between OWL and Description Logics. In fact, the formal semantics of OWL DL sub-language is grounded in the Description Logics theoretical studies. We assume the reader be familiar with the basics of Description Logics and with two standard inference services provided by a DL reasoner: *Subsumption* and *Satisfiablity* [2].

Given a query Q and an item to be retrieved I the following match classes can be identified with respect to an ontology $\mathcal{T}$(see [12], [18], [25]).

- ***exact*** - $\mathcal{T} \models \mathsf{Q} \equiv \mathsf{I}$. I *is semantically equivalent to* Q. All the characteristics expressed in Q are presented in I and I does not expose any additional characteristic with respect to Q.
- ***full*** - $\mathcal{T} \models \mathsf{I} \sqsubseteq \mathsf{Q}$. I *is more specific than* Q. All the characteristics expressed in Q are provided by I and I exposes also other characteristics both not required by Q and not in conflict with the ones in Q.
- ***plug-in*** - $\mathcal{T} \models \mathsf{Q} \sqsubseteq \mathsf{I}$. Q *is more specific than* I. All the characteristics expressed in I are provided by Q and Q requires also other characteristics both not exposed by I and not in conflict with the ones in I.
- ***potential*** - $\mathcal{T} \not\models \mathsf{I} \sqcap \mathsf{Q} \sqsubseteq \bot$. Q *is compatible with* I. Nothing in Q is logically in conflict with anything in I.
- ***partial*** - $\mathcal{T} \models \mathsf{I} \sqcap \mathsf{Q} \sqsubseteq \bot$. Q *is not compatible with* I. Something in Q is logically in conflict with some characteristic in I.

With respect to the above classification, in case of potential match a similarity measure is needed to understand "*how potentially*" I satisfies Q.

The semantic similarity between a query and an item to be retrieved can be computed with the aid of the algorithm *rankPotential* [12]. Starting from the unfolded version (*i.e.*, normalized with respect to the reference ontology) of both the query and the item description, the algorithm is able to quantify how many information requested in the query are missing in the item description. In order to understand the approach we consider the following trivial example where the ontology is just a simple taxonomy[1].

$$\mathcal{T} = \begin{cases} B & \sqsubseteq & A \\ C & \sqsubseteq & B \sqcap E \\ D & \equiv & A \sqcap F \end{cases}$$

With respect to the previous $\mathcal{T}$ consider the query $\mathsf{Q} = C \sqcap D$ and the item $\mathsf{I} = E \sqcap B \sqcap G$. Referring to the above classification we see that Q and I are a **potential** match. Unfolding $\mathcal{T}$ in both Q and I we obtain $\mathsf{Q} = C \sqcap B \sqcap A \sqcap E \sqcap F$ and $\mathsf{I} = E \sqcap B \sqcap A \sqcap G$. Since in the ontology the third one is an equivalence axiom, we rewrite D with $A \sqcap F$ instead of expanding it as for B and C. Once we have the unfolded version of Q and I we say that **two** pieces of information $\{C,F\}$ are missing in I in order to completely satisfy Q (and then reach a full match). Since the maximum number of missing pieces of information is equal to the length of the unfolded Q, in this case **five**, we assign a normalized semantic similarity score of $(1 - \frac{2}{5})$ to the previous example match. Then in the most general case, given an ontology $\mathcal{T}$ and two concept Q and I, the semantic similarity score is computed by the following formula:

$$rank = 1 - \frac{rankPotential(\mathsf{I}, \mathsf{Q})}{rankPotential(\top, \mathsf{Q})} \tag{1}$$

where $\top$ is the most generic concept in every DLs ontology. Obviously the previous score is equal to **1** only in the case of *full match*.

On the other hand, we consider the problem of learning user profiles as a binary text categorization task [29]. Each document has to be classified as interesting or not with respect to the user preferences. Therefore, the set of categories is $C = \{c_+, c_-\}$, where c_+ is the positive class (***user-likes***) and c_- the negative one (user-dislikes). We present a method able to learn sense-based profiles by exploiting an indexing procedure based on WordNet.

We extend the classical BOW model [29] to a model in which the senses (meanings) corresponding to the words in the documents are considered as features. The goal of the WSD algorithm is to associate the appropriate sense s to a word w in document d, by exploiting its *context* C (a set of words that precede and follow w). The sense s is selected from a predefined set of possibilities, usually known as *sense inventory*, that in our algorithm is obtained from WordNet [26]. The basic building block for WordNet is the SYNSET (SYNonym SET), a set of words with synonymous meanings which represents a specific sense of a word. The text in d is processed by two basic phases: the document is first tokenized and then, after removing stopwords, part of speech (POS) ambiguities are solved for each token. Reduction to lemmas is performed and then synset identification with WSD is performed: w is disambiguated by determining the degree of *semantic similarity* among candidate synsets for w and those of each word in C. The proper synset assigned to w is that with the highest similarity with respect to its context of use. The semantic similarity measure adopted is the Leacock-Chodorow measure [17]. Similarity between synsets a and b is inversely proportional to the distance between them in the WordNet *is-a* hierarchy, measured by the number of hops in the shortest path from a to b. The algorithm starts by defining the context C of w as the set of words in the same slot of w having the same POS as w, then it identifies both the sense inventory X_w for w and the sense inventory X_j for each word w_j in C. The sense inventory T for the whole context C is given by the union of all X_j. After this step, we measure the similarity of each candidate sense $s_i \in X_w$ to that of each sense $s_h \in T$ and then the sense assigned to w is the one with the highest similarity score. Each document is mapped into a list of WordNet synsets following three steps.

1) each monosemous word w in a slot of d is mapped into the corresponding WordNet synset;
2) for each pair of words $\langle$*noun,noun*$\rangle$ or $\langle$*adjective,noun*$\rangle$, a search in WordNet is made to verify if at least one synset exists for the bigram $\langle w_1, w_2 \rangle$. In the positive case, the algorithm is applied on the bigram, otherwise it is applied separately on w_1 and w_2; in both cases all words in the slot are used as the context C of the word(s) to be disambiguated;
3) each polysemous unigram w is disambiguated by the algorithm, using all words in the slot as the context C of w.

A new version of the WSD algorithm has been recently produced [30].

The WSD procedure is used to obtain a synset-based vector space representation that we called Bag-Of-Synsets (BOS). In this model, a synset vector corresponds to a document, instead of a word vector. Each document is represented by a set of *slots*. Each slot is a textual field corresponding to a specific feature of the document, in an attempt to take into account also the structure of documents.

Formally, assume that we have a collection of N documents, each document being subdivided into M slots. Let m be the index of the slot, $n = 1, 2, ..., N$, the n-th document is reduced to 3 bags of synsets, one for each slot:

$$d_n^m = \langle t_{n1}^m, t_{n2}^m, \dots, t_{nD_{nm}}^m \rangle$$

where t_{nk}^m is the k-th synset in slot s_m of document d_n and D_{nm} is the total number of synsets appearing in the m-th slot of document d_n. For all n, k and m, $t_{nk}^m \in V_m$, which is the vocabulary for the slot s_m (the set of all different synsets found in slot s_m). Document d_n is finally represented in the vector space by M synset-frequency vectors:

$$f_n^m = \langle w_{n1}^m, w_{n2}^m, \dots, w_{nD_{nm}}^m \rangle$$

where w_{nk}^m is the weight of the synset t_{nk}^m in the slot s_m of document d_n and can be computed in different ways: It can be simply the number of times synset t_k appears in slot s_m or a more complex TF-IDF score. Our hypothesis is that the proposed indexing procedure helps to obtain profiles able to recommend documents semantically closer to the user interests.

[1]For the sake of simplicity in this example we do not consider roles even if *rankPotential* is able to deal with them for $\mathcal{ALN}$ ontologies.

As a strategy to learn user profiles on BOS-indexed documents, ITem Recommender (ITR) uses a Naïve Bayes text categorization algorithm to build profiles as binary classifiers (*user-likes* vs *user-dislikes*). The induced probabilistic model estimates the *a posteriori* probability, $P(c_j|d_i)$, of document d_i belonging to class c_j as follows:

$$P(c_j|d_i) = P(c_j) \prod_{w \in d_i} P(t_k|c_j)^{N(d_i,t_k)} \tag{2}$$

where $N(d_i, t_k)$ is the number of times token t_k occurs in document d_i. In ITR, each document is encoded as a vector of BOS, one for each slot. Therefore, equation (2) becomes:

$$P(c_j|d_i) = \frac{P(c_j)}{P(d_i)} \prod_{m=1}^{|S|} \prod_{k=1}^{|b_{im}|} P(t_k|c_j, s_m)^{n_{kim}} \tag{3}$$

where $S = \{s_1, s_2, \ldots, s_{|S|}\}$ is the set of slots, b_{im} is the BOS in the slot s_m of d_i, n_{kim} is the number of occurrences of token t_k in b_{im}. Training is performed on BOS-represented documents, thus tokens are WordNet synsets, and the induced model relies on synset frequencies. To calculate (3), the system has to estimate $P(c_j)$ and $P(t_k|c_j, s_m)$ in the training phase. The documents used to train the system are rated on a discrete scale from 1 to MAX, where MAX is the maximum rating that can be assigned to a document. According to an idea proposed in [22], each training document d_i is labeled with two scores, a "user-likes" score w_+^i and a "user-dislikes" score w_-^i, obtained from the original rating r:

$$w_+^i = \frac{r-1}{MAX-1}; \qquad w_-^i = 1 - w_+^i \tag{4}$$

The scores in (4) are exploited for weighting the occurrences of tokens in the documents and to estimate their probabilities from the training set TR. The prior probabilities of the classes are computed according to the following equation:

$$\hat{P}(c_j) = \frac{\sum_{i=1}^{|TR|} w_j^i + 1}{|TR| + 2} \tag{5}$$

Witten-Bell smoothing [34] is adopted to compute $P(t_k|c_j, s_m)$, by taking into account that documents are structured into slots and that token occurrences are weighted using scores in equation (4):

$$\hat{P}(t_k|c_j, s_m) = \begin{cases} \frac{N(t_k,c_j,s_m)}{V_{c_j}+\sum_i N(t_i,c_j,s_m)} & \text{if } N(t_k,c_j,s_m) \neq 0 \\ \frac{V_{c_j}}{V_{c_j}+\sum_i N(t_i,c_j,s_m)} \frac{1}{V-V_{c_j}} & \text{otherwise} \end{cases} \tag{6}$$

where $N(t_k, c_j, s_m)$ is the count of the weighted occurrences of token t_k in the slot s_m in the training data for class c_j, V_{c_j} is the total number of unique tokens in class c_j, and V is the total number of unique tokens across all classes. $N(t_k, c_j, s_m)$ is computed as follows:

$$N(t_k, c_j, s_m) = \sum_{i=1}^{|TR|} w_j^i n_{kim} \tag{7}$$

In (7), n_{kim} is the number of occurrences of token t_k in slot s_m of document d_i. The sum of all $N(t_k, c_j, s_m)$ in the denominator of equation (6) denotes the total weighted length of the slot s_m in class c_j. In other words, $\hat{P}(t_k|c_j, s_m)$ is estimated as the ratio between the weighted occurrences of t_k in slot s_m of class c_j and the total weighted length of the slot. The final outcome of the learning process is a probabilistic model used to classify a new document in the class c_+ or c_-. This model is the user profile, which includes those tokens that turn out to be most indicative of the user preferences, according to the value of the conditional probabilities in (6).

IV. System Features and Architecture

Based on the previous techniques we built a system (see 1) enabling users to perform:

- semantic searching by selecting ontology classes and properties;
- personalized searching based on user profiles and item information;
- semantic-personalized searching obtained by combining the two types of searching;

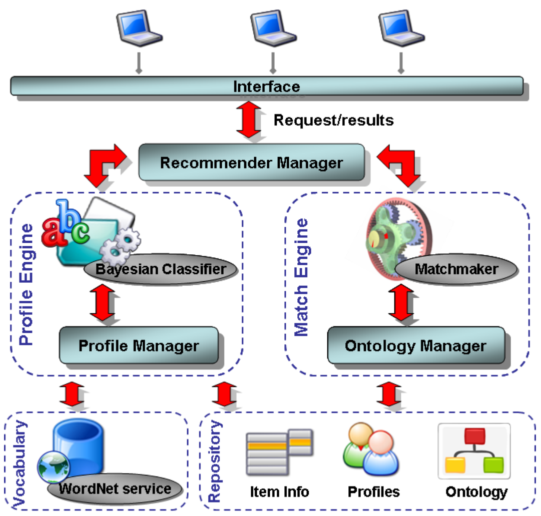

Fig. 1. Ontology-based recommender system architecture

In our approach, an item is represented by means of the following data:

- ***item information*** - defined by two sets of information. The first set is composed of intrinsic information. For example, in a bibliographic research scenario, intrinsic information could be: item identification number, authors, title, abstract, slot types. The second set is composed of information produced by the classifier during the training step: the bag of synsets obtained by the WSD process on each slot;

- ***user profile*** - learned by the ITR system, as described in the previous section;
- ***item semantic description*** - an OWL description of the reference ontology.

All the above information is stored in a repository, while the adopted reference vocabulary to define semantic profile is the WordNet lexical database. Recommender system architecture is composed of several modules and each one has a specific role and instantiates a part of the repository. The ***Interface Module*** allows to define semantic item description and user request. It provides a GUI to browse the hierarchy of concepts and to outline properties of the selected class. This feature of the GUI supports a user to define requests as descriptions which are logically consistent w.r.t. the reference ontology. The user request is split into two main parts:

- *full* : in the *full* part of the request, the user sets the constraints she wants to be satisfied by (in full match with) the retrieved items.
- *potential* : here the user sets her preferences, *i.e.*, her *wishful* options. The ontology-based score is computed measuring "how many" of these constraints are satisfied by a retrieved item.

We stress the fact that in order to fulfill user needs the research process could be an iterative one. On the basis of returned results, a user is able to refine the previous request exchanging a *full* constraint for a *potential* one and vice versa. Our recommender system considers user query features as *full* constraints by default and as *potential* only whenever explicitly stated by the user [10].

The ***Profile Engine Module*** prepares the item information by performing WSD on the textual descriptions of the items to be recommended. Mainly built on the ITR system, it performs the training step on the disambiguated text in order to infer the user profiles which will be exploited in the recommendation process. Each inferred user profile is a binary classifier able to categorize an item as interesting or not interesting according to the classification score of the class *user-likes*.

The ***Match Engine Module*** implements matchmaking and ranking algorithms. It allows to compare the query with the description of items referring to the same OWL ontology. The reasoner is not embedded within the application, so the *Ontology Manager* communicates with the *Matchmaker* via a DIG 1.1 interface over HTTP. The Match Engine is used to manage *full* and *potential* constraints. We think that when the user sets a request characteristic as full , she wants to be sure that *full* feature are explicitly mentioned in the item description to be retrieved. Matchmaker Engine evaluates full or potential match in the following way:

- *all* potential *constraints* - if user sets all request features as *potential* ones then returned items will be those potentially satisfying user request. According to an Open World Assumption, returned items could have additional or missing features w.r.t the request but no features in logical conflict with any in the request. In this case the module runs a potential match;
- *all* full *constraints* - if user sets all request features as *full* ones then returned items have to express explicitly at least all requested features. Of course this does not prevent the item description to include also not requested features. In this case the module evaluates a full match;
- *mixed* - in this case the module first compute a full matches considering a temporary request composed of *full* features only. The set of items returned in the previous step could contain also features in logical conflict with someone in the *potential* request, so the module runs a potential match with the *potential* part of the request to discard these results.

Results returned by Profile Engine are defined by the pair $\langle \textit{item identifier}, \textit{relevance rate} \rangle$ whilst the ones returned by Match Engine are defined by the pair $\langle \textit{item identifier}, \textit{rank value} \rangle$ according to equations (1) and (3).

The ***Recommender Manager*** allows the communication and synchronization between Profile and Match Engine. This module is designed to deal with two issues: results ordering and synchronization. In web-based retrieval systems, *ranked lists* are surely preferable to unordered sets of items. Nevertheless they become less effective and usable as the complexity of the item description increases, together with possible user preferences. At least two implementations of the Recommender Engine are possible:

1) Recommender Manager operates in two steps. In the first step, this module receives user requests. They are OWL description translated in DIG; then it activates Profile and Match Engine. In the second step, Recommender Engine fetches user request results by two previous modules, then it computes a unique score adding match rank and classifier relevance rate value. The new ranked results will be sent to the GUI;
2) Match Engine works as a filter to produce a subset of items which is sent to Profile Engine. Profile Engine uses only this subset to answer to the user request.

Our recommender system implements the first approach and computes results score according to the following simple formula: $score = \alpha * relevance + \beta * rank$ where α and β are numeric coefficients. As an initial attempt we set both of coefficients to 0.5 value. The evaluation of several experimental tests and usage of different reference ontologies could require the change of the previous coefficients values.

As real scenario we propose queries formulated in terms of an ontology which models the bibliographic research domain [16]. The ontology is defined by the following classes (Figure 2):

- `Item` - has subclass as `Article`, `inProceeding`, `Book`;
- `Event` - has subclass as `Conference`, `Workshop`, `Meeting`;
- `Topic` - topic hierarchy will be defined by specific research topic and we may use Computer Science terms of ACM Topic Hierarchy [1] for example;
- `Author` - author hierarchy will be defined by sev-

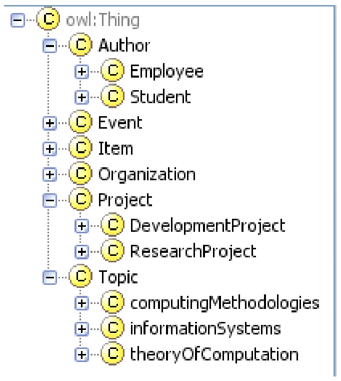

Fig. 2. Reference Ontology Hierarchy

eral professional figure like `AcademicStaff`, `Manager`, `PhDstudent`;

- `Organization` - has subclass as `University`, `Enterprise`, `Institute`;
- `Project` - project hierarchy will be defined by specific research project adding properties as `financedBy`;

Besides, the item concept can be defined by the following properties: *aboutProject* - has `project class` as range -, *hasPublicationYear*, *presentedAt* - has `event class` as range -, *hasAuthor* - has `author class` as range -, *developedBy* - has `organization class` as range -, *hasTopic* - has `topic class` as range. Obviously this domain ontology can be extended by specific journal item properties like `hasVolume`, `hasMonth` and by specific book item properties like `hasPublisher` and `hasEdition`.

Finally, in order to retriev only interesting items several disjoint sets are defined. For example `Book` is disjoint by `Article`, `inProceeding`, `inBook`, `inCollection` and `Proceedings` while `inProceeding` is disjoint by `Book`, `Thesis`, `Booklet` and `Manual`.

In this scenario a user can propose queries such as the combination of (a)*"I'm looking for an inProceeding item published after 2004, developed in a research project by both Enterprise and University"* and (b)*"I am interested in items with `matchmaking` as keyword in title, including my profile"*. In the previous request (a) is the semantic query for matchmaker, (b) is the profile-based query for the classifier. According to domain ontology the previous semantic query (a) is translated in the following DL description $\texttt{inProceeding} \sqcap (\geq 2004\ \texttt{hasPublicationYear}) \sqcap \forall \texttt{aboutProject}.(\texttt{ResearchProject} \sqcap \forall \texttt{developedBy}.(\texttt{Enterprise} \sqcap \texttt{University}))$.

V. Conclusion

In this paper, we have described a strategy to design an advanced semantic search engine able to combine logic based matchmaking with Bayesian text categorization. The use of Wordnet has been exploited in order to enhance a text based categorization exploited by a Bayesian approach for automated user profile learning. Combining probabilistic and logic based similarity measure we have shown how to compute a score representing the match degree between a query and an item description. The score takes into account both the ontology-based query and the user profile.

An initial prototype implementing the proposed approach has been developed and presented in the paper. Currently, we are performing experiments on large datasets in order to validate the proposed approach.

References

[1] ACM. Top Two Levels of The ACM Computing Classification System . www.acm.org/class/1998/overview.html, 1998.

[2] F. Baader, D. Calvanese, D. Mc Guinness, D. Nardi, and P. Patel-Schneider. *The Description Logic Handbook*. Cambridge University Press, 2002.

[3] S. Bloedhorn and A. Hotho. Boosting for text classification with semantic features. In *Proc. of 10th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, Mining for and from the Semantic Web Workshop*, pages 70–87, 2004.

[4] R. Burke, M. Claypool, A. Gokhale, T. Miranda, P. Murnikov, D. Netes, and M. Sartin. Integrating knowledge-based and collaborative-filtering recommender systems. In *Proceedings of the Workshop on AI and Electronic Commerce. AAAI 99*, Orlando, Florida, 1999.

[5] Robin Burke. Knowledge-based recommender systems. In A. Kent, editor, *Encyclopedia of Library and Information Systems*, volume 69. New York, 2000.

[6] Robin D. Burke, Kristian J. Hammond, and Benjamin C. Young. The findme approach to assisted browsing. *IEEE Expert*, 12(4):32–40, 1997.

[7] Tiziana Catarci, Paolo Dongilli, Tania Di Mascio, Enrico Franconi, Giuseppe Santucci, and Sergio Tessaris. An ontology based visual tool for query formulation support. In *Proceedings of the 16th European Conference onArtificial Intelligence (ECAI '04)*, pages 308–312, 2004.

[8] Trevor Collins, Paul Mulholland, and Zdenek Zdráhal. Semantic browsing of digital collections. In *proc. of 4th International Semantic Web Conference (ISWC 2005)*, pages 127–141, 2005.

[9] Simona Colucci, Tommaso Di Noia, Eugenio Di Sciascio, Francesco M. Donini, and Marina Mongiello. Concept abduction and contraction for semantic-based discovery of matches and negotiation spaces in an e-marketplace. *Electronic Commerce Research and Applications*, 4(4):345–361, 2005.

[10] Simona Colucci, Tommaso Di Noia, Eugenio Di Sciascio, Francesco M. Donini, and Azzurra Ragone. Knowledge elicitation for query refinement in a semantic-enabled e-marketplace. In *7th International Conference on Electronic Commerce ICEC 05 ACM Press*, pages 685–691. ACM, 2005.

[11] Marco Degemmis, Pasquale Lops, and Pierpaolo Basile. An intelligent personalized service for conference partecipants. In *16th International Symposium on Methodologies for Intelligent Systems (ISMIS'06)*, 2006.

[12] Tommaso Di Noia, Francesco M. Di Sciascio, Eugenio andDonini, and Marina Mongiello. A system for principled matchmaking in anelectronic marketplace. *International Journal of Electronic Commerce*, 8(4):9–37, 2004.

[13] Eugenio Di Sciascio, Simona Colucci, Tommaso DiNoia, Francesco M. Donini, Azzurra Ragone, and Raffaele Rizzi. A semantic-based fully visual application formatchmaking and query refinement in b2ce-marketplaces. In *8th International conference on ElectronicCommerce, ICEC 06*, pages 174–184. ACM, ACM Press, 2006.

[14] N. Guarino, C. Masolo, and G. Vetere. Content-based access to the web. *IEEE Intelligent Systems*, 14(3):70–80, 1999.

[15] Ramanathan V. Guha, Rob McCool, and Eric Miller. Semantic search. In *Proceedings of the Twelfth International World Wide Web Conference, WWW2003*, pages 700–709, 2003.

[16] P. Haase, J. Broekstra, M.Ehrig, M. Menken, P.Mika, M. Plechawski, P. Pyszlak, B. Schnizler, R. Siebes, S. Staab, and C. Tempich. Bibster - a semantics-based bibliographic peer-to-peer system. In *the International Semantic Web Conference (ISWC2004)*, 2004.

[17] C. Leacock and M. Chodorow. *Combining local context and WordNet similarity for word sense identification*, pages 305–332. In C. Fellbaum (Ed.), MIT Press, 1998.

[18] L. Li and I. Horrocks. A Software Framework for Matchmaking Based on Semantic Web Technology. *International Journal of Electronic Commerce*, 8(4), 2004.

[19] Greg Linden, Steve Hanks, and Neal Lesh. Interactive assessment of user preference models: The automated travel assistant. In *Proceedings of the Sixth International Conference on User Modeling*, pages 67–78, Vienna, 1997.

[20] B. Magnini and C. Strapparava. Improving user modelling with content-based techniques. In *Proc. 8th Int. Conf. User Modeling*, pages 74–83. Springer, 2001.

[21] George Miller. Wordnet: An on-line lexical database. *International Journal of Lexicography*, 3(4), 1990. (Special Issue).

[22] Raymond J. Mooney and Loriene Roy. Content-based book recommending using learning for text categorization. In *Proceedings of the* 5^{th} *ACM Conference on Digital Libraries*, pages 195–204, San Antonio, US, 2000. ACM Press, New York, US.

[23] Paul Mulholland, Trevor Collins, and Zdenek Zdráhal. Story fountain: intelligent support for story research and exploration. In *Proc. of Intelligent User Interfaces Conf.*, pages 62–69, 2004.

[24] OWL. Web Ontology Language. www.w3.org/TR/owl-features/, 2004.

[25] M. Paolucci, T. Kawamura, T.R. Payne, and K. Sycara. Semantic Matching of Web Services Capabilities. In *proc. of International Semantic Web Conference (ISWC 2002)*, number 2342 in LNCS. 2002.

[26] Princeton University. WordNet. http://wordnet.princeton.edu/, 2005.

[27] Christian Räck, Stefan Arbanowski, and Stephan Steglich. Context-aware, ontology-based recommendations. In *international Symposium on Applications and the Internet Workshops (SAINTW'06)*, 2005.

[28] Sam Scott and Stan Matwin. Text classification using wordnet hypernyms. In *COLING-ACL Workshop on usage of WordNet in NLP Systems*, pages 45–51, 1998.

[29] F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1), 2002.

[30] Giovanni Semeraro, Marco Degemmis, Pasquale Lops, and Pierpaolo Basile. Combining learning and word sense disambiguation for intelligent user profiling. In *Twentieth International Joint Conference on Artificial Intelligence, January 6-12, 2007, Hyderabad, India (to appear)*, 2007.

[31] Sybil Shearin and Henry Lieberman. Intelligent profiling by example. In *Proceedings of the International Conference on Intelligent User Interfaces (IUI 2001)*, pages 145–151. ACM press.

[32] M. Theobald, R. Schenkel, and G. Weikum. Exploting structure, annotation, and ontological knowledge for automatic classification of xml data. In *Proceedings of International Workshop on Web and Databases*, pages 1–6, 2004.

[33] Marc Torrens, Boi Faltings, and Pearl Pu. Smartclients: Constraint satisfaction as a paradigm for scaleable intelligent information systems. *Constraints*, 7(1):49–69, 2002.

[34] I.H. Witten and T.C. Bell. The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. *IEEE Transactions on Information Theory*, 37(4), 1991.