P2P Case Retrieval with an Unspecified Ontology

Shlomo Berkovsky[1], Tsvi Kuflik[2], and Francesco Ricci[3]

[1] University of Haifa, Computer Science Department
slavax@cs.haifa.ac.il
[2] University of Haifa, Management Information Systems Department
tsvikak@is.haifa.ac.il
[3] ITC-irst, Trento
ricci@itc.it

Abstract. Traditional CBR approaches imply centralized storage of the case base and, most of them, the retrieval of similar cases by an exhaustive comparison of the case to be solved with the whole set of cases. In this work we propose a novel approach for storage of the case base in a decentralized Peer-to-Peer environment using the notion of Unspecified Ontology. In our approach the cases are stored in a number of network nodes that is comparable with the number of cases. We also develop an approximated algorithm for efficient retrieval of most-similar cases. The experiments show that the approximated algorithm successfully retrieves the most-similar cases while reducing the number of cases to be compared.

1 Introduction

Peer-to-Peer (P2P) networks provide a distributed computing platform with theoretically unlimited storage, communication and processing capabilities. P2P systems, however, lack any notion of centralized management, depending rather on distributed autonomous management of the resources contributed by the connected peers. Storage resources of P2P systems were used until now for data sharing and distribution, mainly multimedia files. P2P is a fast growing research area that is producing interesting applications [9].

This paper proposes using the resources of a P2P network as a case-based reasoning tool for the purposes of distributed storage of a case base and to support distributed retrieval of similar cases. Since a P2P system does not require central management, and in fact this is the primary motivation for P2P solutions, the cases inserted autonomously by a community of users are described in a fully distributed way, possibly using different features for the same type of cases. This implies that similar cases might be not only different with respect to certain feature values, as in classical CBR approaches, but also capable of being described in different ways (different features, different names for the same concept). Thus, efficient management of the case base requires a stable semantic infrastructure allowing the identification of similarities between these heterogeneous cases. Moreover, this infrastructure should support a retrieval process that: i) maintains decentralized retrieval with low communication overhead, and ii) guarantees discovery of the most similar cases or at least a reasonably good approximation.

H. Muñoz-Avila and F. Ricci (Eds.): ICCBR 2005, LNCS 3620, pp. 91 – 105, 2005.

We used the hypercube-based approach of UNSO (UNSpecified Ontology) [1] to maintain P2P storage of the case base and we developed an approximated algorithm for case retrieval that copes with the constraints of such a distributed storage structure (e.g. reduction of node to node communication). The algorithm is based on the notion of implicit clustering in UNSO, i.e., on the fact that similar cases are inherently located in a relatively close vicinity. Thus, an approximated retrieval algorithm over UNSO is performed through a localized search. This means that the case most similar to a given probe is not searched in the whole case base, but rather that the search focuses on the nodes (storing cases) closer to the node storing the probe, thus decreasing the number of target cases that are compared with the case to be solved. Moreover, since the underlying network consists of distributed connected peers, the computation effort needed to assess the case similarity might also be spread among the peers, eliminating a single computation bottleneck in traditional centralized CBR systems.

The above approach was evaluated in five case bases storing E-Commerce advertisements from five different domains. We have compared the proposed approximated retrieval algorithm with a traditional exhaustive algorithm, and we showed that the former significantly decreases the number of cases that are compared during the retrieval stage, while preserving the essential quality of the retrieved cases. In particular, we measured the recall of the proposed algorithm and the number of case similarity computations performed. We achieved high recall results while still keeping the number of similarity computations smaller than that required by an exhaustive linear search.

The ideas of distributed problem solving in CBR arose in multi-agent CBR systems such as those proposed in [10] and [11], and more recently in [7] and [8]. In [10] the task was solved by iterative negotiations between the agents, targeted to resolve the constraints' conflicts, while in [11] the agents exploited the past experience of other agents. Our approach is quite different, since we assume that the agent/nodes of the P2P network jointly contribute to a single retrieval rather than participating in the problem solving process by performing independent retrievals or exchanging problem solving knowledge.

Thus, the contributions of this work are two-fold. First, we propose a novel notion of pure decentralized storage of the case base in a P2P mode. Second, we develop and evaluate an efficient approximated algorithm for the retrieval of the most similar cases over the above platform.

The rest of this paper is structured as follows. In Section *2* we present semantic approaches to P2P data management. In Section *3* we describe our case model and the UNSO distributed storage. In Section *4* we define the distance metric we have used in our evaluation, and we present the approximate case retrieval algorithm. Finally in Section *5* we present the results of the empirical evaluation, and we conclude in Section *6* summarizing the obtained results and mentioning future lines of research.

2 P2P and Semantic Data Management

Peer-to-Peer (P2P) computing refers to a subclass of distributed computing, where functionality is achieved in a decentralized way by using a set of distributed

resources, such as computing power, data and network traffic. P2P systems usually lack a dedicated centralized infrastructure, but rather depend on the voluntary contribution of resources by the connected peers. Systems based on the P2P approach are usually characterized by one or more of the following advantages: cost sharing/reduction, improved scalability/reliability, resource aggregation and operability, increased autonomy, dynamism, and high levels of anonymity/privacy [9].

A number of content-addressable P2P systems for data sharing, such as CAN [12] and Pastry [13] were recently developed. These applications implement a highly scalable self-organizing infrastructure for fault-tolerant routing using distributed hash tables (DHT) [6]. In DHT-based systems users and resources are assigned unique identifiers (called *nodes* and *resources*, respectively) from a sparse space. As accepted in the P2P system, the resources are distributed and stored at the user-side, i.e, each user $node_i$ manages a set of resources $R_i=\{resource_1, resource_2, \ldots, resource_n\}$. The resources are inserted into the system through $put(user_i, resource_j)$ operation that uses globally-known hashing functions that assign $user_i$ as a provider of $resource_j$ by coupling their identifiers in the DHT. As such, DHT is also partitioned and stored distributively at the user-side. This setting facilitates further discovery of key_j's provider by other users through $get(key_j)$ operation exploiting the same global hashing mechanism.

DHT-based systems are highly scalable, and provide a robust, self-organizable, and completely decentralized structure. Due to an effective routing algorithm that routes the messages to the relevant peers only instead of expensive network flooding, their overall traffic is significantly lower [12,13]. However, DHT-based systems basically rely on hashing-based *put()* and *get()* operations. This results in two major limitations:

- Support for exact-matching lookups only. Similar, but not identical keys key_1 and key_2, will be treated as two diverse resources. Hence, just the searches, specifying the same term used at the insertion of a *key*, will succeed to locate it.
- Support for single-key lookups only. The above *put()* and *get()* primitives handle a single key only, i.e., a resource is described by a single string. Although the key might be represented as a concatenation of the substrings representing parts of the key, any minor change in one of them will prevent identifying the matching.

This has lead to the development of a more complex kind of P2P network, built upon peers using their own, not shared, schemas to describe the objects. This approach is further referred to as semantic or ontology-based data management. A key concept in semantic data management is ontology, i.e., a formal shared conceptualization of a particular domain of interest [5]. It acts as a standardized reference model, providing both human-understandable and machine-processable semantic mechanisms, allowing enterprises and application systems to collaborate efficiently. Techniques for ontology-based data management in P2P networks were initially proposed in HyperCup [14] and further extended in UNSO [1].

HyperCup [14] proposes a flexible ontology-based P2P platform generating a hypercube-like graph of users, where each user is treated as a data source. HyperCup needs predefined ontology of the domain, such that the dimensions of the hypercube match the concepts (features characterizing the domain) of the ontology. According to the above ontology, each user is categorized as the provider of particular data. This categorization determines the location of the user within the hypercube. Thus, the

hypercube is virtually constructed of the connected users, whereas each user maintains a data structure of its respective neighbors. For example, in *3*-dimensional hypercube a node located in coordinates (*x,y,z*) will be connected to 6 neighbors: (*x+1,y,z*), (*x-1,y, z*), (*x,y+1,z*), (*x,y-1,z*), (*x,y,z+1*) and (*x,y,z-1*). The user providing data from a number of domains will maintain a set of hypercube locations, i.e., a separate data structure of neighbors for each location.

Due the fact that the ontology is predefined, and remains unchanged, data sources providing the same content are mapped to the same location. Moreover, if the possible values for ontology concepts (features) can be ordered a priori (as for instance in the feature "quality of the paper") then providers of similar contents are mapped to close locations. This implies causes the formation of so-called "concept clusters", which facilitate multiple-key search by efficient semantic routing of queries constructed as a logical combination of ontology concepts. However, such a setting where data providers share a single global ontology would require central management of the ontology, contradicting the decentralized spirit of a P2P network.

3 Case Representation and Storage in UNSO

UNSO (UNSpecified Ontology) [1] extends the above ideas by assuming that the domain ontology is not fully defined and that parts of it can be dynamically specified by the peers. The description of the resource is relatively free and is represented as an "unspecified" vector $<f_1:v_1, f_2:v_2,\ldots, f_n:v_n>$, where f_i corresponds to a feature of the resource being described, and v_i to the value of the respective feature.

To manage pure distributive storage and retrieval of cases, we adopt the UNSO representation of a case as an unspecified vector. Hence, the cases are represented as a dynamic list of features and their respective values. Different domains may exploit different features or values to describe a case. For example, a medical case may contain features describing the patient and the disease, whereas, a weather forecasting case base could include various geographical and climatic features. Note that when operating in a pure decentralized environment without any form of central management and any predefined ontology, different data sources might represent cases from the same domain in different ways. As such, neither the set of features specified when describing a particular case, nor their respective values can be anticipated. Thus, the main target of UNSO is to manage ontologies that can grow and support updating in a fully distributive way.

In detail, the UNSO generalization of a regular notion of ontology is performed in the following manner:

- The ontological vector description can dynamically grow by allowing it to be formulated by the users as a list of pairs $feature_i:value_i$. Two hash functions are used to map the "unspecified" vector to its location in the hypercube: one maps the feature $feature_i$ to a dimension of the hypercube, while another maps the value $value_i$ to a coordinate value within that dimension. For example, consider a case describing E-Commerce advertisement: <*manufacturer:BMW* | *engine volume:3000* | *year of production:1987*>. This is inserted into the hypercube by applying $hash_1$(*manufacturer*), $hash_1$(*engine volume*), and $hash_1$(*year of production*) to obtain the relevant dimensions of the hypercube, while $hash_2$(*BMW*), $hash_2$(*3000*)

and $hash_2(1987)$ will determine the coordinate values in the above dimensions. We note that using two hash functions allows the order of $feature_i$:$value_i$ pairs appearing in the case model to be ignored.

- The description can be organized in a "hierarchical" multi-layered structure (instead of a single "flat" vector), constructing a hypercube, the vertices of which recursively contain other hypercubes. This is regarded as a multi-layered hypercube (MLH). For example, a 2-layered ontology with three vectors <*manufacturer* | *color* | *year of production*> + < *engine volume* | *transmission* | *ABS*> with two possible values for each slot will generate a hypercube with *8* recursive nodes, containing "inner" hypercubes of up to *8* nodes. For example, if *ABS* feature might be present only in cars of production years *1970* and after, there is no sense in entering this dimension in the inner hypercubes of cars produced before *1970*. This would be compared to a fixed size *64*-nodes hypercube, had we used one flat vector for the whole ontological case model.

The conversion of a fixed specified ontology to the Unspecified Ontology is summarized in figure *1*. In the case of fixed ontology, a predefined set of features and values is mapped to a location in the underlying hypercube graph. Conversely in UNSO the number of $feature_i$:$value_i$ pairs in the description of a case is unlimited and their order is insignificant. Thus, UNSO dynamically generates a hypercube-like graph structure, where each vertex is recursively made of another hypercube. Note that the ontology of inner hypercubes is also dynamic and depends on the properties characterizing the cases which correspond to the location in the upper-level hypercube.

To address the problem that cases may have different terms with the same semantic meaning (synonyms), the $feature_i$ names are standardized using WordNet [3]. In WordNet, English nouns, verbs, adjectives and adverbs are organized into synonym sets, each representing one underlying lexical concept. For each concept, the set of synonyms can be sorted according to the frequency of usage. To eliminate possible ambiguity and improve the precision, the terms mentioned by the user in the description of a case undergo a simple semantic standardization, substituting the original terms with its most frequent synonyms. Thus, similar, but not identical terms specified by the user as $feature_i$ names, are replaced by a single 'representative' term.

As a negative effect of the hashing used in UNSO, the order relations on the stored resources are lost, since neither are the keys anticipated, nor do hashing primitives keep the distance relation. Despite this, UNSO supports the notion of concept clusters due to the fact that contents whose categorizations are identical with respect to a subset of ontology concepts are mapped to locations, whose coordinates have a common identical subset of coordinates. For example, two cases <*manufacturer:BMW* | *color:red* | *year of production:1987*> and <*manufacturer:BMW* | *color:green* | *year of production:1987*> will obviously be located closer than two arbitrary vectors, as two features (coordinates in UNSO) do overlap.

Thus, UNSO provides a hypercube-based dynamic infrastructure for distributed and fully decentralized management of the case base. The cases, modeled as unspecified ontological vectors, are organized in the hypercube in such a way that similar cases are located in close locations, thus facilitating efficient answering of semantic queries. Moreover, the connectivity maintenance protocols developed in [14] keep the hypercube stable despite sporadic joinings and departures of cases.

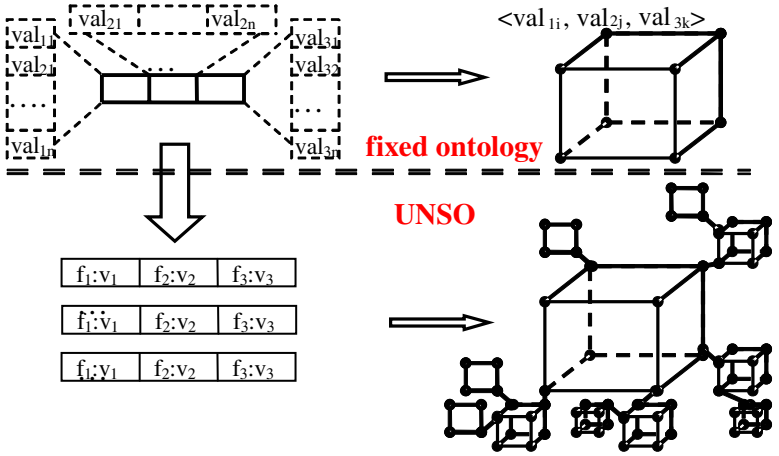

Fig. 1. Generalization of the Fixed Ontology to the Unspecified Ontology

In this paper we consider a case base that stores E-Commerce advertisements (in short, ads). Ads can be divided to two categories: supply ads where users offer products or services in exchange for payment, and demand ads where users seek products or services provided by other users. The system is required to provide a matching functionality between appropriate demand and supply ads. In a decentralized setting, where the users do not use any predefined forms when inserting demand and supply ads into the system, matching functionality is harder to achieve, as the system should be capable of finding the relevant ads basing on possibly partial and incomplete descriptions of the items.

In CBR terms, the case base is a collection of supply ads, stored in a distributed manner and containing a set of descriptions of the products the users are interested in selling. Each case (ad) is formulated as a list of $<feature_i:value_i>$ pairs, while neither properties nor values being known a priori. Hence, the descriptions of demand ads might be incomplete in comparison to the existing supply ads. The demand ad serves as a problem to be solved, whereas the supply ads in the case base provide possible solutions. Since the main target is providing matching functionality, the system should implement the search of the most-similar supply ads, i.e., cases. In the next section we shall describe the similarity metrics on possibly incomplete cases, and the retrieval algorithm.

4 Case Similarity and Retrieval

Retrieving the most-similar cases is one of the primary goals of a CBR system. The more accurate and efficient the similarity assessment, the more quickly the system will indicate the most useful cases. Two policies for case retrieval are typically implemented:

- Retrieve a set of K most-similar cases - calculates the similarity for each target case, ranks the cases according to their similarities and returns K highest cases from the ranked list.
- Retrieve a set of cases whose similarity is above a given threshold β - calculates the similarity for each target case and returns the case if the similarity is higher than β.

Each of the above-mentioned policies requires that the similarity function to be explicitly defined. In this paper we compute the similarity of two cases c_1 and c_2 as $1\text{-}dist(c_1,c_2)$, where *dist* is a distance metrics. When cases are represented as a list of $<feature_i:value_i>$ pairs, in order to compute the distance between the cases, we consider each feature separately. Hence, for the homogeneous cases (all the cases contain the same set of features), and assuming a linear dependency between the features and the distance, the distance metrics $dist(c_1, c_2)$ is defined by:

$$dist(c_1, c_2) = \sum_{i=1}^{|d|} w_i \cdot dist(c_1^i, c_2^i),$$

where d denotes the set of features specified in the cases, w_i is the normalized ($\Sigma w_i=1$) relative weight of the feature f_i, and $dist(c_1^i, c_2^i)$ is the local distance metrics between c_1 and c_2 with respect to the same feature f_i.

To compute the local distance between the values of two features we consider the possible types of values [2]. For Boolean values (true or false), the distance is a trivial comparison between two values, giving *0* if the two values are equal or *1* otherwise. For numeric features, the distance is the difference between them normalized by dividing it by the maximal possible difference (range). Moreover, the distance between two locations in a tree-like taxonomy is defined as the shortest path distance between the two nodes on the tree.

For symbolic or free-language values the distance can be computed as the difference between the numeric representations of the values. Although in particular domains the translation between symbolic and numeric values can be performed manually by a human expert, it is not clear whether it is feasible in any domain, especially if the features are not anticipated by the system. As the current work focuses on distributed retrieval of similar cases, feature distance metric for symbolic and free-language values is defined similarly as for the Boolean features, by:

$$dist(c_1^i, c_2^i) = \begin{cases} 0 & c_1^i = c_2^i \\ 1 & \text{otherwise} \end{cases}$$

Note that both k-nearest neighbor and above threshold retrieval techniques require that the case to be solved be compared with the whole set of the cases in the case base. Given a case base containing descriptions of N cases, the number of case distance computations needed for these retrievals is $O(N)$. More efficient multidimensional retrieval techniques, such as those based on K-d trees [4], were proposed in [15]. K-d tree uses a multi-dimensional tree for management and retrieval of cases. This technique requires $O(N\,logN)$ to build the tree and $O(logN)$ to retrieve the most-similar case (for $N>>2^d$). However, these techniques are applicable when cases are

described only with numeric features and they do not resolve the issue of cases with symbolic or free-language feature values. Moreover, it is not obvious how this algorithm could be implemented in a distributed environment, as that described here, where case feature values are retrieved exploiting node-to-node communication.

As we noted above, in a P2P environment with no predefined ontology, users can describe their cases in a relatively free form, hence cases will differ in terms of mentioned features. Therefore, we define a distance metrics for cases with an arbitrary set of features. But, since we are interested in retrieving similar cases (certainly, from the same domain), we can assume that there will be a set of overlapping features that are mentioned both in c_1 and c_2. Thus, we modify the distance function to compute the distance between two cases c_1 and c_2 exploiting only the set of overlapping features:

$$dist'(c_1, c_2) = \sum_{i=1}^{|d'|} w_i \cdot dist(c_1^i, c_2^i),$$

where *d'* denotes the set of overlapping features, w_i is their normalized relative weight, and $dist(c_1^i, c_2^i)$ is the local distance metrics between the features. We note that similarity ranges between *0* and *1* since also the distance is normalized.

We have exploited the P2P infrastructure of UNSO and the implicit clustering of similar cases for improving the efficiency of case retrieval. Figure *2* presents the pseudo-code of the algorithm we propose for the retrieval of cases whose similarity with the case to be solved is above a threshold β.

Initially, the algorithm determines the location of the case to be solved using a hashing mechanism similar to that used while inserting cases into the hypercube (steps *1-3*). Two functions are used to map the case. The first determines the dimensions of the hypercube and the second the coordinate values within the dimensions. Then the algorithm analyzes the cases stored in adjacent locations by assessing the similarity between each candidate case and the case to be solved (steps *4-7*). The rationale is that in UNSO similar cases are located in close vicinity and therefore to find similar cases we can check only a small portion of the case base. In other words UNSO provides an implicit indexing of the case base. The similarity between each one of the cases and the case to be solved is calculated using the distance metrics discussed above. If the similarity value is higher than the threshold β, then the retrieved case is added to the set of cases with required similarity (steps *8-9*). Finally, the whole set of appropriate cases is returned (step *10*).

For example, consider the following representation of a case to be solved c_s=<*manufacturer:Ford* | *doors:2* | *color:red*> and a flat 3-dimensional hypercube. Assume that the case c_s is mapped to a location (*2,3,4*) in the hypercube. Then the search will compare the c_s with the cases located at (**,3,4*), (*2,*,4*), and (*2,3,**), where * denotes any possible value in the respective dimension. In simple words, c_s will be compared against all the cases describing either *2*-door red cars, or red Fords, or *2*-door Fords.

The search for top-*K* similar users is performed in a similar manner; however, the length of the set of *retrieved_cases* is limited to *K* and every appropriate case is added to *retrieved_cases* in such a way that the whole set remains sorted.

```
Retrieve (target_case, β)
(1)  map target_case to the hypercube of dimension n
(2)  assume the location of target_case is (c1,…,cn)
(3)  let retrieved_cases be a set of cases, initially empty
(4)  for i=1 to n
(5)  for each possible value x of the i^th coordinate
(6)    let current be the set of cases stored in
       the location (c1,…,ci-1,x,ci+1,…,cn)
(7)    for each test_case∈ current
(8)    if sim(target_case, test_case) > β
(9)      retrieved_cases = retrieved_cases ∪ test_case
(10) return retrieved_cases
```

Fig. 2. Algorithm for retrieving cases with similarity metrics above β

Note that the above algorithm will compute an approximated solution since it checks only cases with one modified feature with respect to the original case to be solved. Intuitively, this is motivated by the fact that the higher number of modified features will be reflected in a lower similarity. Thus, the probability of discovering a similar case decreases with the number of modified features. However, in sparse case bases, or for the retrieval of the K most-similar cases (for large K), it might be necessary to search cases with more modified features. In order to adapt the algorithm to this constraint, the loop in step *4* is replaced by a nested loop that will retrieve cases from locations with a higher number of modified coordinates.

For the sake of simplicity, the above algorithm discusses the retrieval of similar cases over a flat hypercube. Converting a flat hypercube to MLH will have a minor impact on the retrieval process. The search for modified coordinates will partially take place in another hypercubes. For example, for *2*-leveled *3*-dimensional cubes $<f_1{:}v_1, f_2{:}v_2, f_3{:}v_3>+<f_4{:}v_4, f_5{:}v_5, f_6{:}v_6>$ the searches for the modified values of f_4, f_5, and f_6 will take place in the current secondary-level hypercube, while the searches for f_1, f_2, and f_3 will require cases stored in another secondary hypercubes (siblings of the current hypercube) to be accessed.

As already noted above, the number of comparisons in a naïve retrieval algorithm is $O(N)$, where N is the number of cases in the case base. Intuitively, the complexity of UNSO-based retrieval is significantly lower, as the case to be solved is compared with only a subset of the cases in the case base. The complexity of the above algorithm is $O(nk)$, where n is the dimension of the hypercube, and k is the maximal number of values for each dimension. Insertion of a case into the hypercube will require routing it to its proper location and connecting it to the neighbors in $O(nk)$ steps. Thus, total computational complexity of managing and retrieving cases in UNSO-based structure is lower than the complexity of naïve exhaustive retrieval for a sufficiently large case base.

Moreover, in this setting the retrieval stage is parallelized and the required computational effort is decreased, as the similarity of cases might be computed in parallel in each node (i.e., processed by the user managing a particular node of the hypercube). To implement this, the query with the description of the case to be solved is sent in parallel over all the dimensions. Upon receiving a query, each node forwards the

query to the next neighbor in a pipeline manner and initiates local computations. Upon discovering cases with the required similarity, nodes asynchronously send them back to the neighbor from which the query was received.

In summary, the semantic mechanism of Unspecified Ontology facilitates maintaining the case base as a distributed hypercube-like graph with a stable structure. The proposed approximated algorithm allows efficient retrieval of the most similar cases that spreads the required computational effort among the users comprising the hypercube.

5 Experimental Results

To validate the proposed retrieval algorithm, five corpuses of real-life E-Commerce ads from different domains were collected from *http://www.recycler.com* Web-site (*61* refrigerator ads with *10* different features mentioned, *65* camera ads with *13* different features, *76* television ads with *11* different features, *94* printer ads with *11* different features, and *130* mobile phone ads with *9* different features), giving in total *426* ads. Most of the ads contain three to four features. Before inserting the ads into the system, each ad was manually converted to the form of an ontological vector. For example, an ad "Philips 50FD995 50" plasma television, new in box, $4800" was converted to the following ontological description: *<price:4800, manufacturer:Philips, model:50FD995, size:50, screen:plasma, condition:new>*. The conversions were done keeping as close as possible to the original contents of the ads to mimic the insertions by naïve users.

The above UNSO-based model for storage and retrieval of cases was implemented. The number of dimensions in the hypercubes was not limited, i.e., it was equal to the number of different features mentioned by the users in their ads. The cardinality of cube dimensions (the range of coordinates in each dimension) was chosen to be *7*. Version *2.0* of WordNet was used during the insertions of ads to standardize the names of the features and to decrease the semantic ambiguity.

5.1 Retrieval Capabilities

In this experiment we retrieved the cases from the hypercube in two ways. Initially we retrieved the set R_e of the most similar cases using a regular CBR approach. This approach exhaustively compares the case to be solved with each one of the cases stored in the case base and retrieves only those cases whose similarity is above a given threshold β. Then we retrieved the set of the most-similar cases R_u using the above retrieval algorithm over the UNSO hypercube. The efficiency of our approach was quantified by dividing the cardinality of the set retrieved using the UNSO retrieval method by the cardinality of the set retrieved by the exhaustive search. As R_e is the true set of cases with the required similarity, this metrics is notated as the recall of the retrieval.

$$recall = \frac{|R_u|}{|R_e|}$$

Recall values are always less than *1* because the set of cases checked by the approximate retrieval algorithm is a subset of the whole case base; hence we can only miss some case that could be in the exact retrieval set.

Figure *3* shows the measurements of the recall as a function of the similarity threshold β and the maximal allowed number of modified features Δ (i.e., the number of nested loops in step *4* of the algorithm). The experiments were conducted on the corpus of *94* printer ads. In each execution a single case to be solved was chosen, and the total recall was computed as an average of recall values for each chosen case (the number of executions is equal to the size of the corpus).

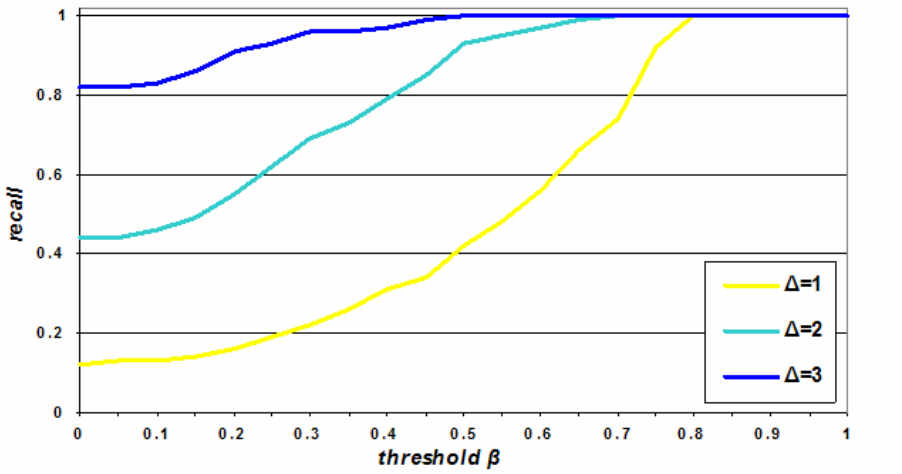

Fig. 3. Recall of the retrieval vs. similarity threshold β for different values of Δ

The results show that for high values of β (similarity close to 1) the recall converges to *1*, i.e., the cardinalities of both sets tend to be equal. This means that for high values of β (for retrieval of relatively similar cases) the set of cases retrieved using the UNSO approach is roughly equal to the set of cases retrieved by the traditional exhaustive approach. For low values of β the recall is low; however, this search might retrieve many cases with low similarity, that are inapplicable in the further adaptation stage of the CBR process. Increasing the value of Δ increases the number of pairs of cases that are compared. Thus, for higher values of Δ the recall converges more quickly and optimal recall is obtained even for relatively low values of β.

The same observation is true also for the other domains. The recall increases with β and the curve converges more quickly to *1* with the increase in Δ. Figure *4* shows the recall for different domains as a function of β for Δ=*2*. Certainly, the origin of the different behavior of the curves is in the different types of data in the domains. For example, recall might be influenced by the number of different features in the ads, their density and so forth. Elaborate analysis of the data is beyond the scope of this work.

A similar behavior could be observed when comparing the results of both retrieval approaches for top-K retrieval. R_e denotes the set of cases retrieved by exhaustive CBR search, and R_u the set of cases retrieved using UNSO. We gradually increase K – the number of cases to be retrieved. The goal of the experiment was to measure the

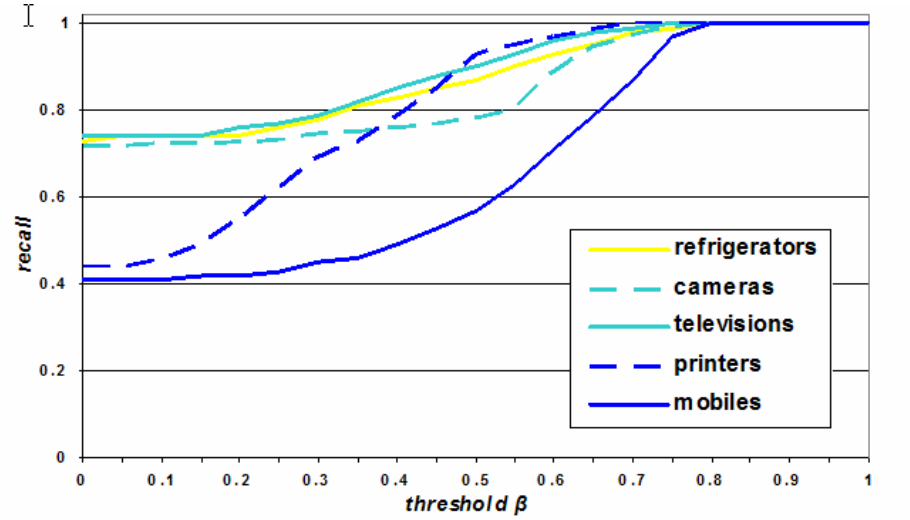

Fig. 4. Recall of the retrieval vs. similarity threshold β for different domains of cases (Δ=2)

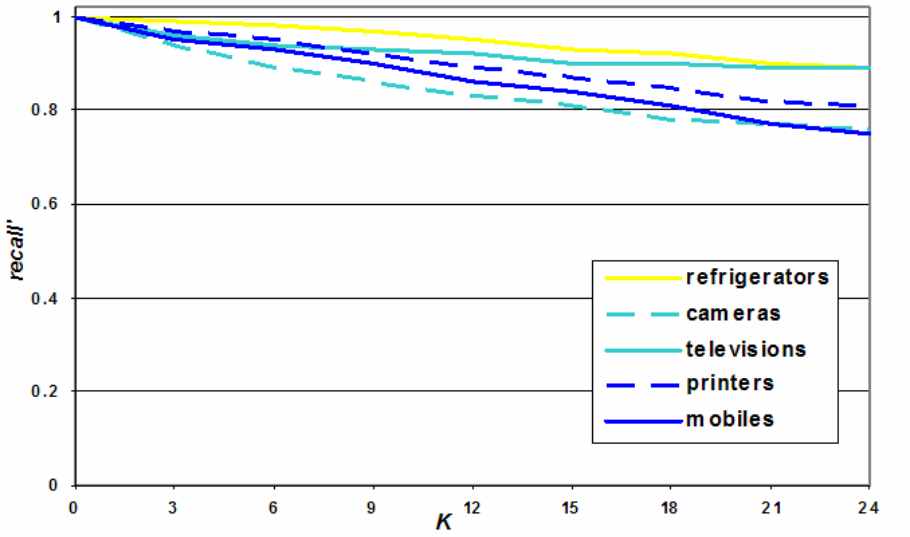

Fig. 5. Recall of top-*K* retrieval vs. *K* for different domains of cases

quality of UNSO retrieval, i.e., to verify that the cases retrieved using UNSO really belong to the set of *K* most-similar cases. We modified the recall metrics to be

$$recall' = \frac{|R_u \cap R_e|}{|R_e|}$$

Figure *5* shows the results of *recall'* measurements for different domain as a function of *K,* with Δ=*2*. It can be seen that for low values of *K* (retrieval of highly similar cases only) there is a high correlation between the set retrieved using UNSO and the real set of *K* most-similar cases found by the exhaustive search. For higher values of *K* (and lower threshold of similarity), *recall'* decreases, as R_e contains fewer similar cases that might not be retrieved using UNSO. Note that for high values of *K* (not illustrated in the figure) *recall'* rises and finally will converge to *1* for *K* equal to the size of the case base. Thus, in the experiment *K* was limited to *24*.

5.2 Computational Optimization

The result of the approximation applied in the UNSO-based retrieval is a decrease in the number of cases compared during the retrieval process. The needed computational effort is reduced because the case to be solved is compared only with cases with at most Δ modified properties, instead of with the whole set of cases stored in the case base. Moreover, as the cases are stored distributively, the comparisons are performed at the user-side, thus not involving central processing. This resolves a possible computational bottleneck of central processing and allows additional spreading of the computational effort.

In this experiment we performed over the threshold retrieval and compared the number of evaluated pairs of cases for both exhaustive and UNSO-based retrieval. In each execution a single case to be solved was chosen, and the total number of compared cases was calculated as an average of the number of comparisons for each chosen case. In this experiment the number of executions was also equal to the size of the relevant corpus.

Figure *6* shows the average number of comparisons in a single UNSO-based retrieval as a function of Δ (the first triplet of bars in each quadruplet), and compares it with the average number of comparisons in the exhaustive retrieval (the fourth bar). The experiments were performed for all the available domains of cases.

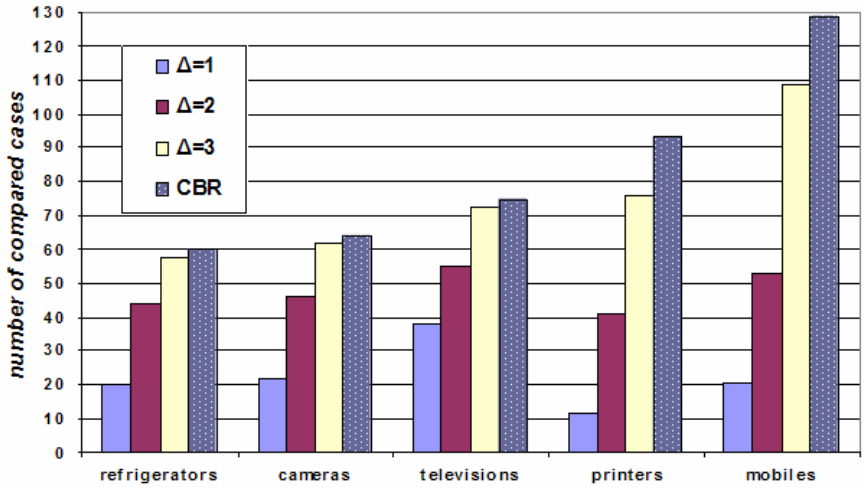

Fig. 6. Average number of comparisons vs. Δ for different domains of cases

The results clearly show that, although in every domain the number of comparisons increases with Δ, even for Δ=*3* it is still lower than in the regular exhaustive retrieval. The ratio of the number of comparisons in UNSO-based retrieval divided by the number of comparisons in the exhaustive search varies as a function of the domain. We hypothesize that this factor depends heavily on the characteristics of data in the domain (density, number of mentioned features and so forth).

6 Conclusions and Future Research

In this paper we presented a novel approach to P2P storage of cases using a hypercube-like graph built using UNSpecified Ontology (UNSO). This facilitates the use of an approximated search algorithm for similar cases retrieval.

The experiments showed that the approximated search succeeds in retrieving the most-similar cases. The results are pleasing both for *K*-nearest neighbor and threshold retrieval techniques. The sets of cases retrieved by the exhaustive technique and UNSO tend to be equal for low values of *K* or high thresholds, while the number of comparisons is significantly lower compared to the traditional exhaustive search.

We plan to extend our work by using a more precise distance metrics that computes similarity of heterogeneous cases using different sets of features, takes into account the relative size of the matched case with respect to the total case, and handles local similarity distance for the values of symbolic or free-language features. We are also interested in exploiting different learning algorithms for determining the relevance metrics for the features and identifying their weights w_i.

Moreover, we plan to investigate the influence of different domains and different types of data (for example, the density of the ads in the hypercube, the average number of features used in the ads, the total dimension of the hypercube, and so forth) on the performance of the approximated algorithm. We also plan to conduct large-scale experiments with real-life case bases containing a high number of cases.

References

1. Y. Ben-Asher, S. Berkovsky, "UNSO: Unspecified Ontologies for Peer-to-Peer E-Commerce Applications", In Proc. of the International Conference on Informatics, Turkey, 2004.
2. L. Coyle, D. Doyle, P. Cunningham, "Representing Similarity for CBR in XML", In Proc. of European Conference on Advances in Case-Based Reasoning, Spain, 2004.
3. C. Fellbaum, "WordNet - An Electronic Lexical Database", The MIT Press Publishers, 1998.
4. J.H. Friedman, J.H. Bentley, R.A. Finkel, "An algorithm for finding best matches in logarithmic expected time", in ACM Transactions in Mathematical Software, vol.3(3), 1977.
5. T.R. Gruber, "A translation approach to portable ontology specifications", Knowledge Acquisition Journal, 6(2), pp. 199–221, 1993.
6. M. Harren, J.M. Hellerstein, R. Huebsch, B.T. Loo, S. Shenker, I. Stoica, "Complex queries in DHT-based Peer-to-Peer networks", In Proc. of the International Workshop on Peer-to-Peer Systems (IPTPS'02), MA, 2002.

7. D.B. Leake, R. Sooriamurthi. “When Two Case Bases are Better then One: Exploiting Multiple Case Bases”. In Proc. of International Conference on Case-Based Reasoning, Canada, 2001.
8. L. McGinty, B. Smyth. “Collaborative Case-Based Reasoning: Applications in Personalised Route Planning”, In Proceedings of International Conference on Case-Based Reasoning, Canada, 2001.
9. D.S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, Z. Xu, “Peer-to-Peer Computing”, Technical Report HPL-2002-57, HP Labs, 2002.
10. M.V. Nagendra Prasad, V. Lesser, S. Lander, “Retrieval and Reasoning in Distributed Case Bases”, in Journal of Visual Communication and Image Representation, Special Issue on Digital Libraries, vol.7(1), 1996.
11. E. Plaza, J.L. Arcos, F. Martin, “Cooperative Case-Based Reasoning, In Proc. of the Workshop Distributed Artificial Intelligence Meets Machine Learning, Hungary, 1996.
12. S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Shenker, “A Scalable Content-Addressable Network”, In Proc. of ACM SIGCOMM, CA, 2001.
13. A. Rowstron, P. Druschel, “Pastry: Scalable, distributed object location and routing for large-scale Peer-to-Peer systems”, In Proc. of International Conference on Distributed Systems Platforms (Middleware), Germany, 2001.
14. M. Schlosser, M. Sintek, S. Decker, W. Nejdl, “A scalable and ontology-based P2P infrastructure for semantic Web services”, In proc. of IEEE International Conference on Peer-to-Peer Computing, Sweden, 2002.
15. S. Wess, K-D. Althoff, G. Derwand, “Using K-d Trees to Improve the Retrieval Step in Case-Based Reasoning”, In Proc. of European Workshop on Case-Based Reasoning, Germany, 1993.