UNSO: Unspecified Ontologies for Peer-to-Peer E-Commerce Applications[1]

Yosi Ben-Asher, Shlomo Berkovsky
{yosi,slavax}@cs.haifa.ac.il

Computer Science Department, University of Haifa
31905 Haifa, Israel

Key words: Ontology, Peer-to-Peer, E-Commerce

Abstract

In this work we consider a general-purpose application for E-Commerce transactions over P2P network. Weakly organized structure of P2P systems may cause data management problems and high communication overhead. We resolve these problems by developing a novel semantic approach to efficiently create, search and organize demand and supply ads in P2P E-Commerce applications. The approach is based on the notion of Unspecified Ontology (UNSO).

Unlike many existing systems, using a global predefined ontology, UNSO approach assumes that the ontology is not fully defined, leaving some parts of it to be dynamically specified by the users. The ads, inserted to the system, organize a multi-layered hypercube graph, providing an infrastructure for semantic search and routing operations. The proposed method has the potential of becoming a practical infrastructure for P2P-based publish-locate applications.

1. Introduction

Peer-to-Peer (P2P) technology, developed in the recent years, offers a solid alternative to the traditional client-server model of computing. While client-server architecture typically bases on a single or small number of servers, in pure P2P systems every node (peer) acts as both the client and the server at the same time. P2P approach allows a dynamic set of users to efficiently share resources without any centralized management. The decentralized nature of P2P is reflected in the following advantages: nearly unlimited scalability, high privacy and anonymity, and low costs. Sharing of the resources guarantees robustness and high availability of P2P systems.

In this work we examine the issue of implementing an infrastructure, dedicated for E-Commerce transactions over P2P networks. We consider the problem of developing an infrastructure, supporting insertion and search of general-purpose E-Commerce advertisements (ads) of *supply* and *demand* types. The service, provided by the system is a matching of appropriate demand and supply ads. This is further referred as publish-locate functionality of the system.

In the existing E-Commerce systems a user, publishing or searching an ad, is usually required to fill a predefined form with fixed fields, describing the ad. For example, in *www.carsmart.com* site, a user seeking to buy a car is required to fill the following fields: manufacturer, geographical location, and the range of prices. On the other hand in *www.motocar.co.il* site, a user is asked to fill a complex form, containing manufacturer, model, the range of production years, gearbox type, and engine volume. This approach of predefined forms with the properties of objects is referred as ontology-based approach. Ontology is a formal explicit specification of the terms in a particular domain. Thus, the form describing a particular type of objects is regarded as the ontology of a domain, while the fields of the form are the ontology slots.

HyperCup [9] proposed a flexible ontology-based hypercube graph topology for P2P networks. In HyperCup, a global predefined ontology was used to categorize peers as providers of particular information, associated with the ontology slots. This categorization determined the location of a peer in the hypercube and allowed further search of any desired information in a bounded number of steps. Decentralized algorithms, capable of constructing and maintaining the hypercube graph (nodes dynamic joins and departures), were also developed.

Note that HyperCup is a pioneer work showing the use of ontologies for semantic routing in P2P system (see [7] and for a discussion on routing in P2P networks). It proposed a novel approach for managing P2P systems, where the ontology enabled locating a data in a dynamic graph of clients. Thus, HyperCup formed a new alternative to distributed hashing [8, 10, 6], flooding [1], local routing tables [2] and others fundamental search techniques in P2P systems. HyperCup approach of predefined ontologies is applicable for a specific kind, or for a limited set of E-Commerce transactions. However, it is not suitable for general-purpose E-Commerce systems, implying constantly changing and unknown set of products and transactions.

A possible solution could be allowing peers to add new types of ontological forms. However, this will flood the system with new types of forms that must be distributed to the peers. Another solution could be developing a comprehensive all-including ontology, containing forms for all possible types of products. This will result in huge and barely manageable structure and will require the peers to share this global ontology, obstructing it from being expanded. All these restrictions raise a problem of developing a flexible mechanism for managing a dynamic set of ontological forms.

In this work we developed a novel approach of an Unspecified Ontology (UNSO). UNSO approach premises that the domain ontology is not fully defined and parts of it can be dynamically specified by the peers. For a semantic routing we extend the original hypercube graph of HyperCup to a multi-layered

[1] The support of The Caesarea Edmond Benjamin de Rothschild Foundation Institute for Interdisciplinary Applications of Computer Science is gratefully acknowledged.

hypercube (MLH) that can be schematically depicted as a hypercube, where each vertex recursively contains another hypercube. We use hashing to deal with the unspecified nature of the ontology and with the variety of terms that can be used. This allows the peers to distributively manage a dynamically growing ontology and uniformly distributes the ads among the MLH. The generated structure allows employing semantic routing algorithms, similar to those, developed in HyperCup. To eliminate ambiguity and enhance system precision, the terms used by the peers in the ontological description of an object, undergo simple semantic standardization using WordNet [3]. In summary, the main contribution of UNSO is in the novel notion of ontologies (as a technique for managing a dynamic set of forms) and its accompanied semantic routing.

The rest of the paper is organized as follows. In section 2 we briefly review the major classes of P2P systems, and discuss the ontology-based approach in P2P networks. Section 3 presents the notion of UNSO, and discusses the generalization of fixed ontologies to UNSOs. Section 4 discusses the details of UNSO implementation. In section 5 we present the experimental results. Section 6 concludes our work and discusses the directions of further research, based on the current work.

2. Background and Prior Work

Peer-to-Peer computing refers to a subclass of distributed computing, where functionality is achieved in a decentralized way by a set of distributed resources (computing power, data, and network traffic). P2P systems usually lack dedicated centralized infrastructure, rather depending on a contribution of resources by the connected peers. The systems, based on P2P approach, are usually characterized by one or more of the following advantages: cost sharing, improved scalability, resource aggregation, increased autonomy, dynamism, anonymity, and collaboration.

The first P2P systems were intended for a large-scale data sharing. Applications like Freenet [2] and Gnutella [1] allowed peers to download data, shared by the other peers. The performance of these systems suffered from severe problems. For example, flooding algorithms of Gnutella limited the system scalability and did not allow proper functioning over a heterogeneous set of peers. Freenet, despite being fully decentralized and using efficient algorithms, could not guarantee reliable data location.

All these problems lead to the development of content-addressable P2P systems, such as CAN [6], Pastry [8] and Chord [10]. All of them implemented highly scalable self-organizing and load-balancing infrastructure for fault-tolerant routing based on distributed hash tables (DHT). In these systems nodes and resources are assigned random identifiers (called *nodeids* and *keys* respectively) from a sparse space. A resource can be inserted by *put(key,value),* and located by *get(key)* hashing primitives in a bounded number of routing hops.

The routing algorithms of content-addressable systems base on the algorithm, proposed by Plaxton et. al. in [5]. The main idea of Plaxton algorithm is in correcting each time a single digit of address. For example, if node *1234* receives a message that should be routed to node *1278* (the first two digits already match), the algorithm forwards it, say, to node *1275* (the first three digits will match). In a P2P version of Plaxton algorithm, a message is constantly forwarded to a node, whose *nodeid* is closer to the message *key* than the current node.

Although the routing algorithms of DHT-based systems clearly outperform the routing algorithms of the first P2P systems, they basically rely on hashing interface primitives of *put(key,value)* and *get(key)*. Thus, one of their major limitations is support only in exact-match lookups, i.e., only the searches, specifying the exact term used at the insertion of a *key*, will succeed in locating it.

In the domains with no accepted naming standards, particularly in general-purpose E-Commerce applications, the peers will probably use different terms to describe the same object. This will lead to difficulties in a resource location. Thus, to develop a general infrastructure for E-Commerce ads, the ads should be handled in a semi-structured form of properties and values. To achieve that, more complex kind of P2P network should be developed, built upon peers using implicit schemas to describe the objects. This approach is further referred as semantic or ontology-based approach.

2.1 Ontology-Based Approach

According to [4], ontology is a formal shared conceptualization of a particular domain of interest. It can act as a standardized reference model, providing a baseline for shared understanding of a domain. The existing approaches of ontology-based information access assume a setting, where information providers share an ontology used to access the data. This technique of shared ontology was implemented in HyperCup [9].

In HyperCup, the peers are connected in a hypercube-like network. The hypercube structure is chosen due to its logarithmic diameter and increased fault tolerance. Moreover, this topology leads to a symmetric structure (as each node holds equal functionality), and to load partitioning in the network. The hypercube dimension *d* and the range of possible values in each dimension (further referred as a coordinates range) *k* determine the maximal number of nodes in the hypercube.

Any edge, connecting a pair of nodes is assigned a numeric value (rank). The routing algorithm of HyperCup is based on forwarding a message only to edges, whose ranks are higher than the rank of the edge the message was received from. This guarantees receiving of a message exactly once, and reaching any connected node in *O(k)* routing hops.

HyperCup also proposes a distributed algorithm for hypercube construction and maintenance. The algorithm is based on the idea that a peer in the hypercube can manage a number of vertices. This is required to “simulate” the missing peers in the implicitly preserved topology of the next biggest complete hypercube. Detailed description and examples of HyperCup algorithms can be found in [9].

The connected peers can be categorized as providers of content, associated with a particular topic. Peers, providing the same or similar contents, are organized in concept clusters using a global predefined ontology defining the relations between the topics. The concept clusters are organized in a hypercube-graph topology and allow querying the generated topology using the discussed routing algorithm.

However, this global predefined ontology constitutes a serious drawback of HyperCup approach. For general-purpose E-Commerce application, the development of all-inclusive ontology will be required. The issue of constructing a global ontology is a very controversial issue [11]. Besides the philosophical question, it will lead to a bottleneck, since further updates and extensions of the ontology will involve a central point of management, unacceptable in P2P networks. Moreover, existence of global ontology will force the peers to explicitly use it, contradicting a "free" decentralized spirit of P2P networks. We resolved these issues by developing a notion of Unspecified Ontology as a method for resources organization in hypercube-like graph structure.

3. Generalization of Ontology to an Unspecified Ontology

In this work we consider a different approach to solve the problem of constructing global ontology that should from one hand grow dynamically with no limited range, from other hand to be used and updated in a fully distributive way. First, we will characterize ontology as a data structure. Any ontology can be viewed as a vector, whose slots correspond to the features of the object being described, and the range of each slot is the set of all possible values of this feature.

For example, consider a simple ontology for cars domain, where the object is described by a vector containing three slots [*manufacturer* | *engine volume* | *year of production*]. Each slot has the following range of values [*{Ford, Mercedes, Jaguar}* | *{1000-1500, 1500-2000, 2000-2500}* | *{1990+, 2000+, new}*]. The semantic P2P system, based on this ontology, will consist of a hypercube of at most *27* vertices, each having up to *6* neighbors.

A generalization of ontology to the unspecified ontology is performed as follows:

- The range of possible slot values can be made unlimited by operating a hashing on the values of each slot, instead of defining a set of fixed values. For example, in the previous ontology we can use hashing to a range of size three, mapping the values to their locations, e.g., [*BMW* | *3000* | *1987*] will be mapped to [*hash(BMW)* | *hash(3000)* | *hash(1987)*]. Consequently, the same *3*-dimensional cube can be used to hold the items, whose values where not anticipated by the slots of a fixed ontology and each user can independently insert new items in a fully distributive way.

- The list of vectors can dynamically grow by letting the peers to expand to the ontology. An unspecified vector (namely, a new vector added to the ad) can be defined as a list of pairs $feature_i$:$value_i$. Two different hash functions are used to map an unspecified vector to the hypercube: one to map the $feature_i$ to a slot number (coordinate in a hypercube), and another one to map the $value_i$ to a numeric value of a slot. For example, an ad using an unspecified vector in cars ontology can be [*ford* | *1600* | *2000*] + [*location*:*LA* | *gear*:*automatic*]. The unspecified vector will be placed in the hypercube by applying $hash_1$*(location)* and $hash_1$*(gear)* to obtain the coordinates in the hypercube, and $hash_2$*(LA)* and $hash_2$*(automatic)* will determine the value in each coordinate. Using two hash functions allows ignoring the order of the unspecified slots in any given vector.

- Multiple vectors can be organized in a "hierarchical" multi-layered structure, instead of one "flat" vector, constructing the underlying semantic hypercube, whose vertices recursively contain another hypercubes. This is further regarded as multi-layered hypercube (MLH). For example, a *3*-layered ontology with three vectors [*a* | *b* | *c*] + [*d* | *e* | *f*] + [*g* | *h* | *i*] with two possible values for each slot will generate a hypercube with *8* vertices where each vertex recursively holds an additional sub-cube (fig. 1). This should be compared to *512*-vertices hypercube, had we used one flat vector for the whole ontology.

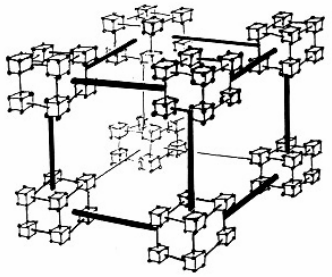

Fig. 1. Multi-Layered Hypercube (MLH)

- Every user can mention a different number of the unspecified pairs. When an unspecified vector is mapped to an existing hypercube, its slots are extended to the maximal dimension of the MLH. For example, the unspecified vector [*location:LA*] is extended to, say, [*location:LA* | *gear:**], when it is inserted to a *2*-dimensional hypercube. If the inserted unspecified vector has more slots than the number of dimensions in the current MLH, all the vectors in the relevant sub-cube are expanded. We assume that these expansions are infrequent, thus, they are feasible (this was confirmed in the experiments).

The above ways of extending a fixed specified ontology to the Unspecified Ontology are summarized in figure 2. Predefined set of slots and values in a fixed ontology is mapped to the location in the underlying hypercube graph. On contrary, in UNSO the number of $feature_i$:$value_i$ pairs in the unspecified part of the ontological description is unlimited. Thus, UNSO dynamically generates a hypercube-like graph structure, where each vertex is recursively constructed of another hypercube.

The generated MLH can act as an infrastructure for the algorithms, developed in HyperCup. During the insertion of ad to the hypercube, it is forwarded to its proper location. The further querying is enabled through the routing technique proposed in HyperCup. The routing will consist of two stages: routing in the primary hypercube to a vertex, containing the relevant type of ads, and further routing inside the secondary MLH to reach a particular ad. In both stages the routing is performed in a manner of semantic routing of HyperCup.

In comparison to the DHT-based systems, UNSO proposes earlier not supported functionalities. Hashing mechanism of DHT-based systems is based on a single key, thus, successful search in DHT systems will require exact matching of the keys. This is not reasonable when the key is a natural language description of an object. On contrary, UNSO hashes a list of features and values, enabling a search of a part of object features, while the order of the specified features is insignificant.

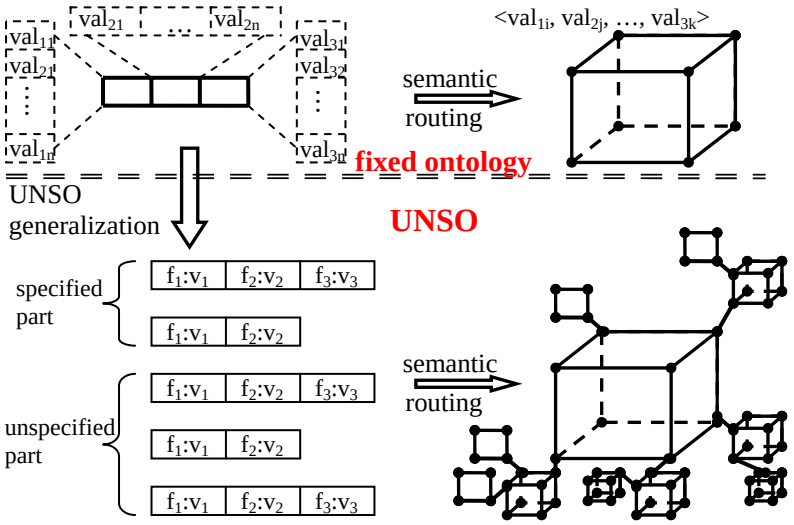

Fig. 2. Generalization of the Fixed Ontology to the Unspecified Ontology

Comparing UNSO to HyperCup shows another important property. HyperCup is based on fixed predefined ontology that must be explicitly used by the peers. On contrary, UNSO users can provide relatively free description of an object in the form of $feature_i$:$value_i$ pairs list. This allows adding new transactions, products, and ontology slots, not requiring update of the ontology and broadcast of the update to maintain consistency. Using unspecified features in the MLH allows distinguishing between ads from different domains.

4. Implementation Details

In this section we discuss the details of UNSO model that was implemented to conduct the experiments. A multi-layered UNSO model, similar to the model, described in the previous section, was implemented. The primary layer of UNSO contained the hypercube constructed by the fixed parts of ontological descriptions of the ads. The secondary MLHs were generated by the unspecified parts of the description.

The specified part of UNSO is the portion of the object description that should by explicitly specified by a peer when inserting an ad to the system. As the targeted application resources are general-purpose E-Commerce ads, all the slots of the specified part of the ontology should be applicable to as wide as possible variety of objects (universal slots). Note that here we relate to real-life tradable objects only (e.g., cars, apartments, books, tickets etc…), which can be a subject of E-Commerce ad. Thus, finding the slots of the specified part of the ontology (common properties for a large group of absolutely different objects) is a complicated task. Clearly, these properties can be only the universal characteristics of the objects, such as size, weight, price, material and so forth.

The following subset of the universal properties of objects was chosen to serve as the slots of the specified part of the ontology:

- *Product* – the original name of the product, i.e., the noun describing the group of objects, the described object is related to. For example, car, book, telephone, computer and so forth.

- *Relative size* – the size of the product. Since the size is a relative concept, we have to provide a unit of size, in relation to which the size of an object will be measured. The unit was chosen to be the average size of human being, and the objects were graded from a ‘very small’ to a ‘very big’ size in comparison to the size of human being. According to this partition, book or telephone are categorized as ‘very small’, while house or truck are ‘very big’.

- *Usage range* – the distance from the current location of the user, where the object can be operated. As this concept is also a relative one, a scale for the possible values should be provided. The usage distance was defined to be ‘very close’, if the objects will be operated “at arm’s reach” relatively to the user, and ‘very far’ if the range of operation is roughly unlimited. Thus, housewares or furniture are categorized as ‘very close’ objects, while car or optical equipment are ‘very far’.

- *Price* – the price of the product, i.e., the amount of money the seller expects to get in exchange for an object, or the amount of money the buyer is ready to pay in exchange for it. Note that price is also a universal feature as it separates between different types of objects, such as houses and cars (though it may fail to separate books from CDs).

The values mentioned by the user in the ontological description of an object, determine the location of the ad in the underlying MLH. The dimension d of the primary hypercube equals to the number of slots in the specified part of the ontology. In particular, the above mentioned four slots determine the first vector of UNSO, yielding a 4-dimensional primary hypercube. The number of dimensions in the secondary hypercubes is theoretically unlimited. Practically, a number of features that can be mentioned by the peers in the unspecified part of ontological description is bounded. Thus, using Hypercup’s approach of simulating missing vertices by the existing vertices (until a connected peer), we were able to handle a generated MLH.
In HyperCup, the mapping of the object description to the location in the hypercube was performed according to the values of the ontology slots. On contrary, in this work we operated a set of hash functions for this purpose. Thus, the range of possible values (coordinates range) k in the slots product and price was determined by the range of the hash functions. In the slots relative size and usage range we restricted the set of values to be

{very small, small, medium, big, very big}. Thus, the range of values in these coordinates was 5 and the total number of possible vertices in a primary hypercube constructed by the above ontology in 25k2, where k is the coordinates range.

One of the advantages of using hashing mapping mechanism over a predefined ontological mapping is uniform distribution of ads among the MLH, ensuring equivalent load partitioning. For example, for k=17 specified vector [cars | very big | very far | 45000] was mapped to the location [0 | 4 | 4 | 13], while the location of [televisions | medium | very close | 400] was [15 | 2 | 0 | 2]. But for k=11 the same vectors were mapped to other locations, respectively [7 | 4 | 4 | 8] and [6 | 2 | 0 | 1].

As for the unspecified part of the ontology, the format is a list of pairs of the form feature:value, where neither feature, nor value are limited by any predefined ontology. For example, consider the following ontological (both specified and unspecified) descriptions of a car and a book: [car | big | very_far | 5000] + [manufacturer:BMW | color:red | mileage:5000] and [book | small | very_close | 20] + [author:Tolkien | name:"Lord of the Rings" | pages:250 | edition:2]. The unspecified part [manufacturer:BMW | color:red | mileage:5000] mentioned in the car ad is mapped to the location (5, 8, 4) if the feature 'manufacturer' is mapped to the coordinate number 1 and the value 'BMW' is mapped to the value 5, 'color' is mapped to coordinate 2 and 'red' mapped to value 8, and respectively 'mileage' is mapped to coordinate 3 and '5000' is mapped to value 4. During the search operation, the ads of the collided domains are filtered according to the features and values, mentioned in the query.

The IP address of the peer inserting the ad was chosen to act as the unique feature distinguishing between the identical ads. Since the IP address is not a real property of the object, described in the ad, it acts just as a distinguishing feature and is ignored when displaying the query results.

Every new feature, mentioned in the ads of a particular domain, increases the dimension of the respective MLH. For example, in the above example of [manufacturer:BMW | color:red | mileage:5000], inserting a new ad, specifying two features that were not mentioned yet, will cause the secondary hypercube to expand to a 5-dimensional hypercube. So, the original ad should be remapped to the location (5, 8, 4, *, *).

However, analyzing relatively small corpus of car ads shows that the users that inserted the ads mentioned totally about 10 different features, while most of the ads contained only 3-4 pairs of features and values. Thus, most of the expansions will occur during the initial phases of secondary MLHs construction. In this stage a hypercube still contains relatively low number of ads, consequently the update is not too expensive. On the other hand, in a hypercube containing many ads, the updates will be infrequent.

A possible drawback of the proposed idea is the unclarity and possible ambiguity in supplying the values of the ontology slots. For instance, the peers publishing a message can describe a car as 'big', while the other one describe it as 'very big'. In this case the ads will be mapped to the different vertices of the hypercube, and will not match each other. We believe that most of the peers will use similar patterns of thinking and will succeed in properly describing their objects. Using common terms most of the ads describing the same type of objects will be mapped to the same area in the MLH thus forming "clusters" of similar ads as depicted in figure 3. Once a search has located a suitable node in the MLH, it can return the ads which are in a certain radius from the current node. In this work the search radius is one, so the answer to a query will contain all the ads that were mapped to the target vertex only. Note that two different ads can be mapped to the same vertex if all their ontological vectors are identical, or all their slots values collides to the same hash values.

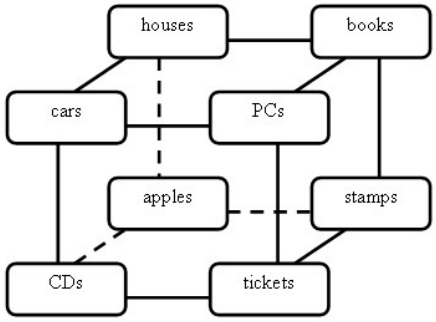

Fig. 3. Cluster of Ads in a 3-Dimensional Primary Hypercube

To resolve a problem of using different terms with the same meaning while describing the same object, the product names were standardized by WordNet [9]. 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. In this implementation we employed standardization, substituting the original term by its most frequent synonym. Thus, the similar, but not the same terms, specified by a user in the product field of the fixed part of the ontology, were replaced by a 'representing' term.

In order to balance traffic load across the system, slightly modified variant of Plaxton routing algorithm [19] was implemented. Plaxton algorithm routes a message from a location A to a location B by correcting each time a single digit of the address. In the original algorithm the order of address digits corrections is known, i.e., the most significant digit of the address is corrected first, and the least significant digit is corrected last. Simultaneous routing of a number of messages to/from the same area (cluster) might cause a communication bottleneck. To resolve this problem we randomized the order of address digits corrections, increasing the number of possible routing paths and balancing the traffic load of the connected peers. This approach will be further referred as 'randomized' Plaxton routing algorithm.

In the rest of this section we will describe the search scenarios in the implemented system. The system implementation included Graphical User Interface (GUI) enabling a user to insert the search queries (see figure 5). The GUI contained fields for inserting the values of the fixed ontology slots, textual fields for inserting the unspecified features and values, and textual area for displaying the answers to the query.

In a typical search scenario the user will fill the values of the fixed slots of the ontology. Assuming common patterns of thinking, the system will succeed to find the vertex, holding the ads of the relevant domain. For a big enough corpora of ads, the

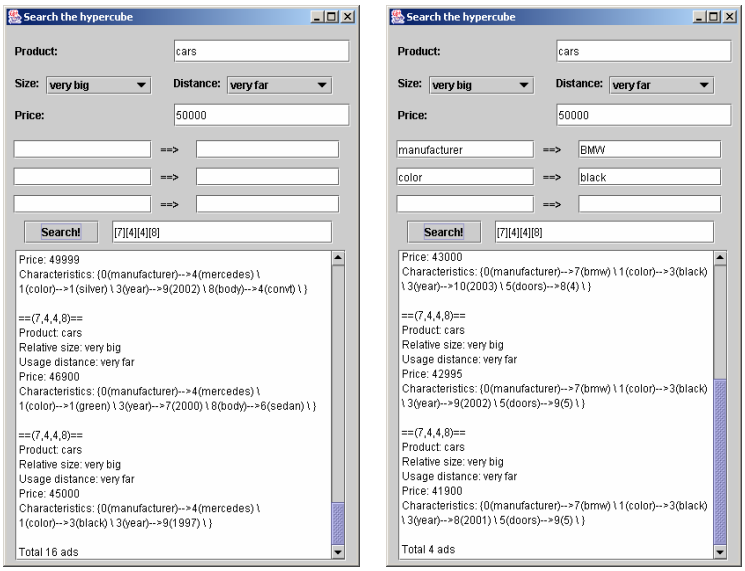

Fig. 4. Typical Search Scenario: (Left) Only the slots of the fixed part of the ontology are specified; (Right) In addition, two unspecified slots are specified.

number of ads in the answer to the query might be too high. Thus, the user will be forced to refine the query by mentioning the unspecified features and values. Consequently, the system will display to the user a shorter list of ads.

Consider the search scenario, depicted in figure 4. At the first query, depicted in figure 4-Left, only the slots of the fixed ontology were filled: [car | very_big | very_far | 50000]. The query was mapped to a vertex [7 | 4 | 4 | 8] of the primary hypercube (for k=11). As no unspecified features were mentioned, the number of returned ads was 16 (out of 34 cars ads in the corpus), which is relatively high. At the second query, the user expanded the description using two unspecified features: manufacturer and color. The search for [car | very_big | very_far | 50000] + [manufacturer:BMW | color:black] was now mapped to [7 | 4 | 4 | 8] + [7 | 3 | * | * | *]. Note that since the secondary level consisted of a 5-dimensional hypercube, the system automatically inserted wildcards instead of not mentioned slots. The use of unspecified features filtered the irrelevant ads, returning only four ads, as depicted in figure 4-Right.

In the next section we will analyse the experimental results and discuss the factors influencing the performance of UNSO.

5. Experimental Results

To conduct the experiments, a corpus of E-Commerce real-life ads of both supply and demand types was taken from *www.recycler.com* site. The corpus of supply ads consisted of *1272* ads from different categories. Before inserting the ads to the system, each ad was manually converted to the form of ontological description. For example, an ad “Philips 50FD995 50" plasma television, brand new in box, $4800” was converted to the following description (both fixed and unspecified): [*television* | *medium* | *very_close* | *4800*] + [*manufacturer:Philips* | *model: 50FD995* | *size:50"* | *type:plasma* | *condition:brand new*]. The conversions were done as close as possible to the original ads to mimic the insertions of ads by naïve users. The demand ads were built by partially changing the features and values of the supply ads.

5.1 Information Retrieval Metrics

In this experiment two traditional metrics of Information Retrieval, *recall* and *precision*, were evaluated [12]. In context of publish-locate application, precision can be defined as the number of relevant ads found in the target vertex, divided by the total number of ads in that vertex. Similarly, recall can be calculated by dividing the number of relevant ads found in the target vertex by the total number of relevant ads. The values of precision and recall were measured for different values of *k* (from *1* to *100*).

The chart in figure 5 shows the values of precision and recall as a function of *k*. The dashed lines show the precision curves, while the continuous curves stand for the recall. Each one of these metrics was measured twice: first for the original terms specified in the ads, and then after standardizing the values of the *product* slot with WordNet.

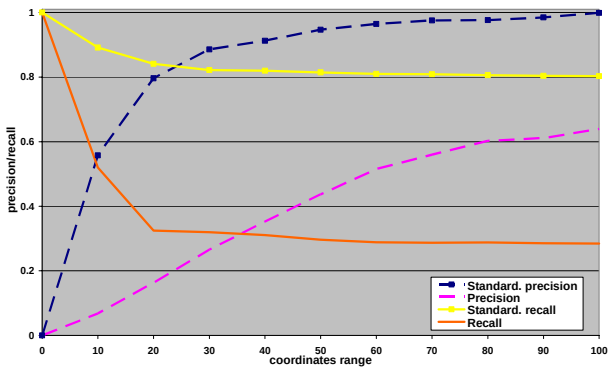

Fig. 5. Recall and Precision as a Function of the Coordinates Range *k*

As *k* increases, the probability of an ad to be occasionally mapped to a given vertex decreases. This causes the precision to ascend with the coordinates range *k*. It can be clearly seen from the chart, that the standardized results outperform the original results. This is reasonable, since WordNet converts the similar terms to the same representing term. Thus the number of different types of ads in the system decreases, and the number of ads occasionally mapped to a given vertex, decreases.

The original recall rate of the system is relatively low. This can be explained by the fact that without the standardization the peers might use different terms to describe an object. Thus, every search will return only a part of the relevant ads. Using WordNet, the recall value is significantly higher. Note that the recall curves initially decrease when the coordinates range *k* rises. This can be explained by the fact that the probability of similar ads to be mapped to the same vertex is higher for low values of *k*.

Obviously, there is a trade-off between the value of *k* and recall/precision of the system. On one hand the recall converges starting from relatively low values of *k*, and the precision increases with *k*. However, the maintaining 'big' MLHs will be difficult for a low number of connected peers. Let us compare two possible coordinate ranges: *k=30* and *k=80*. The values of recall and precision for the above *k* values are close: the precision is *0.89* and *0.97*, while the recall roughly remains unchanged *0.81*. But, the estimated size of the primary hypercubes increases dramatically, and every connected peer will simulate about *50* times more vertices, resulting in high load of the peers and heavy traffic in the network. Therefore, the optimal value of *k* for a moderate number of connected peers is around *20*. For this value of *k* both precision and recall are about *0.8*, while the size of the primary hypercube remains reasonable.

5.2 Locality

An important measure of UNSO's quality is locality, i.e., mapping of similar ads to the close locations. To check the locality, we modified *j* values in the ontological descriptions of a subset of ads, and measured the editing distance *d* between the locations of the original and the modified ads. The editing distance *d* between the locations of two given ads in the MLH is defined as a sum of the editing distances (i.e., differences in coordinates) in each one of the dimensions. For example, the distance between (*1, 3, 5, 7*) and (*8, 6, 4, 2*) in a *4*-dimensional hypercube is *7+3+1+5=16*. In MLH topology, editing distance between two locations is the sum of editing distances in each one of the layers. The measures were conducted for the different values of *k* (from *1* to *100*). The chart in figure 6 shows the editing distance *d* as a function of *j*.

Each quadruplet of bars in the chart shows the average editing distance between the ads for *j=1* (leftmost bar), *2*, *3* and *4* (leftmost bar). It can be seen that for any coordinates range *k*, the editing distance increases with the number of changes inserted to the original ad. For example, for *k=55*, the editing distance for one change is *12.8*, while for four changes it is *45.4*. This shows that the property of locality holds in UNSO: distance between similar ads is significantly lower, than distance between diverse ads.

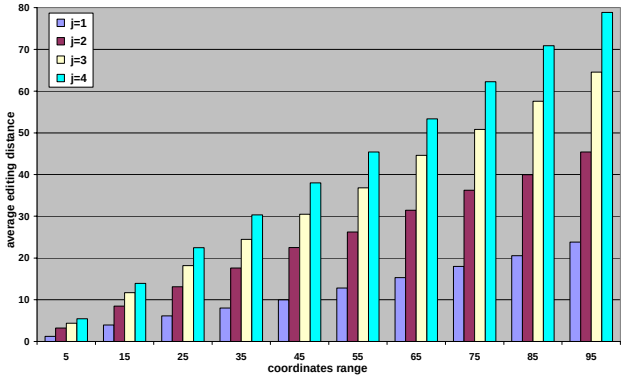

Fig. 6. Average Editing Distance as a Function of a Number of Changes (j) for Different Values of the Coordinates Range *k*

5.3 Stability

In this experiment we measured the "average size" of the generated network as a function of the number of inserted ads. The key for evaluating the size of the network is the characteristic path length. For any two given ads A and B, a path length between their locations in the hypercube can be measured. According to Plaxton algorithm [5], routing from A to B can be performed by correcting each time a single digit of the address. In terms of MLH, each step of the routing algorithm should correct a single coordinate in one of the dimensions of one of the hypercubes. Thus, a path length can be defined as a minimal number of corrections needed to reach the location of B starting at the location of A. The characteristic path length of the network is defined as the average of path lengths over all nodes in the network. In this experiment we gradually increased the number of ads in the system and calculated the characteristic path length. The results of the experiment are depicted in figure 7.

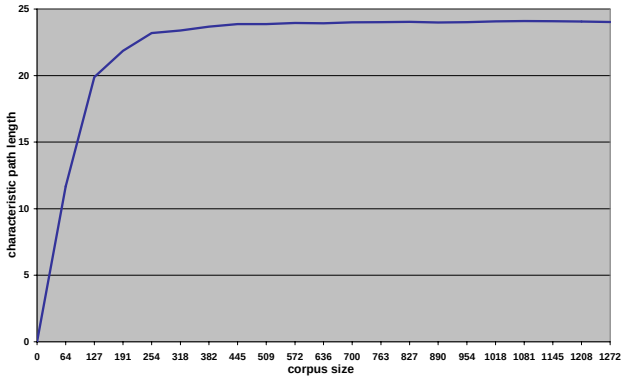

Fig. 7. Characteristic Path Length as a Function of the Corpus Size

The form of the curve shows that the initial insertions of ads enlarge the hypercube and increase the number of dimensions in the secondary MLHs. However, starting from approximately 20% of the corpus size, inserting additional ads roughly will not change the characteristic path length. This supports the assumption that the total size of the set of features that might be used while describing an object is final and bounded. Moreover, the major part of this set is used in the initial descriptions, while the rest of the descriptions contribute a very few new characteristics. Therefore, the proposed algorithm of secondary

MLHs expansion during the insertion of higher-dimensional ads will not cause a significant overhead due to the expansion.

6 Conclusions and Future Work

In this work we developed a novel notion of Unspecified Ontology (UNSO). Comparing to the fixed ontology, UNSO approach proposes a more flexible way to describe an object. It allows constructing a multi-layered hypercube (MLH) graph topology, supporting efficient semantic routing. Using UNSO does not force peers to share or to use any explicit ontology. A peer can provide relatively free ontological description of an object, specifying the pairs of object features and the respective values. Instead of using strict ontological mapping, hashing is used to map the ads to their location in the hypercube. Thus, the order of features, mentioned by the user is insignificant, and they will be mapped to the same hypercube coordinates in any order.

Experiments show a good performance of UNSO. The precision of the system is high, and it increases with the rise of the hypercube coordinates range. Recall values are moderate, and they become very high as a result of performing a simple standardization. Unlike in classical Information Retrieval systems and search engines, high values of both recall and precision (at the same time) can be obtained. From a P2P point of view UNSO also handles a good performance. The approach is highly scalable, as for a relatively low number of concurrent routings, the system reaches its maximal traffic load, and further routing operations do not increase it. Sophisticated standardization tools and dynamic hash functions, uniformly distributing the ads among the hypercube, will help obtaining better results.

The following directions of further research are of particular interest:

- Integrating sophisticated Natural Language Processing (NLP) tools to improve the IR performance of the system.
- Discovering the weights of object features and the distances between the different values of the same feature for the results ranking.
- Developing a smarter routing algorithm, dynamically generating the optimal routing path as a function of changing workloads in the network.
- Adding E-Commerce functionalities to the system. For example, developing UNSO-based systems, conducting public auctions or market clearing.
- Implementing real P2P client, launching it over the Web, and performing large-scale experiments with real users and real ads.

We believe that the flexible nature of UNSO will facilitate UNSO usage not only for E-Commerce applications, but in a wide range of publish-locate applications.

References

1. E.Adar, B.Huberman, “Free riding on Gnutella”, Technical report, Xerox PARC, 2000.
2. I.Clarke, O.Sandberg, B.Wiley, T.Hong, “Freenet: A distributed anonymous information storage and retrieval system”, In proceedings of the ICSI Workshop on Design Issues in Anonymity and Unobservability, 2000.
3. C.Fellbaum, “WordNet - An Electronic Lexical Database”, The MIT Press Publishers, 1998.
4. T.R.Gruber, “A translation approach to portable ontology specifications”, Knowledge Acquisition Journal, 6(2), pp. 199–221, 1993.
5. C.Plaxton, R.Rajaraman, A.Richa, “Accessing nearby copies of replicated objects in a distributed environment”, In proceedings of ACM SPAA, 1997.
6. S.Ratnasamy, P.Francis, M.Handley, R.Karp, S.Shenker, “A Scalable Content-Addressable Network”, In proceedings of ACM SIGCOMM, 2001.
7. S.Ratnasamy, S.Shenker, I.Stoica, “Routing Algorithms for DHTs: Some Open Questions”, In proceedings of the 1st International Workshop on P2P Systems, 2002.
8. A.Rowstron, P.Druschel, “Pastry: Scalable, distributed object location and routing for large-scale P2P systems”, In proceedings of the International Conference on Distributed Systems Platforms, 2001.
9. M.Schlosser, M.Sintek, S.Decker, W.Nejdl, “A scalable and ontology-based P2P infrastructure for semantic Web services”, In proceedings of the 2nd IEEE Conference on P2P Computing, 2002.
10. I.Stoica, R.Morris, D.Karger, M.F.Kaashoek, H.Balakrishan, “Chord: A scalable Peer-to-Peer lookup service for internet applications”, In proceedings of ACM SIGCOMM, 2001.
11. M.Uschold, “Creating, integrating and maintaining local and global ontologies”, In proceedings of the 14th European Conference on Artificial Intelligence, 2000.
12. I.H.Witten, A.Moffat, T.C.Bell, “Managing gigabytes: Compressing and Indexing Documents and Images”, Morgan Kaufmann Publishers, 1999.